DB2 DataJoiner® for UNIX® Systems

Planning, Installation, and Configuration Guide

Version 2 Release 1 Modification 1

SC26-9145-02

DB2 DataJoiner® for UNIX® Systems

Planning, Installation, and Configuration Guide

Version 2 Release 1 Modification 1

SC26-9145-02

Note

Before using this information and the product it supports, read the information in "Notices" on page 371.

Third Edition (October 1999)

This edition replaces SC26-9145-01.

Contents

Welcome to DataJoiner

This book provides you with planning information and installation and configuration procedures for DataJoiner for UNIX. This book is for database administrators, system administrators, and any other knowledge workers who have responsibility for administering and operating in a heterogeneous environment.

Part 1 gives a brief overview of the functionality provided by DataJoiner and lists the new features for this version.

Part 2 provides information about planning for and installing DataJoiner.

Part 3 describes how to configure different data sources to communicate with DataJoiner.

Part 4 provides information about configuring communications between DataJoiner as a DRDA Application Server (AS).

Part 5 describes how to maintain DataJoiner.

Part 6 contains information about replication, including planning, configuring, defining, replication sources and targets, and maintaining replication objects.

Part 7 contains information about DRDA AS bind options, national language support, and available education and service for DataJoiner.

Highlighting Conventions

This book uses these highlighting conventions:

Boldface type
Indicates commands and graphical user interface (GUI) controls (for example, names of fields, names of folders, menu choices). Boldface type also indicates examples of SQL keywords in the *Application Programming and SQL Reference Supplement*.

`Monospace type`
Indicates examples of coding or of text that you type.

Italic type
Indicates variables that you should replace with a value. Italic type also indicates book titles and emphasizes words.

UPPERCASE TYPE
: Indicates SQL keywords and names of objects (for example, tables, views, and servers).

Graphical Conventions

This book uses these graphical conventions:

This icon marks a fast path — fast paths tell you if you can skip sections to get to the next ″how to″ information. Fast path also indicate where you should go next.

This icon marks a tip — additional information that can help you complete a task.

How to Read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this book:

Arrow symbols

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

▶▶── Indicates the beginning of a statement.

──▶ Indicates that the statement syntax is continued on the next line.

▶── Indicates that a statement is continued from the previous line.

──▶◀ Indicates the end of a statement.

Diagrams of syntactical units other than complete statements start with the ▶── symbol and end with the ──▶ symbol.

Conventions

- SQL commands appear in uppercase.
- Variables appear in italics (for example, *column-name*). They represent user-defined parameters or suboptions.
- When entering commands, separate parameters and keywords by at least one blank if there is no intervening punctuation.
- Enter punctuation marks (slashes, commas, periods, parentheses, quotation marks, equal signs) and numbers exactly as given.
- Footnotes are shown by a number in parentheses, for example, (1).
- A ƀ symbol indicates one blank position.

Required items

Required items appear on the horizontal line (the main path).

```
►►──REQUIRED-ITEM──────────────────────────────────────────────────────►◄
```

Optional items

Optional items appear below the main path.

```
►►──REQUIRED-ITEM──┬───────────────┬───────────────────────────────────►◄
                   └─optional-item─┘
```

If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

```
                   ┌─optional-item─┐
►►──REQUIRED-ITEM──┴───────────────┴───────────────────────────────────►◄
```

Multiple required or optional items

If you can choose from two or more items, they appear vertically in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.

```
►►───┬─REQUIRED-ITEM────┬──────────────────────────────────────────────►◄
     ├─required-choice1─┤
     └─required-choice2─┘
```

If choosing one of the items is optional, the entire stack appears below the main path.

```
►►───┬──────────────────┬──────────────────────────────────────────────►◄
     ├─optional-choice1─┤
     └─optional-choice2─┘
```

Repeatable items

An arrow returning to the left above the main line indicates that an item can be repeated.

```
     ┌────────────────────────────────────┐
     ▼                                    │
►►─────REQUIRED-ITEM──repeatable-item─────┴────────────────────────────►◄
```

If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
       ┌─,──────────────────────────────┐
       ▼                                │
►►─────REQUIRED-ITEM──repeatable-item───┴──────────────────────────────────────►◄
```

A repeat arrow above a stack indicates that you can specify more than one of the choices in the stack.

Default keywords

IBM-supplied default keywords appear above the main path, and the remaining choices are shown below the main path. In the parameter list following the syntax diagram, the default choices are underlined.

```
       ┌─default-choice───┐
►►─────┼──────────────────┼────────────────────────────────────────────────────►◄
       ├─optional-choice1─┤
       └─optional-choice2─┘
```

How to Send Your Comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DataJoiner documentation. You can use any of the following methods to provide comments:

- Send your comments from the Web. Visit the Web site at http://www.ibm.com/software/data/datajoiner

  The Web site has a feedback page that you can use to enter and send comments.
- Send your comments by e-mail to comments@vnet.ibm.com. Be sure to include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).
- Complete the readers’ comment form at the back of this book and return it by mail, by fax (800-426-7773 for the United States and Canada), or by giving it to an IBM representative.

Part 1. Introduction to DataJoiner

Chapter 1. About DataJoiner

IBM's DataJoiner allows you to access data residing on multiple and diverse platforms, both IBM and multi-vendor, relational and non-relational, as a single database image. With DataJoiner, you can access all the data in your enterprise as if it were local.

DataJoiner V2.1.1 consists of DB2 Version 2 products and books, additional multi-database functionality, replication functionality, and other IBM products, including:

The DataJoiner server
A *relational* database management system containing functions and tools that you can use to create, update, control, and manage relational databases using Structured Query Language (SQL)

Software Developer's Kit (SDK) functionality
A collection of tools for database application developers

Net.Data
A Web server gateway that application developers can use to create Internet applications that access data from DB2 database

The replication administration tool
The administrative component of heterogeneous replication, which you can use to define and prepare relational data sources and targets for replication. The replication administration tool runs on a Windows 95 or Windows NT client

Apply program
The replication component that provides the actual replication functionality

Spatial Extender
The component that enables DataJoiner to store and manipulate geographic data types

Documentation
The entire DataJoiner library and relevant DB2 books in HTML and PostScript formats

In addition to these components:

- Remote clients need the **DB2 Client Application Enabler** (CAE). With the DB2 CAE, client workstations from a variety of platforms can connect to any DataJoiner server. The DB2 CAE CD-ROM is included with DataJoiner.

- If a DB2 client is a replication source, the Capture program must be installed on the client.

Working with Data Using DataJoiner

DataJoiner includes graphical tools that you can use to tune performance, access remote DB2 and DataJoiner servers, manage all servers from a single site, develop powerful applications, and process SQL queries.

DataJoiner includes the DB2 database engine and the Administrator's Toolkit, which provide the following components:

- The **Command Line Processor**, used to access and manipulate databases from the system command prompt
- The **Database Director** tool, used for administrative tasks such as configuring the system, managing directories, backing up and recovering the system, and managing media
- The DB2 **Performance Monitor**, used to monitor the performance of your DataJoiner system for tuning purposes
- The **Visual Explain** tool, used to graphically view and navigate complex SQL access plans

See "Database Administration Tools" on page 6 for more information about these components.

Accessing DataJoiner from Client Workstations

The DB2 Client Application Enabler provides a run-time environment for client workstations that want to access multiple remote databases. Support is available for clients on the following platforms:

- DOS
- Linux
- Macintosh
- OS/2
- UNIX (AIX, HP-UX, Solaris, SINIX)
- Windows NT, Windows 98, Windows 95, and Windows 3.x

In addition to the clients listed above, VM, MVS, OS/400, and any other DRDA clients are supported through the DRDA application server feature. For more information, see the *DB2 Installation and Operation Guide*.

Accessing Database Data From Remote Clients

DataJoiner can be accessed by remote clients that have the DB2 Client Application Enabler installed. Figure 1 shows a DataJoiner server working with local applications and remote clients.

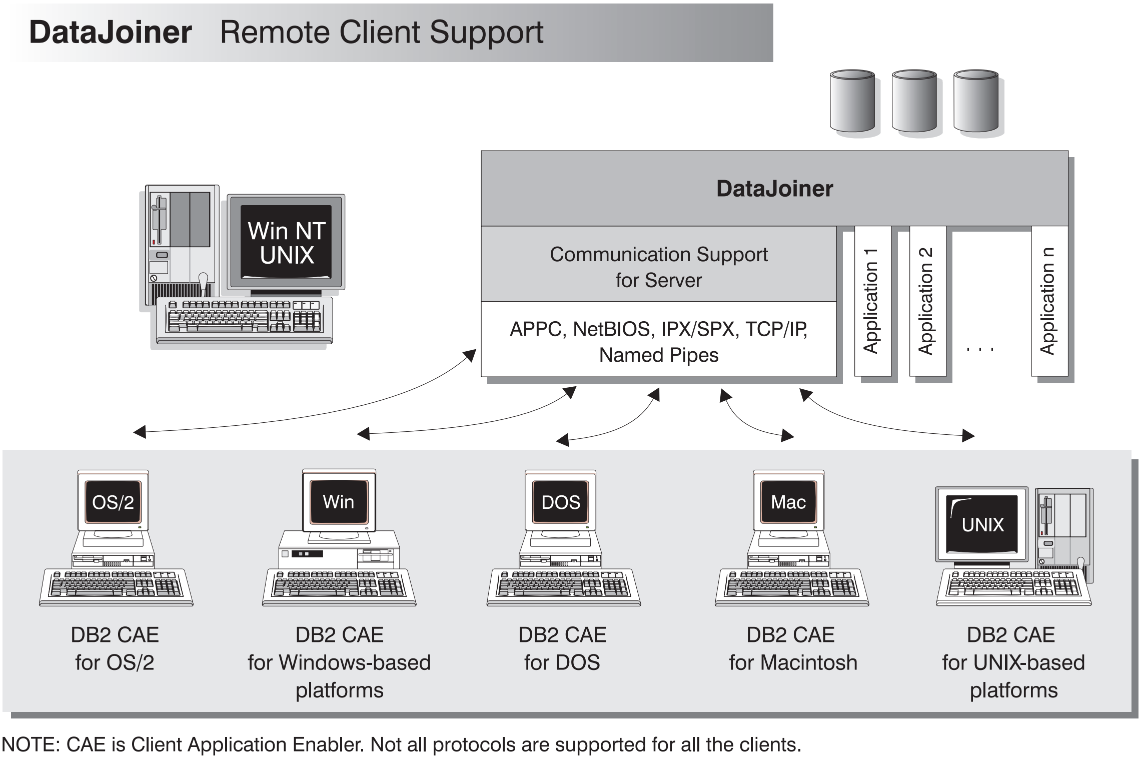
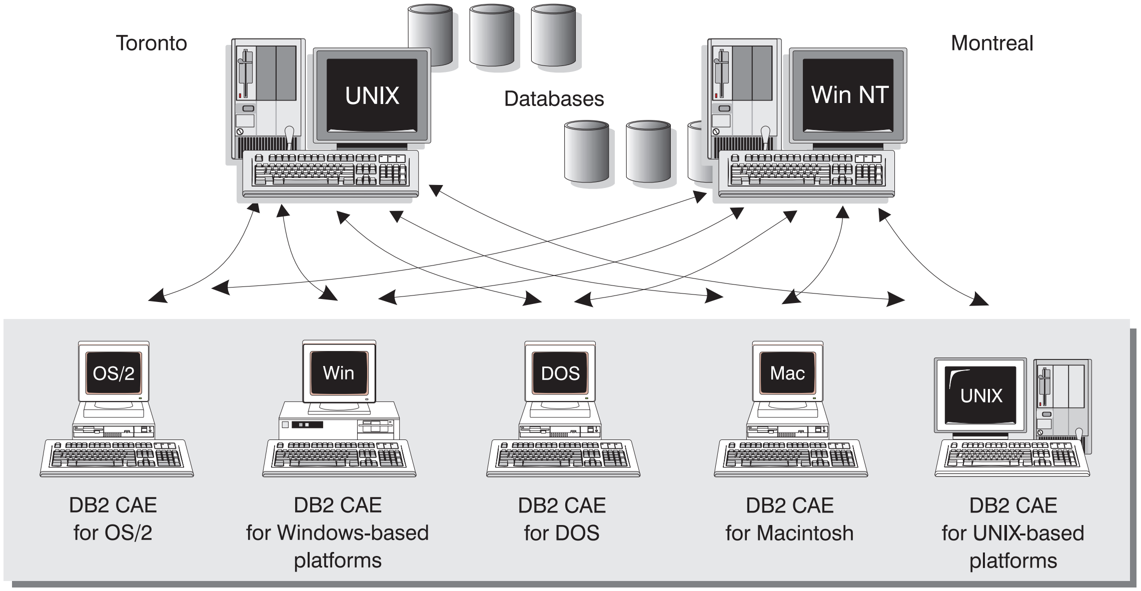

DataJoiner Remote Client Support
Win NT
UNIX
DataJoiner
Communication Support for Server
APPC, NetBIOS, IPX/SPX, TCP/IP, Named Pipes
Application 1
Application 2
. . .
Application n
OS/2
Win
DOS
Mac
UNIX
DB2 CAE for OS/2
DB2 CAE for Windows-based platforms
DB2 CAE for DOS
DB2 CAE for Macintosh
DB2 CAE for UNIX-based platforms
NOTE: CAE is Client Application Enabler. Not all protocols are supported for all the clients.

Figure 1. DataJoiner Server

Accessing Multiple DataJoiner Servers

No additional software is required to enable LAN-to-LAN connections between DataJoiner clients and DataJoiner servers. Suppose you have a DataJoiner server on a Windows NT workstation connected to a LAN located in Montreal and a DataJoiner server on a UNIX workstation connected to a LAN located in Toronto. If there is a connection between the two LANs, clients on either network can access either server. See Figure 2.



NOTE: CAE is Client Application Enabler.

Figure 2. Accessing Multiple DataJoiner Servers

Two-phase commit, or distributed-unit-of-work access, is also supported. Within a single transaction, databases on multiple servers can be accessed and updated with full integrity. See the *DB2 Information and Concepts Guide* for more information about two-phase commits.

Database Administration Tools

You can perform database administration tasks locally or remotely with a collection of tools that helps database administrators (DBAs) manage databases. These tools include:

- The Command Line Processor
- The Database Director
- Visual Explain
- The DB2 Performance Monitor

These tools are described in the following sections.

Note: DataJoiner for Solaris systems does not support the tools included in the Administrator's Toolkit (the Database Director, Visual Explain, and the DB2 Performance Monitor).

The Command Line Processor

The Command Line Processor (CLP) is part of DataJoiner base functionality. Use the CLP to enter commands and SQL statements.

Database Director

The Database Director is included in the Administrator's Toolkit. It is an easy-to-use graphical interface that displays database objects (such as databases, tables, and packages) and their relationship to each other. Using the Database Director, you can select one or more database objects and perform the following tasks:

- Configure databases and database manager instances
- Manage the directories necessary for accessing local and remote databases
- Back up and recover databases or table spaces
- Manage table spaces
- Launch Visual Explain (see "Visual Explain" for information)
- Launch the DB2 Performance Monitor (see "DB2 Performance Monitor" on page 7 for information).

Visual Explain

Visual Explain is included in the Administrator's Toolkit. It is a tool for analyzing and tuning SQL statements. This tool helps database administrators and application developers:

- View the access plan chosen by the database manager's optimizer for a given SQL statement.
- View the details of the access plan including statistics in the system catalogs.
- Determine the source of problems in SQL statements.

The online help Visual Explain contains additional information.

DB2 Performance Monitor

The DB2 Performance Monitor is included in the Administrator's Toolkit. Use it to help you monitor the performance of your DataJoiner system for tuning purposes. With the DB2 Performance Monitor, you can:

- Determine and analyze performance problems in the database manager or database applications.
- Tune SQL statements for better performance.

- Identify exception conditions based on thresholds that you define.
- Define your own statistics, in addition to the default set provided.

You can choose to monitor snapshots or events. Snapshot monitoring captures point-in-time information at specified intervals. Event monitoring records performance information over the duration of an event such as a connection.

Application Development Using the Software Developer's Kit

The Software Developer's Kit is a collection of tools for database application developers. The Software Developer's Kit includes programming libraries, header files, online documentation, and sample programs to build character-based, multimedia, or object-oriented applications.

Software Developer's Kit tools support several programming languages (including COBOL, C, and C++) for application development, and precompilers for the supported languages are provided.

The Software Developer's Kit can be installed with the DataJoiner server or on a separate machine to provide an application development environment from both server and client platforms. When installed on the server, it provides access to both remote and local databases. When installed on the client, it provides access to remote databases.

There is a platform-specific version of the Software Developer's Kit available for each of the supported client and server environments. The DataJoiner CD-ROM contains Software Developer's Kit functionality for supported UNIX platforms. Applications developed with it will run on any client platform where the equivalent DB2 Client Application Enabler component is installed, and can access all DataJoiner and DB2 Servers as well as any other database server that implements the DRDA protocol.

The Software Developer's Kit allows you to develop applications using the following interfaces:

- Embedded SQL
- Call level interface development environment (compatible with Microsoft's ODBC)
- Application programming interfaces (APIs) to access database utilities
- A prototyping environment using the command line processor's interactive SQL

See the *DB2 SDK Building Your Applications* book for your platform for complete information on using the Software Developer's Kit, or for a list of the supported compilers.

Internet Connection Using Net.Data

The fast-growing popularity of the Internet and the World Wide Web has created a demand for Web access to enterprise data. Net.Data is a Web server gateway to IBM's DB2 family. It enables an application developer to build Web applications that access DB2 databases by using HTML forms and dynamic SQL. These applications are then stored on the Web server, and the end user can use any Web browser to access the HTML forms and the reports returning DB2 data.

A DataJoiner application built with Net.Data displays a typical Web page form (an HTML document) to the end user who can select values from a list or type in values to define the query. The user then clicks a push button to submit the query to the Net.Data run-time engine on the Web server. At the Web server, the complete SQL statement is dynamically built with the user inputs and then sent to the database. If user input is not required for the query, a link in the HTML document can trigger a prewritten SQL query and display the resulting report. This type of automated query could be used for repeated access to current DataJoiner data.

A live demonstration of several Net.Data applications is available for frame-enabled browsers on the Web at URL http://www.ibm.com/software/. (This address is for the IBM Software Page. From here, select the Data Management link to find the Net.Data information.) The applications for this demo allow customers to query the CelDial Company database for the status of their accounts, to update account information, and to retrieve information about pending orders.

Net.Data can access databases locally, if it is installed with the DataJoiner server or IBM Internet Connection Server. To run Net.Data as a remote client to a DataJoiner server, you need to install Net.Data and IBM Internet Connection Server on a client workstation. See Figure 3 for an example of this scenario.

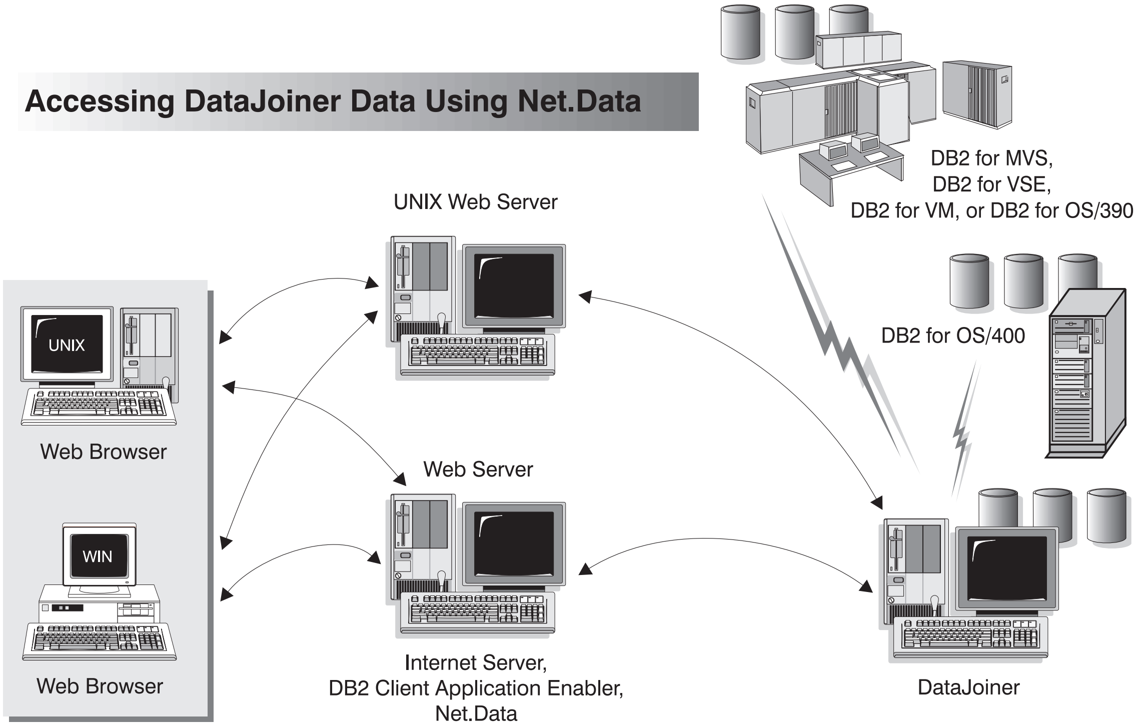


Figure 3. Net.Data

Host Data Access from the Desktop Using DataJoiner DDCS

DataJoiner contains Distributed Database Connection Services (DDCS) functionality, which gives clients on your LAN access to data stored in host systems. A huge amount of organizational data is managed by DB2 on systems such as DB2 for MVS, DB2 for VSE or VM, and DB2 for OS/400. Applications running on any of the supported client platforms can work with this data transparently as if it were managed by a local database server.

DataJoiner DDCS provides applications with transparent online access to data by implementing a standard architecture for managing distributed data. This standard is known as Distributed Relational Database Architecture (DRDA). For example, you can use DataJoiner DDCS with:

- Spreadsheets, such as Lotus 1-2-3 and Microsoft Excel, to analyze real-time data without the cost and complexity of data extract and import procedures.
- Decision support tools, such as Intersolv Q+E Database Editor, and Crystal Reports, to provide real-time information.
- Personal database products, such as Lotus Approach and Microsoft Access.

- Development tools, such as IBM VisualAge C or C++, Micro Focus COBOL, Microsoft Visual C or Visual C++, to create client/server solutions.
- The Replication Administration Tool and the Apply program to propagate data from or to DB2 systems. DataJoiner's nickname function is used only for access to non-DB2 systems.

DataJoiner DDCS provides the ability for multiple clients to connect to host data. By concentrating definitions and configurations for host access, this configuration can significantly reduce the effort required for implementing and administering access to enterprise data. See Figure 4.

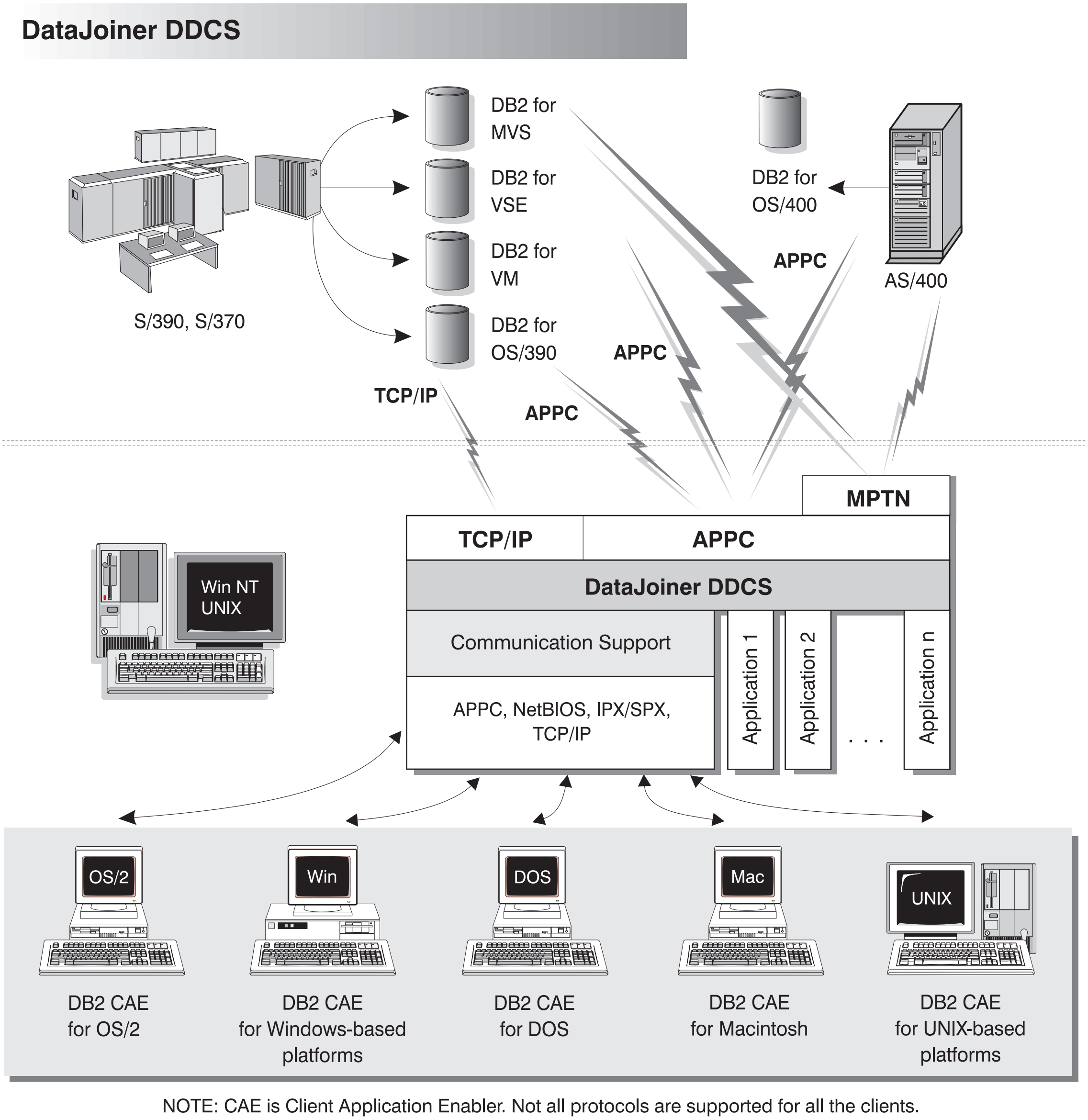


Figure 4. DataJoiner DDCS

DataJoiner DDCS allows DB2 systems on the host to assume the role of an enterprise super server. This solution allows for established data management procedures to be maintained while applications are distributed to workstation platforms to take advantage of graphical user interfaces, distributed processing power, and excellent development tools.

The DDCS functionality included with DataJoiner is equivalent to DDCS 2.3.2 with additional functionality added that allows DataJoiner to access DRDA data sources using TCP/IP.

Heterogeneous Data Replication

With DataJoiner and the Apply program, you can propagate data between DB2 systems, other relational databases, and non-IBM relational databases. DataJoiner's Apply program is equivalent to the Apply program available with DB2 Universal Database Version 5. DataJoiner's replication administration tool is an administrative interface to help you manage replication operations from a single point of control. The replication administration tool runs on a Windows 95 or Windows NT client workstation.

For the replication administration tool and the Apply program to communicate with DB2 data sources, you need to configure DataJoiner DDCS; for the replication administration tool and the Apply program to communicate with non-DB2 data sources, you need to configure server and user mappings. See "Chapter 7. Configuring Access to Data Sources" on page 71 for more information.

Chapter 2. What’s New in DataJoiner Version 2?

DataJoiner Version 2 offers new features and enhancements, including:

DB2 Version 2 functionality

DataJoiner is built on the DB2 Version 2 code base, which means that DataJoiner provides all the major functional enhancements provided by DB2, including:

- Extended SQL capabilities
- An enhanced SQL optimizer
- Improved database performance
- Systems management support
- Robust integrity and data protection
- Object relational capabilities
- National language support (NLS)
- Support for the Java Development Kit (JDK) 1.1 for the Java Database Connectivity (JDBC) API
- DDL transparency

For detailed information about many of these features, see the *DB2 Administration Guide.*

DataJoiner for Solaris systems

DataJoiner now runs on the Sun Solaris operating system. DataJoiner for Solaris systems supports the same SQL as DataJoiner for AIX systems.

DataJoiner for Windows NT systems

DataJoiner provides industrial strength heterogeneous database management on Windows NT systems. DataJoiner for Windows NT systems supports the same SQL and features as DataJoiner for AIX systems.

Support for Oracle 8, RDB, SQL Anywhere, and Teradata

With Version 2, DataJoiner continues to increase the number of natively supported data sources. The most recent additions are:

- Oracle 8
- Oracle RDB Version 6 and above (on DataJoiner for Windows NT systems only)
- Sybase SQL Anywhere Version 5.0 (on DataJoiner for Windows NT systems only)
- Teradata (on DataJoiner for AIX and DataJoiner for Windows NT systems only)

Spatial Extender

DataJoiner now supports geographic information system (GIS) data (also known as spatial or geographic data). New data types, spatially-enabled columns, and spatial join capability allow you to take advantage of geographic data in your applications. Included are powerful two-dimensional functions that allow you to create specific relationships among the geographic objects you define. Included with the spatial extender are the following components:

- A set of spatial data types
- A set of spatial operations and predicates
- A set of spatial index data types
- An administration tool suite for the spatial extender
- Sample programs

You can also take advantage of existing geographic data stores using the load and transform capability of the Spatial Extender.

Expanded DataJoiner SQL support

This version of DataJoiner contains many new and modified SQL statements. New DDL statements provide greater flexibility and safety in defining your DataJoiner environment—users can create, alter, and drop mappings for data sources, users, user-defined and built-in functions, and data types. Additionally, new SQL DML statements provide enhanced functions for local and distributed queries; an example is the CASE expression, which is useful for selecting an expression based on the evaluation of one or more conditions.

DataJoiner SQL for creating, altering, and deleting data source tables

Version 2 includes a new DataJoiner SQL statement for creating tables in different types of data sources. If the native SQL for creating tables in these data sources includes a unique option—for example, the option in DB2 for OS/390 for specifying what database you want a table to reside in—you can code this option in the new DataJoiner statement. If you create a data source table with this new statement, you can also alter and delete it with DataJoiner SQL.

Heterogeneous data replication

DataJoiner now provides replication administration as an integrated component. You can define, automate, and manage replication data from a single control point across your enterprise. The replication administration tool provides administrative support for the replication environment, with objects and actions that define and manage source and target table definitions. DataJoiner's Apply component performs the actual replication, tailoring and enhancing data as you specify, and serving as the interface point to and from your various data sources. DataJoiner also supplies an executable, IBM DB2 DataPropagator for

Microsoft Jet, that allows you to replicate server data for browsing and updating in LAN, occasionally connected, and mobile environments.

Distributed heterogeneous update support

DataJoiner for AIX and DataJoiner for Windows NT systems now allow you to update multiple heterogeneous data sources within a distributed unit of work while maintaining transaction atomicity. This task is accomplished through adherence to the two-phase commit model. Supported data sources include most versions of the DB2 Family and, with the appropriate XA libraries, various other data sources as well.

New graphical installation, configuration, and administration tools

A variety of new tools is available to help you accomplish administrative chores. Wizards walk you through data source configuration. And the Administrator's Toolkit provides a collection of tools designed to assist you with the day-to-day operation of DataJoiner. It includes the following components:

The Database Director

Allows you to perform configuration, backup and recovery, directory management, and media management tasks.

Visual Explain

A tool for graphically viewing and navigating complex SQL access plans.

The DB2 Performance Monitor

Monitors the performance of your DB2 system for tuning purposes.

The Administrator's Toolkit is available with DataJoiner for AIX and DataJoiner for Windows NT systems.

Stored procedures

DataJoiner now supports stored procedures at remote data sources as well as the local DataJoiner database. Use stored procedures to speed application performance. For example, applications that process huge amounts of data at a server but return smaller result sets should run faster as stored procedures. Another benefit is that stored procedures usually reduce network traffic between clients and databases.

DataJoiner stored procedures can augment standard data security. For example, in a 3-tier environment, data can be retrieved from a remote server and then processed at the DataJoiner server; only a subset of data needs to be available to the client.

System catalog information available in views

DataJoiner provides views from which you can access system catalog information about each DataJoiner database. Some of these views contain data—for example, data about tables, indexes, and servers—that was accessible only from tables in previous versions of DataJoiner. Other views contain data—for example, data about stored procedures, server options, and server functions—that is now available in Version 2.

Performance enhancements

In addition to general engine performance improvements, this latest version offers new query rewrite capabilities, improved pushdown performance, and remote query caching.

Part 2. Planning for and Installing DataJoiner

Chapter 3. Hardware and Software Requirements

This chapter lists the hardware and software that you need to successfully install and operate DataJoiner on your UNIX system and includes the following sections:

- "Hardware Requirements"
- "Software Requirements" on page 26

Use this information to ensure that your system meets the prerequisites before attempting to install DataJoiner.

Hardware Requirements

This section identifies the hardware requirements for DataJoiner only. It does not include hardware requirements for products that DataJoiner accesses, such as Informix, Oracle, or Sybase, nor does it contain hardware requirements for remote systems where clients reside. Requirements for DataJoiner's replication components, the replication administration tool and Apply, are provided in "Part 6. Replicating Data" on page 209. Classic Connect requirements are provided in the *DataJoiner Classic Connect Installation, Configuration, and Reference Guide* V2.1.1.

General Requirements

The minimum hardware configuration for DataJoiner is a machine that is capable of:

- Reading a CD-ROM.
- Supporting the appropriate version of the AIX or Solaris operating system, as documented in "Software Requirements" on page 26.

When you estimate your hardware requirements for DataJoiner, you need to consider both:

- The fixed-disk requirements to installing product files on the target machine
- The fixed-disk requirements and memory requirements for each instance of the product

An *instance* is a DataJoiner server, and you can have more than one instance on the same machine. Additional resource is required for each instance of DataJoiner that is created, each database, and each client. You can read more about instances in "Chapter 6. Setting up an Instance" on page 53.

Memory and Disk Requirements

This section summarizes the memory requirements and disk requirements that are related to DataJoiner.

To estimate the memory requirements and disk requirements for a particular configuration, add the recommended memory size for each installed function and application that you plan to run concurrently.

The memory requirement is less if you do not run all installed functions and applications concurrently. System memory requirements are based only on applications and operating system functions that run concurrently.

Using less than the calculated recommended memory size can increase the time that is required to switch from one function or application to another, reduce keystroke responsiveness, reduce throughput, and increase the size of the virtual memory paging space that is required.

Recommended Memory Sizes

Table 1 and Table 2 on page 23 provide the recommended memory size in megabytes (MB) for concurrent operation of various functions and the recommended disk size in MB. Use these tables to calculate your requirements.

Table 1. Memory and Disk Requirements for DataJoiner for AIX Systems

Functions	Recommended Working Set Memory (MB)	Recommended Disk Space (MB)
Requirements for product files[1]		
DataJoiner base (djx)	N/A	60.0
DRDA AR AS (drda_ar_as)	N/A	1.1
Classic SNA (djx_classic_sna)	N/A	.22
SNA clients (djx_sna_clients)	N/A	.17
DBA Tool (djx_dba)	N/A	1.1
DBA Tool messages per locale (msg.*.dba)	N/A	.9
Messages per locale (msg.*.djx)	N/A	1.1
Requirements for each instance of DataJoiner		
Instance Base[2]	buffer[3] + 2.7	.75[4]
Instance Base for Classic Connect[2]	buffer[3] + 5.0	.75[4]
For each additional concurrent database	buffer[3] + 1.7	1.1[4]
DataJoiner instance requirements for each client connection		

Table 1. Memory and Disk Requirements for DataJoiner for AIX Systems (continued)

For each concurrent local user or application	1.0	N/A
For each concurrent spatial application	n[5] * 1.2	N/A
For each concurrent remote client	.75	N/A
For each Informix data source accessed	.85	N/A
For each Classic Connect data source accessed	2.5	N/A
For each other data source accessed, other than Informix or Classic Connect	.18	N/A
TOTAL (for your requirements)		

Notes:

1. 15 MB of additional temporary space is required during installation.
2. Assumes one process using one local database on one instance.
3. Additional memory for some database configuration parameters, including the database buffer pool or the sort heap, might be required depending on your workload. The buffer pool default is 4MB. If you decide later that the buffer size needs to change, see the *DB2 Administration Guide* for instructions on how to change it. The buffer pool applies to locally stored DataJoiner data only, not data that is retrieved from data sources. Keep this fact in mind when you read instructions in the *DB2 Administration Guide*.

   Working set memory includes the parts of a program’s executable code and data areas that are being used intensively. Working set memory figures assume that default values were accepted.
4. The extra disk space for each database is for table definitions and internal structures for each database and does not include user data. The extra disk space is subject to many variables, so actual requirements might differ.

   The default configuration allocates three log files, each 4MB in size. These logs can be relocated to another file system and can be made larger or smaller as required. See the *DB2 Administration Guide* for more information.
5. *n* is the maximum number of ADT parameters that are passed to a spatial function in your query. For example, the query `SELECT cust_id FROM customers WHERE within (loc, buffer(loc, 100))` has four parameters (100, loc, buffer(loc,100), and loc). If this is the largest query in your application, the amount of working set memory required is 4 * 1.2 = 4.8MB.

   This memory requirement is specified by the environment variable **UDF_MEM_SZ**. The recommended setting for this variable is 2048 (8MB).

Table 2. Memory and Disk Requirements for DataJoiner for Solaris Systems

Functions	**Recommended Working Set Memory (MB)**	**Recommended Disk Space (MB)**
Requirements for product files[1]		

Table 2. Memory and Disk Requirements for DataJoiner for Solaris Systems (continued)

DataJoiner for Solaris systems base (DB2.SERVER)	N/A	60.0
Client Application Enabler for Solaris systems (DB2.CAE)	N/A	10.0
Software Developer’s Kit for Solaris systems (DB2.SDK)	N/A	15.0
DataJoiner DDCS functionality (DB2.DDCS)	N/A	30.0
Messages per locale	N/A	1.1
Requirements for each instance of DataJoiner		
Instance Base[2]	buffer[3] + 2.7	.75[4]
For each additional concurrent database	buffer[3] + 1.7	1.1[4]
DataJoiner instance requirements for each client connection[3]		
For each concurrent local user or application	1.0	N/A
For each concurrent remote client	.75	N/A
For each data source accessed	.18	N/A
TOTAL (for your requirements)		

Notes:

1. 15 MB of additional temporary space is required during installation.
2. Assumes one process using one local database on one instance.
3. Additional memory for some database configuration parameters, including the database buffer pool or the sort heap, might be required depending on your workload. The buffer pool default is 4 MB. If you decide later that the buffer size needs to change, see the *DB2 Administration Guide* for instructions on how to change it. The buffer pool applies to locally stored DataJoiner data only, not data that is retrieved from data sources. Keep this fact in mind when you read instructions in the *DB2 Administration Guide.*

   Working set memory includes the parts of a program’s executable code and data areas that are being used intensively. Working set memory figures assume that default values were accepted.
4. The extra disk space for each database is for table definitions and internal structures for each database and does not include user data. The extra disk space is subject to many variables, so actual requirements might differ.

   The default configuration allocates three log files, each 4 MB in size. These logs can be relocated to another file system and can be made larger or smaller as required. See the *DB2 Administration Guide* for more information.

Calculating Memory Requirements

If there is not enough real memory on a DataJoiner server, DataJoiner runs in virtual memory that is provided by the operating system. In this case, performance drops significantly because virtual memory requires disk I/O time to page in and out of real memory. Be sure to consider the memory requirements for your operating system and other applications that will run concurrently with DataJoiner when estimating the requirements for your machine configuration.

You can calculate the real memory requirements for DataJoiner with your specific configuration by adding together the memory requirements for all instances that will run concurrently. Use one of the following formulas to calculate the memory requirement (in MB) for a specific instance.

General form:

Instance base +
Additional concurrent databases +
Additional concurrent users +
Data sources

Formula in more detail:

```
bmem + buffer +
(dnum -1) * (1.7 + buffer) +
(cnum - 1) * cmem +
cnum * (smem[1] + smem[2] + ... + smem[s])
```

where:

bmem Is the memory for instance base in Table 1 on page 22. The requirement of instance base for Classic Connect is used in Table 1 on page 22.

buffer Is the buffer space referred to in footnote c of Table 1 on page 22.

dnum Is the number of concurrent databases. Note that the instance base requirement already contains the requirement for one database.

1.7 Is the memory requirement for an additional concurrent database in Table 1 on page 22.

cnum Is the number of concurrent local users, local applications, or remote clients.

cmem Is the memory requirement for one local user, local application, or remote client.

smem[1] ... smem[s]
Is the memory requirement for the data sources accessed. Summarize

each data source one by one. Note in Table 1 on page 22 that the requirement for Informix and Classic Connect data sources is different from that of other data sources.

For example, assume that there are 20 concurrent applications that run . These 20 applications use one DataJoiner instance and two databases. The instance is configured to access Classic Connect. The applications access one Oracle, one Informix, and one Classic Connect data source. The database buffer size referred to in footnote c is 4 MB.

Memory requirements for this example are:

```
bmem + buffer +
dnum * (1.7 + buffer) +
(cnum - 1) * cmem +
cnum * (smem[1] + smem[2] + ... + smem[s])
```

or

```
6.7 + 4.0 +
(2 - 1) * (1.7 + 4.0) +
(20 - 1) * 1.0 +
20 * (0.18 + 0.85 + 2.5)
```

which equals 106 MB.

Software Requirements

This section identifies the software requirements for DataJoiner, including:

- Supported operating system levels
- Required communication products for the different protocols that can be used with DataJoiner
- Client software that is required to access non-DB2 Family data sources

Software requirements for the replication administration tool and the Apply program are provided in the *IBM DB2 Replication Guide and Reference (V5)*.

Operating System and Communications Requirements

Table 3 lists the operating system and communications software required for DataJoiner.

Table 3. Operating System and Communication Protocol Requirements

Operating System Requirements	Communications Requirements
DataJoiner for AIX Systems	

Table 3. Operating System and Communication Protocol Requirements (continued)

Operating System Requirements	**Communications Requirements**
DataJoiner for AIX systems requires one of the following operating systems: • IBM AIX Version 3.2.5 with the following PTFs or any PTFs that supersede any of the following PTFs: – U403173 – U412397 – U412815 • IBM AIX Version 4.1 or later **Notes:** 1. To take advantage of the DB2 Spatial Extender option (djx_02_01_01.se), AIX Version 4.1.4 or later is required. 2. If you installed the IBM RISC System/6000 feature for common mode development (FCM) static libraries, and you are experiencing problems with DataJoiner, contact IBM Service. 3. The dce.pthreads.rte package is required to use the Oracle Net8 data access module.	APPC, IPX/SPX, and TCP/IP for server function; APPC and TCP/IP for client function For IBM AIX Version 3.2.5 systems, you need: • For APPC connectivity, one of the following products: – IBM SNA Server/6000 Version 2.1 or later – IBM eNetwork Communications Server for AIX V5.0.2.4 – Bull DPX/20 SNA/20 • For IPX/SPX connectivity, IBM AIX NetBIOS and IPX Support/6000 Version 1.1 • For TCP/IP connectivity, the TCP/IP component of the operating system must be installed For AIX Version 4.1 systems, you need: • For APPC connectivity, one of the following products: – IBM SNA Server/6000 Version 3.1 or later – IBM Communications Server Version 4 or later – IBM eNetwork Communications Server for AIX V5.0.2.4 – Bull DPX/20 SNA/20 • For IPX/SPX connectivity, one of the following products: – IBM AIX NetBIOS and IPX Support/6000 Version 2.1 – IBM NetWare for AIX Version 3.11B. (This is the complete product which includes a NetWare file server) • For TCP/IP connectivity, the TCP/IP component of AIX must be installed.

Table 3. Operating System and Communication Protocol Requirements (continued)

Operating System Requirements	Communications Requirements
Notes: 1. The database server can accept requests from multiple protocols concurrently. If you want to use several protocols, ensure that you have all the required products installed. 2. If you want to install the DRDA Application Server function of DataJoiner, you must install the Syncpoint Manager (SPM) and its prerequisites to use two-phase commit. 3. If you plan to use the ADSTAR Distributed Storage Manager (ADSM) facilities for backing up and restoring your databases, you need the ADSM Client Version 3.0 or later. 4. If you plan to use the Simple Network Management Protocol (SNMP) subagent, you need DPI 2.0 provided by IBM Netfinity Agent.	
DataJoiner for Solaris Systems	
DataJoiner for Solaris systems requires Solaris version 2.5.1 or later. The following patches are required for Solaris version 2.5.1: • 101242 Rev. 11 or later • 103566 Rev. 08 or later • 103600 Rev. 13 or later • 103640 Rev. 20 or later The following patches are required for Solaris version 2.6: • 105568 Rev. 12 or later • 105210 Rev. 13 or later • 105181 Rev. 06 or later	APPC, IPX/SPX, or TCP/IP; for DRDA AS function, TCP/IP • TCP/IP is provided with the Solaris base operating system. • IPX/SPX connectivity is provided with SolarNet PC Protocol Services 1.1 with IPX/SPX for the Solaris 2.x operating environment. • For APPC connectivity, you need Sun Link SNA 9.0 or later and the following communications products: – SunLink P2P LU6.2 9.0 or later – SunLink PU2.1 9.0 or later – SunLink P2P CPI-C 9.0 or later **Notes:** 1. DB2 Connect for Solaris requires Version 2.6 or later. 2. If you plan to use the ADSTAR Distributed Storage Manager (ADSM) facilities for backing up and restoring of your databases, you need the ADSM Client Version 3.0 or later.

Requirements for DataJoiner Data Sources

Table 4 lists the data sources that DataJoiner supports, as well as any required client software needed to access these data sources.

Table 4. DataJoiner Data Sources

Data Source	Client Software Needed on the DataJoiner System
IBM DataJoiner	included in DataJoiner

Table 4. DataJoiner Data Sources (continued)

Data Source	Client Software Needed on the DataJoiner System
IBM DB2 Universal Database V5.0	included in DataJoiner
IBM DB2 for CS V2	included in DataJoiner
• IBM DB2 for MVS V2R3 with APAR PN43153, UN75958, UN54600, UN56735 • IBM DB2 for MVS V3R1 with APAR PN70612, UN42626, UN54601, UN56736, UN73393 • IBM DB2 for MVS V4R1 with APAR PN70612 • IBM DB2 for OS/390	included in DataJoiner
• IBM DB2 for OS/400 V2R2 with PTF SF13747, SF13748 • IBM DB2 for OS/400 V3R1 or later	included in DataJoiner
IBM SQL/DS V3R3 with APAR PN43497 or later	included in DataJoiner
• IBM SQL/400 V2R2 with PTFs SF13747 and SF13748; or in a DBCS environment, IBM SQL/400 V2R2 with SF31979 • IBM SQL/400 V3R1 with PTF SF23247 or later; or in a DBCS environment, IBM SQL/400 V3R1 with SF32970	included in DataJoiner
IBM DB2/PE	included in DataJoiner
IBM DB2 for Solaris V1 and V1.2	included in DataJoiner
IBM DB2 for HP-UX V1.2	included in DataJoiner
IMS V2*	DataJoiner Classic Connect
Informix V5 or V6*	Informix Net V5.03 or later and ESQL/C V5.03 or later libraries.
Informix OnLine V7.1 or later (non-GLS, -XPS, and -SE versions only)	Informix ESQL/RT or equivalent libraries
Microsoft SQL Server V6.5 or lower	Sybase Open Client dblib
Microsoft SQL Server V6.5*	MERANT DataDirect Connect ODBC
Microsoft SQL Server V7.0*	MERANT DataDirect SequeLink ODBC Edition
ODBC, X/Open CLI-compliant data sources through the generic access API	ODBC driver
Oracle V7.0.13 or later	Oracle SQL*Net V1 or V2 or Oracle Net8
Sybase V4R6 or later	Sybase Open Client
Teradata V2*	Teradata CLI for AIX
VSAM*	DataJoiner Classic Connect

Table 4. DataJoiner Data Sources (continued)

Data Source	Client Software Needed on the DataJoiner System
Other nonrelational data sources*	Cross Access Corporation's CrossAccess Data Delivery System

* Not supported by DataJoiner for Solaris Systems

Graphical User Interface Requirements

To use the Database Director tool, you must have the following software installed on your AIX system:

- IBM AIXWindows Version 1.2.3 or later
- Motif Version 1.2 (x11r5)

DataJoiner for Solaris systems does not support the Database Director tool.

Chapter 4. Planning for Installation

This chapter describes various DataJoiner options and components, as well as products that can be used with DataJoiner. The following sections contain Information about these options, components, and products:

- "Remote Clients"
- "Supported DRDA Databases" on page 32
- "Replication Issues" on page 33
- "Supported Transaction Managers" on page 34
- "High Availability Using HACMP" on page 35
- "DB2 SNMP Subagent" on page 35
- "Two-Phase Commit Requirements" on page 35
- "Supported Compilers for DataJoiner" on page 35
- "ADSTAR Distributed Storage Manager" on page 36
- "Prerequisites for DataJoiner for AIX Options" on page 37

Remote Clients

Remote clients must access DataJoiner through one of the DB2 Client Application Enablers (CAEs) and Software Developer's Kits (SDKs). DataJoiner supports the V2.1.1 or later CAE. However, it is recommended that you use the most current supported CAE. Because remote clients can run in a variety of operating environments (including OS/2, several UNIX-based environments, DOS, Windows 95, and Windows NT), a platform-specific version of the DB2 CAE and SDK is available for each of these environments.

A DB2 CAE component is built into each of the DB2 for CS products. Also, when you purchase DataJoiner or the server version of DB2, you receive a Client Pack that includes each of the available DB2 CAEs. You can copy the CAEs to the appropriate workstations for each remote client that will connect to the server.

A workstation that has a DB2 CAE and SDK installed can access any DataJoiner or DB2 server through any of the supported communication protocols that are documented in "Part 4. Configuring DataJoiner as a DRDA AS" on page 175. This support protects your investment in client workstations and allows you to select a server machine that is most appropriate for your database environment.

To determine which protocols can be used with your clients, see *Installing and Using Clients* for your particular platform.

Supported DRDA Databases

DataJoiner and DDCS support the following IBM host relational database products:

- **DB2 for MVS/ESA Version 2 Release 3**

  PTF UN75958 Required to use DB2 for MVS/ESA as an application requester.

  PTF UN54600 Required for connection with UNIX operating systems.

  PTF UN56735 Required for connection with UNIX operating systems.

- **DB2 for MVS/ESA Version 3 Release 1**

  PTF UN73393 Required to use two-phase commit with CICS/6000 and to connect to DB2 for MVS/ESA using DCE Services from UNIX clients.

  PTF UN75959 Required to use DB2 for MVS/ESA as an application requester.

  PTF UN54601 Required for connection with UNIX operating systems.

  PTF UN56736 Required for connection with UNIX operating systems; also required for two-phase commit with CICS clients.

- **DB2 for MVS/ESA Version 4 Release 1**

  APAR PN69689
  Required to use DRDA stored procedures.

  APAR PN69748
  Required to use DRDA stored procedures.

- **DB2 for OS/390**
- **SQL/DS Version 3 Release 3 (DB2 for VSE and VM)**[1]

  PTF PN43497 Required for connection with UNIX operating systems.

  PTF UN47865 Required to use DB2 for VSE and VM as an application requester.

  APAR PN69073
  Required to use DB2 for VSE and VM as an application requester. Additional scripts must be run on the OS/2 or AIX application server. See the *DB2 Installation and Operation Guide* for more information.

- **SQL/DS Version 3 Release 4 (DB2 for VSE and VM)**[1]

- **DB2 for OS/400 Version 2 Release 2**[1]

  PTF SF13747 Required for connection with UNIX operating systems.

  PTF SF13748 Required for connection with UNIX operating systems.

  DB2 for OS/400 Version 3 Release 1[1]

Note: Consult with your host database coordinator to ensure that the appropriate prerequisites and security considerations have been considered and installed to support your UNIX environment and host database environments.

Replication Issues

If you will be using DataJoiner to replicate data, here are some things you need to know:

- The replication administration tool runs on a Windows 95 or Windows NT workstation and accesses DataJoiner through the DB2 CAE for Windows 95 or NT. This means that you will need to configure the DataJoiner instance for remote client access.
- Access to different types of data sources is achieved as follows:
  - Both the replication administration tool and the Apply program must use DataJoiner's data access modules to access non-IBM data sources. See the appropriate section in “Chapter 7. Configuring Access to Data Sources” on page 71 for details.
  - To access DB2 for CS or DB2 Universal Database data sources, the Apply program can use DataJoiner's DB2RA protocol, while the replication administration tool can access these data sources directly through the CAE. See “Chapter 9. Accessing DB2RA Data Sources” on page 107 for more information.
  - To access IBM IMS/ESA nonrelational databases, you need DPRONR.
  - To access DB2 for OS/390 data sources, the Apply program should go through DataJoiner DDCS, while the replication administration tool can use either DataJoiner DDCS or DDCS on another workstation.

    Consider network performance when deciding how to access data sources. For example, using DDCS for Windows NT on the same workstation as the replication administration tool typically provides faster access to DB2 for OS/390 than going to another workstation to access DataJoiner DDCS and subsequently DB2 for OS/390. This is dependent on your site's network.

[1] The replication administration tool currently does not support this data source.

- When replicating from one or more non-IBM relational databases, one DataJoiner database is required for each non-IBM database. Plan accordingly for file space and memory impact. When replicating to one or more non-IBM relational databases, all targets can be accessed through a single DataJoiner database, thus saving resources.
- When replicating from non-IBM databases to DB2 for OS/390 databases, Apply/MVS typically is used to replicate data. If Apply/MVS is to be used rather than the Apply component within DataJoiner, configure DataJoiner as a DRDA application server as described in “Chapter 18. Configuring the DataJoiner DRDA Application Server and Clients” on page 177.
- When replicating from Oracle databases, the DataJoiner database you create to access Oracle must be created with the COLLATE USING IDENTITY clause.
- For existing replication installations that used DataPropagator Relational (DPROPR) Version 1, there is no migration program to IBM Replication (the replication tool used in DB2 UDB Version 5). You must use DataJoiner’s replication administration tool to set up your replication configuration.
- The DPROPR V1 Capture program cannot be used with the DataJoiner Apply program; you must use the DataPropagator Relational V5 or higher Capture program.

Replication planning and configuration information are discussed more fully in “Part 6. Replicating Data” on page 209.

Supported Transaction Managers

DataJoiner functions as a transaction manager, or you can install any of the following supported transaction managers:

- IBM CICS/6000 Version 1.2 or 2.1
- Encina TP Monitor Version 1.3
- Tuxedo System/T Version 4.2.2

To support DRDA two-phase commit with Encina TP Monitor or CICS/6000 against DB2 for MVS/ESA Version 3.1 or DB2 for OS/400 Version 3.1, you must also install Encina PPC Gateway Version 1.3 with APARs IX49555 and IX50207.

Two-phase commit restrictions:

1. Tuxedo System/T cannot be used for two-phase commit with DRDA databases.
2. DataJoiner for Solaris systems does not support two-phase commit transactions.

High Availability Using HACMP

The IBM AIX High Availability Cluster Multi-Processing/6000 (HACMP/6000) system enables clustered RISC System servers to recover from server, disk, network, or network interface failures. To use HACMP with DataJoiner, install IBM HACMP/6000 Version 2.1 or later on your system.

DB2 SNMP Subagent

Simple Network Management Protocol (SNMP) products include IBM NetView for AIX and IBM NetView for OS/2. DataJoiner provides support for SNMP management products through the DataJoiner SNMP Subagent, included as a component of the DataJoiner server. SNMP management products, such as NetView, allow centralized management of a system’s hardware and software components. The DataJoiner SNMP Subagent allows DataJoiner servers to be managed just like any other SNMP-managed resource.

See the readme file for the software requirements for the SNMP Subagent.

Two-Phase Commit Requirements

DataJoiner for AIX systems supports two-phase commit for Oracle, Informix, Sybase (ctlib only), and various versions of DB2. The XA libraries must be installed and activated for any data source participating in a two-phase commit transaction. Additionally, the XA libraries must be link-edited with the DataJoiner server. These requirements do not apply to DB2-Family data sources.

See “Configuring for Two-Phase Commit” on page 73 for more information.

Supported Compilers for DataJoiner

DataJoiner provides support for C, C++, COBOL, and FORTRAN compiled languages. Writing an application in a compiled language gives the programmer a high level of flexibility. For a compiled language, an appropriate precompiler must be available to process the SQL statements. DataJoiner for UNIX systems provides C, C++, and FORTRAN precompilers. The Micro Focus COBOL compiler provides a COBOL precompiler for use with DataJoiner.

The application development environment for remote clients is provided by the DB2 SDK V1 or higher. Please check your SDK documentation for an up-to-date list of compilers that are supported by the SDKs.

The specific compilers that are supported by DataJoiner for AIX systems include:

- IBM XL C Version 1.2.1 or 1.3
- IBM C for AIX Version 3.1
- IBM C Set++ Version 2.1
- IBM C Set++ for AIX Version 3.1
- IBM AIX XL FORTRAN Version 2.3, 3.1, or 3.2
- IBM COBOL Set for AIX Version 1.1
- Micro Focus COBOL Version 3.1.49 or later
- IBM REXX/6000 (available as PRPQ 5764-057)

The specific compilers that are supported by DataJoiner for Solaris systems include:

- IBM C Set++ for the Solaris Operating Environment Version 1
- SPARCompiler C 3.0.1
- SPARCompiler C++ 4.0.1
- SPARCompiler Fortran Version 3.0.1
- Micro Focus COBOL for UNIX Version 3.2

The DB2 SDK products provide an API that application development tool providers can use to offer a DB2 precompiler within their products. For example, IBM PL/I for OS/2 Professional Edition Version 1.2 and IBM PL/I for AIX Version 1.1 include precompilers for DB2 that exploit this interface.

ADSTAR Distributed Storage Manager

If you plan to use the ADSTAR Distributed Storage Manager facilities for backup and restore of your databases, you must have the ADSTAR Distributed Storage Manager product installed on your server. The ADSTAR Distributed Storage Manager API, which allows the database manager to interface with the host product, is included with DataJoiner.

Information and support for ADSM and its APIs are provided by ADSTAR. To contact ADSTAR from the U.S. or Canada, call 1-800-4-ADSTAR (1-800-423-7827). Outside of the U.S. and Canada, use the World Trade access number, which is 1-408-284-0385.

Prerequisites for DataJoiner for AIX Options

Many of the options that are contained on the DataJoiner for AIX Version 2 CD-ROM require you to select prerequisite options or install prerequisite software. This section lists options on the DataJoiner for AIX Version 2 CD-ROM and their prerequisite requirements. The options are similar to DB2 Version 2 options.

If you are using smitty to install DataJoiner, you will need to scroll right to see an option's abbreviated name (djx_02_01_01.xxx).

DB2 Client Application Enabler (djx_02_01_01.client)

- Prerequisite
  - – IBM AIX (R) Version 3.2.5

  or
  - – IBM AIX (R) Version 4.1

    On AIX 4.1 systems, the following apply:
    - xlC.rte 3.1 or higher must be installed.
    - On AIX 4.1.4 or higher systems, dce.pthreads.rte 2.1.0 or higher must be installed.
    - DCE directories are supported only on AIX 4.1.4 or higher.
- Selectable options
  - – None

The message catalog for US English is included.

Command Line Processor Option (djx_02_01_01.clp)

- Prerequisite
  - – djx_02_01_01.client
- Selectable options
  - – None

SDK Options (djx_02_01_01.sdk...)

- Prerequisite
  - – djx_02_01_01.clp
- Selectable options
  - – djx_02_01_01.sdk.c (C include files and sample programs)
  - – djx_02_01_01.sdk.fortran (Fortran include files and sample programs)
  - – djx_02_01_01.sdk.cobol (COBOL include files and sample programs)
  - – djx_02_01_01.sdk.cli (Call Level Interface samples)
  - – djx_02_01_01.sdk.misc (SDK utilities and samples)

DataJoiner Executables Options (djx_02_01_01.db2)

- Prerequisite
  - djx_02_01_01.clp
- Selectable options
  - djx_02_01_01.db2.rte (DataJoiner executables)
  - djx_02_01_01.db2.misc (DataJoiner utilities and samples)

Code Page Conversion Option (djx_02_01_01.conv)

- Prerequisite
  - djx_02_01_01.client
- Selectable options
  - None

Communication Support Options (djx_02_01_01.cs)

- Prerequisite
  - djx_02_01_01.db2.rte
- Selectable options
  - djx_02_01_01.cs.rte (Base with TCP/IP support)
    - Prerequisite
      - djx_02_01_01.db2.rte
  - djx_02_01_01.cs.sna (SNA support)
    - Prerequisites
      - djx_02_01_01.cs.rte
      - SNA Server/6000 Version 2.1

      or
      - Desktop SNA Version 1 (AIX 4.1 only)
  - djx_02_01_01.cs.drda (DRDA Application Server)
    - Prerequisites
      - djx_02_01_01.cs.sna
      - SNA Server/6000 Version 2.1.1 with PTFs U435033 and U435034
  - djx_02_01_01.cs.ipx (IPX/SPX support)
    - Prerequisites
      - djx_02_01_01.cs.rte
      - IBM Netware for AIX Version 3.11B

      or
      - AIX NetBIOS and IPX Support/6000 Version 1.1

DDCS Options (djx_02_01_01.ddcs)

- Prerequisites contain one of the following products:
  - SNA Server/6000 Version 2.1
  - Desktop SNA Version 1 (AIX 4.1 only)

  - Encina Server for AIX Version 1.3 (encServ.obj 1.3 on AIX 3.2 or AIX 4.1 equivalent)
- Selectable options
  - None

DB2 Database Director Option (djx_02_01_01.dd)

- Prerequisites
  - djx_02_01_01.clp
  - IBM Information Presentation Facility
  - X11fte.motif1.2.obj (1.2.3 or higher on AIX 3.2 only)
- Selectable options
  - None

DB2 Visual Explain Option (djx_02_01_01.ve)

- Prerequisite
  - djx_02_01_01.dd
- Selectable options
  - None

DB2 Spatial Extender Option (djx_02_01_01.db2se)

- Prerequisite
  - IBM AIX (R) Version 4.1
- Selectable options
  - None

Chapter 5. Installing DataJoiner

The information in this chapter describes how to install DataJoiner on your UNIX system and includes the following sections:

- "Before You Install DataJoiner"
- "A Common Installation Problem for AIX 3.2" on page 42
- "Installation" on page 43

Before You Install DataJoiner

This section discusses issues that can affect the successful installation of DataJoiner. Please read this entire section **before attempting to install DataJoiner**, and give careful consideration to all topics that apply to you.

Install Prerequisite Software

Be sure that you understand DataJoiner's hardware and software requirements and that your system meets the minimum required levels. See "Hardware Requirements" on page 21 and "Software Requirements" on page 26.

After familiarizing yourself with DataJoiner's software requirements, make sure that all required prerequisite software is installed. For example, you might need client software for access to data sources like Sybase and Oracle.

Understand DataJoiner Security

Read the entire security section of the *DataJoiner Administration Supplement*. Understanding security is especially important here because DataJoiner accesses so many data sources.

Decide on Appropriate Authorization

Develop a plan for setting up proper access to your databases. A key part of this plan is to authorize users according to their responsibilities. Consider what authorization levels you will grant to certain users. For more information, see the *DataJoiner Administration Guide*.

Decide Which Client Protocols to Use

Determine which communication protocols your clients will use to communicate with DataJoiner. DataJoiner supports advanced program-to-program communication (APPC), Transmission Control

Protocol/Internet Protocol (TCP/IP), and Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX).

For DataJoiner for AIX systems, support for these protocols is enabled either by installing the entire DataJoiner product, which enables all three protocols, or by selecting a separate installable option for each protocol that you want.

For DataJoiner for Solaris systems, support for all three protocols is enabled by default.

Adding API Entry Points for Micro Focus COBOL

If you plan to develop DataJoiner applications with the Micro Focus** COBOL precompiler front end, runtime interpreter, or debugger (the ″Animator″), you must add the DataJoiner Generic API entry points to the Micro Focus runtime library.

To add the DataJoiner Generic API entry points, you must execute the **mkrts** command that is provided by Micro Focus.

- Login as *root*.
- Execute **mkrts** with the arguments that are supplied in `/usr/lpp/djx_02_01_01/lib/db2mkrts.args`.

For example:

```
mkrts `cat /usr/lpp/djx_02_01_01/lib/db2mkrts.args`
```

Note: The ″`″ character in the preceding command is the right prime character, which sometimes is referred to as the back-quote character.

Refer to the Micro Focus documentation for more information about the **mkrts** command and its other arguments that may be appropriate for your specific environment.

A Common Installation Problem for AIX 3.2

The DataJoiner options can be installed on systems that are running either AIX 3.2 or AIX 4.1. The operating system check is performed when the `djx_02_01_01.db2.rte` option is installed. Because of the way prerequisite failures are reported on AIX 3.2 systems, the `djx_02_01_01.db2.rte` option might fail to install and incorrectly report that an AIX 4.1 prerequisite is missing. When this problem occurs, it is usually caused by one of the following conditions:

- The operating system (AIX 3.2) is in an inconsistent state.
- A required AIX 3.2 program temporary fix (PTF) is missing.

To determine if there is an inconsistency in the AIX 3.2 operating system, enter the following command:

```
lppchk -v bos.obj
```

If this command reports any error conditions, correct the situation by installing the reported PTFs before you try to install the `djx_02_01_01.db2.rte` option again.

To check for the presence of a PTF that is required for a product, enter the following commands:

```
export ODMDIR=/usr/lib/objrepos
odmget -q lpp_name=Name product | grep PTFnumber
```

Name is the name of the product to which the PTF should be applied (for example, bos.obj). `PTFnumber` is the number of a required PTF (PTFs have the form U4*XXXX*, where *X* indicates a digit from 0 to 9). Make sure that there are no spaces on either side of the equal sign (=).

If the required PTF is not found, you must install it before proceeding with the DataJoiner install.

Installation

This section contains instructions for installing DataJoiner and its components. Specific DataJoiner options are listed in “Prerequisites for DataJoiner for AIX Options” on page 37. For information about installing the replication administration tool, see “Part 6. Replicating Data” on page 209.

The steps to install DataJoiner are:

- “Step 1: Install DataJoiner Product Files”
- “Step 2: Update Solaris Kernel Configuration Parameters” on page 45
- “Step 3: Link DataJoiner to Data Source Client Software” on page 46
- “Step 4: Adding DataJoiner Options after Initial Installation” on page 52

Note: To view the documentation that is supplied on the DataJoiner for AIX CD-ROM before installing DataJoiner. See “Appendix E. Viewing Documentation” on page 359.

Step 1: Install DataJoiner Product Files

This step provides instructions for installing DataJoiner product files on both AIX and Solaris systems. Go to the section that applies to your operating environment:

- “Installing DataJoiner Product Files on AIX Systems”
- “Installing DataJoiner Product Files on Solaris Systems” on page 45

Installing DataJoiner Product Files on AIX Systems

Use either installp or smitty to install DataJoiner for AIX systems. Instructions for using these utilities are provided in the following sections.

> The AIX operating system provides detailed help for installp and smitty. Help for installp is available by entering `man installp`.

Installp: To install DataJoiner from a CD-ROM drive using installp, enter the following command:

```
installp -ag -d /dev/cd0 djx_02_01_01.db2.rte
```

This installs only the base DataJoiner product. To install other product options, you must enter their names separately. See “Smitty” for instructions on installing individual product components.

Smitty: The following instructions apply to AIX version 4.1. Previous versions of AIX might have renamed or reorganized the smitty panels. The overall process, however, should remain relatively similar.

To install with smitty:

1. Enter **smitty** from the command line.
2. Select **Software Installation and Maintenance**.
3. Select **Install and Update Software**.
4. Select **Install/Update Selectable Software (Custom Install)**.
5. Select **Install Software Products at Latest Level**.
6. Select **Install New Software Products at Latest Level**.
7. Press F4 to display a list of devices. Position the cursor on the CD-ROM drive where the DataJoiner CD-ROM is located and press Enter.
8. With the **SOFTWARE to Install** option highlighted, press F4 to display a list of software to install. Use the F7 key to select DataJoiner options:
   - To install the entire DataJoiner product and all its options, select `djx_02_01_01` (you will need to scroll right to see the abbreviated name). This automatically selects all indented options that are listed under the `djx_02_01_01` option. AIX 3.2.5 users need to select `djx_02_01_01.db2.rte`.
   - To install individual DataJoiner components, position the cursor to the left of each option, and press F7. See “Prerequisites for DataJoiner for AIX Options” on page 37 for information about individual options and their prerequisite requirements.

9. Press Enter and review your selections if necessary by scrolling right.
10. Press Enter again to begin the installation.

Installing DataJoiner Product Files on Solaris Systems

To install DataJoiner on a Solaris system, use the **db2_install** command. This command is available on the distribution media for DataJoiner.

Notes:

1. The **db2_install** program uses the Solaris **pkgadd** command; however, do not use the **pkgadd** command directly to install DataJoiner.
2. The **db2_install** command prompts you for the products to be installed and for the base directory where the product files are to be installed. It creates the subdirectory `IBMdjx/V2.1.1` under the base directory *BASEDIR*. The default value for *BASEDIR* is `/opt`.

To install DataJoiner for Solaris systems from a CD-ROM:

1. Login as root and place the CD-ROM into the CD-ROM drive.
2. If the Volume Manager is not installed on your system, enter the following commands to mount the CD-ROM:

   ```
   mkdir -p /cdrom/IBMdjx
   mount -F ufs -r /dev/dsk/c0t6d0s2 /cdrom/IBMdjx
   ```

   If the Volume Manager (vold) is installed on your system, the CD-ROM is automatically mounted as:

   ```
   /cdrom/IBMdjx
   ```
3. Run the **db2_install** command:

   ```
   /cdrom/IBMdjx/db2_install
   ```

   When prompted, enter the products to be installed and the name of the base directory. Minimally, you must select the DATAJOINER product.

Step 2: Update Solaris Kernel Configuration Parameters

This step applies to DataJoiner for Solaris systems only. Before you can run DataJoiner for Solaris, you need to update some kernel configuration parameters; Figure 5 on page 46 shows the recommended minimum values:

```
* msg queues: msgmax & msgmnb must be 64K, the others can be tailored
set msgsys:msginfo_msgmax = 65535
set msgsys:msginfo_msgmnb = 65535
set msgsys:msginfo_msgmap = 602
set msgsys:msginfo_msgmni = 400
set msgsys:msginfo_msgssz = 16
set msgsys:msginfo_msgtql = 600
set msgsys:msginfo_msgseg = 32768
*
* shrd mem: shmmax limits your database configuration (esp buffer pool)
set shmsys:shminfo_shmmax = 256000000
set shmsys:shminfo_shmseg = 10
set shmsys:shminfo_shmmni = 256
*
set semsys:seminfo_semmni = 512
set semsys:seminfo_semmap = 514
set semsys:seminfo_semmns = 1024
set semsys:seminfo_semmnu = 1400
```

Figure 5. Recommended values for the Solaris kernel configuration parameters

These lines can be appended at the end of the `/etc/system` file by using an editor. After editing the files, you must reboot the system for the changes to take effect.

Step 3: Link DataJoiner to Data Source Client Software

To enable access to data source types, the DataJoiner server must be link-edited to the client libraries. The link-edit process creates a data access module for each data source with which the DataJoiner server will communicate.

There are two ways to create a data access module:

- By running the djxlink.sh script.
- By editing and running djxlink.makefile.

The djxlink.sh script creates a data access module for nearly all commonly-used data sources. In most environments, running djxlink.sh is all that needs to be done. However, for library levels that are different from those that are supported by djxlink.sh, you need to edit and run djxlink.makefile to link all needed data source types. Both methods are described in the following sections.

Running djxlink.sh

To run djxlink.sh:

1. Make sure that required environment variables are set.

To access certain data sources, you need to set an environment variable to indicate the path where the data soure's client library resides. This step must be performed as root. The following list indicates the data sources that require this step and the corresponding environment variables:

Microsoft SQL Server

- DJX_ODBC_LIBRARY_PATH (through MERANT DataDirect Connect ODBC Driver)
- SQLNK_HOME (through MERANT DataDirect SequeLink ODBC Edition Driver)
- SYBASE_HOME (through Sybase Open Client dblib)

Informix
INFORMIX_HOME

Oracle ORACLE_HOME

Sybase
SYBASE_HOME

Teradata
TERADATA_HOME

2. Build your data access modules by using the djxlink.sh script that is located in `/usr/lpp/djx_02_01_01/lib` on AIX systems and in | `/opt/IBMdjx/V2.1.1/lib` on Solaris systems.

| Figure 6 on page 48 shows the typical results of running djxlink.sh:

```
  Starting djxlink.sh on urchin at Mon Aug 24 09:58:15 PDT 1999

Attempting to build mssqlodbc Data Access Module
  Successfully built mssqlodbc Data Access Module
Attempting to build djxsqlnk Data Access Module
  Successfully built djxsqlnk Data Access Module
Attempting to build dblib Data Access Module
  Successfully built dblib Data Access Module
Attempting to build ctlib Data Access Module
  Trying variation 1
  Trying variation 2
  Successfully built ctlib Data Access Module
Attempting to build informix Data Access Module
  Warning:           Cannot prepare informix Data Access Module for use.
  Problem:           INFORMIX_HOME environment variable not correctly set
  Corrective action: Please set INFORMIX_HOME environment variable to the path where
                     Informix*Net is installed and re-run djxlink.sh
Attempting to build informix7 Data Access Module
  Trying variation 1
  Successfully built informix7 Data Access Module
Attempting to build sqlnet Data Access Module
  Trying variation 1
  Trying variation 2
  Trying variation 3
  Trying variation 4
  Trying variation 5
  Trying variation 6
  Trying variation 7
  Successfully built sqlnet Data Access Module
Attempting to build net8 Data Access Module
  Was not able to find link-edit parameters for net8 that would work
Attempting to build teradata Data Access Module
  Successfully built teradata Data Access Module
Attempting to build xaccess Data Access Module
  Successfully built xaccess Data Access Module


Results of djxlink
------------------
mssqlodbc       Success
djxsqlnk        Success
dblib           Success
ctlib           Success
informix        Failure
informix7       Success
sqlnet          Success
net8            Failure
teradata        Success
xaccess         Success

Finished djxlink.sh on urchin at Mon Aug 24 09:58:59 PDT 1999
```

Figure 6. Example of output that results from running the djxlink.sh script

The output from djxlink.sh indicates which data access modules were successfully built and which ones could not be built. For those data access modules that could not be built, instructions for corrective actions are supplied.

Editing djxlink.makefile

If your library levels are different from those that are supported by djxlink.sh, you need to edit and run djxlink.makefile, supplied in `/opt/IBMdjx/V2.1.1/lib`.

Before you edit djxlink.makefile, consider the following information:

For all data sources:

- Ensure that you are authorized to edit djxlink.makefile. If not, use the **chmod** or **chown** command to change your permissions.
- Data source types that have more than one entry in djxlink.makefile are prefixed with the comment character (#). To enable access to these data sources, you must remove the comment character. When removing comment characters, be careful to preserve the tab character that immediately follows. Also, make sure that there are no spaces after the backslashes (\) that end each line.
- On AIX systems:
  - If you are using the `-b` flag, ensure that the character that immediately follows is an uppercase 'I' and *not* a lowercase '`l`' (the '`l`' would cause an overwrite of the file, which is not what you want).
  - For data sources that will be participating in two-phase commit transactions, you must modify djxlink.makefile to include XA library information. See "Configuring for Two-Phase Commit" on page 73 for examples.

For Oracle data sources:

- The shell/environment variables for Oracle data sources need to be set only if you plan to use Oracle as a data source. The Oracle libraries you link-edit and the directories where the libraries reside might vary depending on the version of Oracle's client software (SQL*Net or Net8) that you are using. The link information is available in the Oracle installation log if the installer requested that SQL*Net or Net8 be relinked during the installtion process.
- If you use SQL*Net, you must use the sqlnet data access module. Similarly, if you use Net8, you must use the net8 data access module. Either option allows you to access both Oracle Version 7 and Oracle Version 8 data sources.

- If you are not sure which Oracle libraries to list in djxlink.makefile for the version and release of Oracle client software you have, see the DataJoiner Web site. This Web site has a list of frequently asked questions (FAQ) that show the libraries needed for some of the more common releases of both SQL*Net and Net8. The DataJoiner Web site is located at:
  http://www.ibm.com/software/data/datajoiner

For Microsoft SQL Server data sources:
Support for Microsoft SQL Server is provided through Sybase Open Client (through the dblib API) or through the MERANT DataDirect drivers (AIX only).

For Sybase data sources:
The shell/environment variables for Sybase data sources need to be set if you plan to use Sybase as a data source and have the appropriate Sybase licenses. The Sybase libraries that you link and the directories where the libraries reside might vary depending on which version of Sybase you are using. See the Sybase product documentation for Sybase link-edit requirements.

When running djxlink.makefile, modify either the dblib or ctlib section. Sybase Open Client dblib or ctlib can be used to access Sybase V4R6 or later. Sybase Open Client dblib is valid for Microsoft SQL Server 6.0 and 6.5. Sybase Open Client cannot be used to access Microsoft SQL Server 7.0.

For Teradata data sources:

Set the DJX_TERADATA_LIBRARY_PATH variable to the directory that contains the Teradata client library (libcliv2.a).

Running djxlink.makefile

1. If DataJoiner is already installed on your system and you want to add or delete a data source, stop all instances of DataJoiner, by entering:

   ```
   db2istop
   ```

2. As root, make sure that you are in the library directory by entering the following command:

   On AIX systems

   ```
   cd /usr/lpp/djx_02_01_01/lib
   ```

   On Solaris systems

   ```
   cd /opt/IBMdjx/V2.1.1/lib
   ```

3. Ensure that you have enough free space in the /usr or /opt file system to run djxlink.makefile, by entering:

   ```
   df -I .
   ```

You need less than 5 MB of free space for each data access module that you are linking. If you do not have enough free space, add more.

4. Run the djxlink.makefile:

   ```
   make -f djxlink.makefile data access module
   ```

   where *data access module* is one of the following data access modules:

   - ctlib
   - dblib
   - djxsqlnk
   - Eda
   - Generic
   - informix
   - informix7
   - net8
   - mssqlodbc
   - sqlnet
   - teradata
   - Xaccess

   Note: Because you run djxlink.makefile and djxlink.sh from the `/usr/lpp/djx_02_01_01/lib` or `/opt/IBMdjx/V2.1.1/lib` directory, you are enabling the data sources for all DataJoiner servers that will be defined on a system.

5. Make djxlink.makefile readable by entering the following command:

   ```
   chmod uog+r data access module
   ```

   where *data access module* is the name of the data access module that you are linking.

If you are installing DataJoiner and this is the first time that you ran djxlink.makefile, you can skip the remaining steps. Otherwise, complete the following steps for your existing DataJoiner instances.

6. For existing DataJoiner instances, run the **db2iupdt** command:

   ```
   /usr/lpp/djx_02_01_01/instance/db2iupdt instanceName
   ```

   or

   ```
   /opt/IBMdjx/V2.1.1/instance/db2iupdt instanceName
   ```

   where *instanceName* is the login name of the instance owner. You must run the **db2iupdt** command for each instance that you have defined. If you do not run **db2iupdt** for an instance, that instance will not pick up the change created by djxlink.makefile.

7. If you stopped any existing instances in step 1 on page 50, restart them by entering **db2start** for each instance.

When you run djxlink.makefile, several screens of warning messages might be displayed. You can ignore the following messages:

- 0706-224 - Duplicate symbols
- 0706-223 - Replacement of local definitions
- 0706-221 - Replacement of import version
- 0711-327 - Entry point not found: __start
- 0706-406 - Symbols imported

These messages indicate a successful link:

- 0706-297 - No unresolved symbols detected
- 0706-476 - RC: Highest return code was 0

Step 4: Adding DataJoiner Options after Initial Installation

This step applies to DataJoiner for AIX systems only. When you install a software product or software product options on AIX 3.2, you can either apply (default) or commit the options. Applied options can be rejected or committed at a later date. Committed options are much more difficult to remove from the system. See "Deleting Options on AIX 3.2" on page 192 for more information.

Therefore, be sure to commit DataJoiner options only if you need to free disk space and you do not plan to remove the software from the system.

Attention: After an option is committed on AIX 3.2, you cannot reapply the option unless it is a newer version, release, or modification level or unless you force the installation by using the -F flag of installp. For example:

```
installp -F -d /dev/cd0 djx_02_01_01.djx_dba
```

On AIX 4.1, products are automatically committed. However, an uninstall option is available to delete installed products. See "Deleting Options on AIX 4.1" on page 193 for more information.

After you have successfully installed DataJoiner, you must create an instance of DataJoiner before you can use the product. How to create an instance is explained in "Chapter 6. Setting up an Instance" on page 53.

Chapter 6. Setting up an Instance

A DataJoiner server is referred to as an *instance* of the product. You must create or migrate a DataJoiner instance to use the product. You might also want to create additional instances on the same system in the future, for example:

- To differentiate between a production environment and a test environment
- To differentiate between groups of users to which you want to assign different levels of security
- To differentiate between certain groups of data sources that require different performance settings

The following steps explain how to set up an instance of DataJoiner:

- "Step 1: Create or Assign Groups and Users"
- "Step 2: Create a DataJoiner Instance" on page 57
- "Step 3: Set Environment Variables" on page 58
- "Step 4: Set AIX Processes" on page 61
- "Step 5: Start the Instance" on page 62
- "Step 6: Create a DataJoiner Database" on page 62
- "Step 7: Create Links for DataJoiner Libraries" on page 66
- "Step 8: Connect DataJoiner to Data Sources" on page 66
- "Step 9: Connect Clients to DataJoiner" on page 66
- "Step 10: Enable a DataJoiner Database for Spatial Data" on page 67

Step 1: Create or Assign Groups and Users

This step explains how to create user and group names for both AIX and Solaris systems. It also describes the rules for creating valid user and group names.

Naming Rules

In addition to the rules that are imposed by the operating system, the user and group names that you create must adhere to the following rules:

- User and group names can contain only the following characters:
  - 0 through 9
  - a through z
  - @, #, or $
- User and group names must not start with:
  - a number (0 - 9)
  - ibm
  - sql
  - sys (allowed for group names)
- Group names must not start with:
  - admins
  - guests
  - local
  - users
- User and group names must not be the same as an SQL reserved word.

Go to the section that applies to your operating environment:

- "Create or Assign Groups and Users on AIX Systems"
- "Create or Assign Groups and Users on Solaris Systems" on page 56

Create or Assign Groups and Users on AIX Systems

You must perform this step as root. If you already identified an existing group and user, you can skip this step. If not, follow these steps to create a group, a user, and an instance:

1. Create a group that will be the primary group for the instance owner. You can create a group by issuing the **mkgroup** command or with smitty.
   - To use the **mkgroup** command to create a group that is named djadm1, enter:

     ```
     mkgroup djadm1
     ```

     Any user that belongs to this group has sysadm authority for this instance of DataJoiner.

- To use smitty to create a group:
  a. Login as root user.
  b. Enter smitty.
  c. Select **Security & Users** –> **Groups** –> **Add a Group**.
  d. Type a unique group name and press Enter. For example, type djadm1.
  e. Press F3 twice to return to the **Security & Users** menu.

2. Create a user to contain the instance. You can create a user by issuing the **mkuser** command or with smitty.
   - To use the **mkuser** command, enter mkuser followed by the appropriate parameters. For example:

     ```
     mkuser pgrp=djadm1 groups=djadm1 home=/home/djinst1 djinst1
     ```
   - To use smitty to create a user:
     a. Select **Security & Users** –> **Users** –> **Add a User**.
     b. Enter a unique user name. For example, djinst1.
     c. Enter the previously defined group name in the **Primary GROUP** field. In this example, use djadm1.
     d. Enter the home directory. For example, type: /home/djinst1, and press Enter.
3. Create a password for the user that you just created. You can create a password by issuing the **passwd** command or with smitty.
   - To use the **passwd** command to assign a password to the user that you just created, enter passwd followed by the user name. For example:

     ```
     passwd djinst1
     ```

     You will be prompted for a password.
   - To use smitty to assign a password to the user that you just created:
     a. Select **Security & Users** –>**Passwords** –> **Change a User’s Password**.
     b. Type the user name you just created.
     c. Type a password for the new user.
     d. Type the password a second time.
     e. Press F10 to exit smitty.
4. Optional: Create a group and user for fenced user-defined functions.

   Fenced user-defined functions (fenced UDFs) and DARIs will execute under this user and group. The group must be the primary group of the user. The user for fenced UDFs is specified as a parameter of the instance creation script. The group for fenced UDFs is implicitly set to the primary group of the specified user.

Recommendation: Use a separate user for fenced UDFs for security reasons.

If existing users and groups have been indentified for these purposes, you do not need to create new ones.

Create or Assign Groups and Users on Solaris Systems

You must perform this step as root.

1. Create a group that will be the primary group for the instance owner.

   Any user that belongs to this group will have system administrator authority for the instance. Use **admintool** or **groupadd** to create groups; for example, to create a group that is named djadm1, enter:

   ```
   groupadd djadm1
   ```

2. Create a user that will be the instance owner. This user name will be the name of the instance. Make this user's primary group the group that you just created.

   Recommmendation: Dedicate the instance owner user ID to that instance's use only. This allows for easier error recovery if a system error occurs.

   Use **admintool** or **useradd** to create users. For example, to create a user that is named djinst1 and assign it to the djadm1 group, enter:

   ```
   useradd -g djadm1 -d /export/home/djinst1 -m djinst1
   ```

3. Create a password for the user that you just created by issuing the **passwd** command; for example:

   ```
   passwd djinst1
   ```

   You will be prompted for a password.

4. Optional: Create a group and user for fenced user-defined functions.

   Fenced user-defined functions (fenced UDFs) and DARIs will execute under this user and group. The group must be the primary group of the user. The user for fenced UDFs is specified as a parameter of the instance creation script. The group for fenced UDFs is implicitly set to the primary group of the specified user.

   Recommendation: You can use the instance owner instead of creating a new user, use a separate user for fenced UDFs for security reasons.

   If existing users and groups have been indentified for these purposes, you do not need to create new ones.

Step 2: Create a DataJoiner Instance

To create a DataJoiner instance, issue the **db2icrt** command.

For DataJoiner for AIX systems, this command is found in `/usr/lpp/djx_02_01_01/instance`; for DataJoiner for Solaris systems, this command is found in `/opt/IBMdjx/V2.1.1/instance`. For example, to issue the command on a DataJoiner for AIX system, you enter:

```
/usr/lpp/djx_02_01_01/instance/db2icrt instanceName -a Authtype -u UdfName
```

where:

instanceName
: Is the login name of the instance owner. In the examples in this section, it is `djinst1`.

-a
: Is an optional parameter that specifies the authentication type of the instance. Valid authentication types are SERVER, CLIENT, and DCS. If the **-a** parameter is not specified, the authentication type will default to SERVER.

-u
: Is a parameter under which fenced UDFs and DARIs will execute.

For DataJoiner for AIX systems, this is an optional parameter; for DataJoiner for Solaris systems, this is a required parameter.

Restriction:

UdfName cannot not be **root** or **bin**.

If *UdfName* is unspecified, the user and group default to nobody:nobody; this means that there are no file access permissions defined for UDFs or DARIs.

You can the *UdfName* later with the **db2iupdt** command, if you do not set it now.

Executing the **db2icrt** command creates the *INSTHOME*/`sqllib` directory, where *INSTHOME* is the home directory of the instance owner. In the example, *INSTHOME* is `/home/djinst1`.

You should not place files or directories under *INSTHOME*/`sqllib` except those that are created by DataJoiner or those that are specified in DataJoiner documentation. Keeping the *INSTHOME*/`sqllib` directory clean prevents a loss of data if an instance is deleted.

Exception:

You can use the *INSTHOME*/`sqllib/function` directory to store UDFs and DARI procedures.

Step 3: Set Environment Variables

This step describes how local clients gain access to DataJoiner and how environment variables are set for an instance. It is not required for remote clients.

DataJoiner builds two script files when the instance is created and puts the files under the instance owner's home directory, in a directory that is named `sqllib`. These scripts are db2profile for the Bourne and Korn shells and db2cshrc for the C shell.

Use these files to set up the database environments. These scripts can be invoked from a local user's and the instance's .profile or .login file. It is important to use only the script file that is appropriate to the shell that you use in your environment. Remember to either log out and log in again or execute the .profile to invoke the db2profile.

Customizing the Scripts

An instance owner or sysadm user can customize the db2profile or db2cshrc script files for all users of an instance. Alternatively, each user can copy and customize these scripts to a specific environment.

1. Update the PATH environment variable by adding the following directories:

   ```
   INSTHOME/sqllib/bin
   INSTHOME/sqllib/adm
   INSTHOME/sqllib/misc
   ```

   where `INSTHOME` is the home directory of the instance owner.
2. Set the DataJoiner environment variables. Both local clients and DataJoiner instances use db2profile. Of the environment variables that are listed here, DB2INSTANCE is the only one a local client must set or change to access a DataJoiner instance. A local client can accept the default values for all other environment variables. Some of these variables are only used by DataJoiner instances. You can either accept the defaults or change the values based on your needs. The environment variables are:
   - **DB2BQTIME** Default = 1 second. Minimum = 1 second.

     Specifies the amount of time the DataJoiner command line processor front end sleeps before checking if the back end process is active and establishing a connection.
   - **DB2BQTRY** Default = 60 retries. Minimum = 0 retries.

     Specifies the number of times the DataJoiner command line processor front end process tries to determine whether the back end process is already active. It works in conjunction with DB2BQTIME.

- **DB2CHKPTR** Default = null (OFF). Any Value = (ON).

  Selectively turns pointer checking ON for debugging. It should be left OFF for normal operation.
- **DB2COMM** Default = null. Values: APPC, IPXSPX, NetBIOS, TCPIP, NONE, or others.

  Controls which, if any, communication protocols are enabled when the DataJoiner server is started. For example, to enable APPC and TCP/IP, set DB2COMM to:

  ```
  export DB2COMM=APPC,TCPIP
  ```

  The NONE setting takes precedence over all other values. If NONE is specified, no communication support is initialized, regardless of other specified values.

  You can set and change DB2COMM at any time; a new setting becomes effective on the next **db2start** command. This environment variable is not used by a local client.

  Set this variable only after the database manager configuration file is updated appropriately. For example, to set DB2COMM to APPC, set the TPNAME variable in the database manager configuration file; to set DB2COMM to TCPIP, set the SVCENAME variable in the database manager configuration file. If DB2COMM is set before the appropriate variable is set in the database manager configuration file, the database manager will not start.
- **DB2DBDFT** Default = SAMPLE.

  Specifies the database alias name of the DataJoiner database to which applications are implicitly connected when started.
- **DJXCOMM** Default = null (OFF). Value: *data-access-module*.

  Specifies whether or not a data access module is preloaded into the DataJoiner instance. By default, DataJoiner loads a data access module only at the moment that a data source is accessed. DJXCOMM tells DataJoiner to load a data access module when DataJoiner is started. Preloading data access modules can improve performance because the data access module is only loaded once between the time DataJoiner is started and stopped.

  Multiple data access modules can be loaded using DJXCOMM.

  For example:

  ```
  export DJXCOMM = sqlnet,net8,drda
  ```

  Possible values include drda, db2ra, informix, informix7, djxclassic, and others. These values are the same as those that are found in the

SERVER_PROTOCOL column of SYSCAT.SERVERS. See “Chapter 7. Configuring Access to Data Sources” on page 71 for more information about data access modules.

- **DJX_CC2_CONFIG** Default = null, Value: name of the Classic Connect config file.

  This environment variable is used only by remote clients and specifies the complete path and file name of the configuration file for the DataJoiner Classic Connect interface. If set and the file exists, support for Classic Connect is enabled. If null, not set, or DataJoiner cannot find the file, the Classic Connect interface cannot be used.

  The Classic Connect interface is used to access data sources that are defined to DataJoiner Classic Connect for MVS. A sample setting is:

  ```
  DJX_CC2_CONFIG=instancehomedir/sqllib/djxclassic2.cfg
  ```

  See the *DataJoiner Classic Connect Installation, Configuration, and Reference Guide* V2.1.1 for more information.

- **DB2INSTANCE** Default = instance_name.

  Specifies the instance that is active by default.

- **DB2IQTIME** Default = 5 seconds. Minimum = 1 second.

  Specifies the amount of time the DataJoiner command line processor back end process waits on the input queue for the front end process to pass commands.

- **DB2RQTIME** Default = 5 seconds. Minimum = 1 second.

  Specifies the amount of time the DataJoiner command line processor waits for a request from the front end process.

Invoking the Scripts

There are several methods to invoke one of these scripts. For a Bourne or Korn shell, choose one of the following techniques:

- Concatenate db2profile with the user’s .profile by entering:

  ```
  cat /home/djinst1/sqllib/db2profile >> /home/djinst1/.profile
  ```

- Execute db2profile from the user’s .profile by typing a new line into the .profile:

  ```
  . /home/djinst1/sqllib/db2profile
  ```

- Manually execute the script when connection to a database is needed by entering:

  ```
  . /home/djinst1/sqllib/db2profile
  ```

- Develop a customized procedure for users. If you choose this option, be sure to run **db2 terminate** before switching instances.

For a C shell, choose one of the following techniques:

- Concatenate db2cshrc with the user’s .login or .cshrc by typing:

  ```
  cat /home/djinst1/sqllib/db2cshrc >> /home/djinst1/.login
  ```

  or

  ```
  cat /home/djinst1/sqllib/db2cshrc >> /home/djinst1/.cshrc
  ```
- Execute the **db2cshrc** command from the user’s .login by typing a new line into the .login:

  ```
  source /home/djinst1/sqllib/db2cshrc
  ```
- Manually execute the script when connection to a database is needed by entering:

  ```
  source /home/djinst1/sqllib/db2cshrc
  ```
- Develop a customized procedure for users. If you choose this option, be sure to run **db2 terminate** before switching instances.

Step 4: Set AIX Processes

This step applies to DataJoiner for AIX systems only. The AIX operating system provides mechanisms that can restrict the number of processes that can execute under an owner id. This limit is controlled through the AIX parameter maxuproc, which has a default setting of 40. The default limit of 40 might not be sufficient if the system is required to support many databases and connections.

The following error occurs during connection to the DataJoiner database if the maxuproc value is exceeded:

```
SQL1224N A database agent could not be started to service a request
or was terminated as a result of a database system shutdown or a force
command.
```

To see the current definition of maxuproc, enter:

```
lsattr -E -l sys0
```

To view the number of processes that are currently executing on the system, enter:

```
ps -ef|grep instdj1|wc -l
```

where *instdj1* represents your DataJoiner instance owner’s id.

To change the value of maxuproc, enter:

```
chdev -l sys0 -a maxuproc='nn'
```

where *nn* is the new integer value of the parameter.

Step 5: Start the Instance

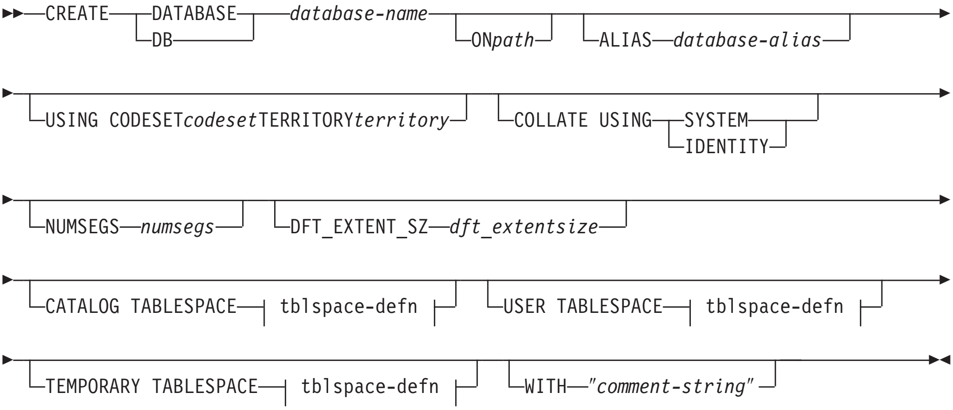
Start the DataJoiner instance by entering:

```
db2start
```

Step 6: Create a DataJoiner Database

DataJoiner uses the catalog from a local database to maintain information about the DataJoiner heterogeneous environment. Therefore, you must create a DataJoiner database.

The **CREATE DATABASE** command has the following syntax:



```
►►──CREATE──┬─DATABASE─┬──database-name──┬──────────┬──┬─────────────────────────┬──────►
            └─DB───────┘                 └─ONpath───┘  └─ALIAS──database-alias───┘

►──┬────────────────────────────────────────────┬──┬──────────────────────────────┬────►
   └─USING CODESETcodesetTERRITORYterritory─────┘  └─COLLATE USING──┬─SYSTEM───┬──┘
                                                                    └─IDENTITY─┘

►──┬──────────────────────┬──┬────────────────────────────────┬────────────────────────►
   └─NUMSEGS──numsegs─────┘  └─DFT_EXTENT_SZ──dft_extentsize──┘

►──┬─────────────────────────────────────────┬──┬──────────────────────────────────────┬──►
   └─CATALOG TABLESPACE──┤ tblspace-defn ├───┘  └─USER TABLESPACE──┤ tblspace-defn ├───┘

►──┬───────────────────────────────────────────┬──┬─────────────────────────────┬──────►◄
   └─TEMPORARY TABLESPACE──┤ tblspace-defn ├───┘  └─WITH──"comment-string"──────┘
```

Figure 7. CREATE DATABASE Command Syntax

Notes:

1. The code set and territory values that are specified must be a valid combination. For a list of valid combinations, see the DB2 Installation and Operation Guide for your platform.
2. For details on the **tblspace-defn** parameters, see the CREATE TABLESPACE statement in the *DB2 SQL Reference*.

Parameters:

database-name

Represents the name to be assigned to the database you are creating. This must be a unique name that differentiates the database from any other databases that exist within the DataJoiner instance. The name must conform to naming conventions for databases. Refer to the *DataJoiner Administration Supplement* for more information about naming conventions.

path Specifies the path on which to create the database.

If you do not specify a path, the database is created in the default database path as specified in the database manager configuration file (dftdbpath parameter). To see the value of dftdbpath, use the **GET DATABASE MANAGER CONFIGURATION** command as documented in the *DB2 Command and API Reference*. The default database path is the HOME directory of the owner of the DataJoiner instance you are using. The path can be a maximum of 215 characters.

database-alias
An alias for the database in the system database directory. If no alias is provided, the specified database name is used.

codeset **and** *territory*
In most cases, you do not need to specify these options.

If the code set and territory are not specified on the create database command, DataJoiner uses the territory and code set of the application that is invoking the command. The application code page is derived from the active environment when the database connection is made.

In UNIX-based environments, the active environment is determined from the environment variables that are associated with the locale, which include information about language, territory, and code set.

If your database needs a code set and territory different from what is described, specify them in the **CREATE DATABASE** command. Examples of code set and territory are:

```
Database code set = IBM-850
Database territory = En_US
```

For more information about code sets and territories that are supported by DataJoiner, see the discussion of National Language Support in “Appendix C. National Language Support” on page 317.

COLLATE USING
Identifies the type of collating sequence to be used for the database. Possible collating sequence types are:

SYSTEM
Indicates that the collating sequence is based on the current country code. This is the default.

IDENTITY
Indicates that the collating sequence is the identity sequence, where strings are compared byte for byte.

When choosing the collating sequence, performance and data usage should be considered. However, when storing binary data (DB2 datatype CHAR(nn) FOR BIT), then IDENTITY should be used. You must use IDENTITY if the DataJoiner database will be used for replication. See the *DataJoiner Administration Supplement* for details on collating sequences.

numsegs **and** *segpages*

In most cases, you do not need to specify these options. The one exception is when you plan to store a large amount of user data in the local DataJoiner database.

Your AIX file system may have limitations on the size of the files and directories. To overcome this limitation, DataJoiner lets you create databases that span multiple file systems by distributing databases and tables across multiple disk segments.

For more information, see the *DB2 Administration Guide*.

dft_extentsize

Specifies the default extent size of table spaces in the database.

CATALOG TABLESPACE *tblspace-defn*

Specifies the definition of the table space that will hold the catalog tables, SYSCATSPACE. If not specified, SYSCATSPACE will be created as a System Managed Space (SMS) table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*.

USER TABLESPACE *tblspace-defn*

Specifies the definition of the initial user table space, USERSPACE1. If not specified, USERSPACE1 will be created as an SMS table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*.

TEMPORARY TABLESPACE *tblspace-defn*

Specifies the definition of the initial user table space, TEMPSPACE1. If not specified, TEMPSPACE1 will be created as an SMS table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*.

comment-string

Describes the database or the database entry in the system directory. Do not use a carriage return or line feed character in the comment. Be sure to enclose the comment text in double quotes. The comment can be a maximum of 30 characters.

Create the Database

From the login for the DataJoiner instance that you created in "Step 1: Create or Assign Groups and Users" on page 53, issue the **CREATE DATABASE** command. For example, type:

```
db2 create database djdb1 on /dbland2
```

When you issue a **CREATE DATABASE** command, DataJoiner performs the following actions:

1. Creates the database.
2. Creates the system catalog tables and recovery log.
3. Catalogs the database in the local database directory on the path where the database was created and in the system database directory. If the database directories do not already exist, they are created. See the *DataJoiner Administration Supplement* for more information on database directories.

   The database is cataloged with an alias that is the same as the database name. To assign another alias to the database, you can first uncatalog the database and then use the **CATALOG DATABASE** command, which are documented in the *DB2 Command and API Reference*.
4. Grants DBADM authority and CONNECT, CREATETAB, and BINDADD privileges to you as the database creator.
5. Assigns the code set and territory.
6. Calls the BIND executor to bind pre-defined DataJoiner bind files to the database.
7. Grants the BIND and EXECUTE privilege as PUBLIC for DataJoiner utilities, such as the Command Line Processor (CLP).

   With DBADM authority, you can grant these privileges to (and revoke them from) other users or PUBLIC. If another administrator with SYSADM or DBADM authority over the databases revokes these privileges, you, as the database creator, retain your privileges.

When you create a database that will be accessed by most of your users, you might want to update the DB2DBDFT environment variable as described in "Step 3: Set Environment Variables" on page 58. If so, set DB2DBDFT to the name of the newly created database.

Spatial Extender users:

If you plan to run the sample application that is provided with the Spatial Extender, name your database ″sample.″ Refer to the *IBM DB2 Spatial Extender Administration Guide and Reference* for more information.

Step 7: Create Links for DataJoiner Libraries

This step applies to DataJoiner for Solaris systems only. It is optional and not recommended.

Use the **db2ln** command to create links for libraries in `/opt/IBMdjx/V2.1.1/lib` to `/usr/lib`, and for include files in `/opt/IBMdjx/V2.1.1/include` to `/usr/include` for a particular version and release level of DataJoiner.

You might want to create these links if you are developing or running applications to avoid having to specify the full path to the product libraries and include files.

If there are existing links to `/usr/lib` and `usr/include` from Version 1, they will automatically be removed by executing **db2ln** to create links for Version 2. If you want to re-establish the links to the Version 1 libraries, then you must execute the **db2rmln** command from Version 2 before you execute the **db2ln** command from Version 1.

Links can only be established for one release of DataJoiner on a each system.

To invoke the **db2ln** command, enter:

```
/opt/IBMdb2/V2.1/cfg/db2ln
```

Step 8: Connect DataJoiner to Data Sources

This step is covered fully in “Part 3. Connecting DataJoiner to Data Sources” on page 69. After reading and following the instructions on Part 3, proceed to “Step 9: Connect Clients to DataJoiner”.

Step 9: Connect Clients to DataJoiner

Local clients require minimal configuration. For example, local clients do not require communication configuration. However, you must run the db2profile shell script as discussed in “Step 3: Set Environment Variables” on page 58.

You can either invoke the shell script directly or add it to a local client’s .profile so that the database environment is established during login. You can also copy the script into a local client’s directory structure and customize it.

A common approach is to change the setting of the DB2INSTANCE environment variable in a local client’s db2profile. Multiple DataJoiner instances can be defined on a given system. DB2INSTANCE records the instance that a client wants to access. If a client has access to multiple DataJoiner instances, you must modify the value of DB2INSTANCE and rerun db2profile before you change instances.

Remote clients require either the DB2 Client Application Enabler (CAE) or the DB2 Software Developer's Kit (SDK) V2.1.1 or later to communicate with a DataJoiner server. Client access is available from any DB2 Family product that provides function equivalent to the DB2 CAE 2.1.1. DataJoiner for AIX supports both APPC (SNA) and TCP/IP clients. Please refer to your client documentation to determine the protocol to use to access DataJoiner.

Step 10: Enable a DataJoiner Database for Spatial Data

| If you are using DataJoiner for AIX, and you plan to work with spatial (GIS)
| data on a DataJoiner database, you must perform this step. See the *IBM DB2*
| *Spatial Extender Administration Guide and Reference* for instructions.

Part 3. Connecting DataJoiner to Data Sources

Chapter 7. Configuring Access to Data Sources

DataJoiner uses data access modules to communicate with data sources. A different data access module is required for each data source type.

Data access modules talk to a data source through the access methods that are provided by the data source for its clients. Therefore:

- A data source must be configured to accept clients through a specific access method.
- For some data sources, DataJoiner must use client software that is provided by the data source vendor.

Use Table 5 to determine which access methods apply to your installation.

Table 5. Data Sources and Their Access Methods

Data Source	**Access Method**	**See...**
DataJoiner V1	IBM DB2RA	page 107
DataJoiner V1.2 and V2	IBM DB2RA or DRDA	page 107 (for DB2RA) or page 79 (for DRDA)
DB2 Universal Database V5	IBM DB2RA or DRDA	page 107 (for DB2RA) or page 79 (for DRDA)
DB2 for OS/390, DB2/MVS, DB2 for OS/400, SQL/DS and DB2 for CS V2[1,2,3]	DRDA	page 79
DB2 for CS (except DB2 for OS/2 V1 and DB2 Parallel Edition)	IBM DB2RA	page 107
DB2 Parallel Edition	IBM DB2RA or DRDA	page 107 (for DB2RA) or page 79 (for DRDA)
IMS[4]	Classic Connect	the *DataJoiner Classic Connect Installation, Configuration, and Reference Guide* V2.1.1
Informix OnLine V5[4]	informix data access module	page 123
Informix OnLine V7.1 or higher (non-GLS, -XPS, and -SE versions only)	informix7 data access module	page 129

Table 5. Data Sources and Their Access Methods (continued)

Microsoft SQL Server V6.5 or lower	Sybase Open Client dblib	page 141
Microsoft SQL Server V6.5[4]	MERANT DataDirect Connect ODBC	page 147
Microsoft SQL Server V7.0[4]	MERANT DataDirect SequeLink ODBC Edition	page 150
ODBC/XOpen CLI	Generic data access module	page 171
Oracle	Oracle SQL*Net or Net8	page 135
Sybase	Sybase Open Client	page 141
Teradata[4]	Teradata data access module	page 155
VSAM[4]	Classic Connect	the *DataJoiner Classic Connect Installation, Configuration, and Reference Guide* V2.1.1
Other relational or nonrelational data sources[4]	CrossAccess	page 165

Notes:

1. Although you can use DRDA to access DB2 for CS V2, IBM DB2RA is the recommended access method for DB2 for CS.
2. DRDA application server data sources that will be participating in two-phase commit transactions must use the DB2RA protocol; see "Configuring for Two-Phase Commit" on page 73.
3. DRDA3 application server data sources (DB2 for OS/390 and DB2 for OS/400 V4.2) can also use the drdaIP protocol. Two-phase commit is not supported for this protocol.
4. DataJoiner for Solaris systems does not support this data source.

Link-Editing Data Access Modules

To access many data sources, DataJoiner libraries must be link-edited with data source client libraries through the djxlink.sh script. For most data sources, the djxlink.sh script builds an executable file that is called a data access module. DataJoiner communicates with data sources through the use of data access modules. The data access module name is used as the protocol name in the CREATE SERVER MAPPING statement when you configure DataJoiner for a data source.

Exception:

DB2 data sources do not need to be link-edited with DataJoiner libraries.

DataJoiner is designed to link-edit data access modules directly to remote data sources client code. See “Step 3: Link DataJoiner to Data Source Client Software” on page 46 for more information.

Loading Data Access Modules

You can control whether or not a data access module is loaded at DataJoiner initialization through the use of the DJXCOMM environment variable. Loading a data access module at initialization reduces system overhead by eliminating the need to load and unload a module each time a data source is accessed. See “Step 3: Set Environment Variables” on page 58 for more information.

Configuring for Two-Phase Commit

This section shows you how to enable data sources to participate in two-phase commit transactions. Not all versions of all data sources supported by DataJoiner are capable of two-phase commit. Also, DataJoiner for Solaris systems does not support two-phase commit transactions. Table 6 shows which data sources support two-phase commit and lists additional requirements for certain data sources. The second column indicates if a data source, with the listed constraints, can participate in a two-phase commit transaction. The third column indicates if the data source supports the creation of new objects at the data source (using a pass-through session) while participating in a two-phase commit transaction. For more information about two-phase commit concepts and terminology, see the *DataJoiner Administration Supplement*.

Table 6. Data Sources Supporting Two-Phase Commit

Data Source	Supports two-phase commit?	Supports two-phase commit DDL?
Classic Connect	No	No
Cross Access	No	No
DB2 CS and DataJoiner, Version 1.2	Yes	Yes
DB2 CS and DataJoiner, Version 2	Yes	Yes
DB2 for OS/390, Version 3 and higher (with APARs PN67179 and PN70102)	Yes	Yes
DB2 for OS/400, before Version 3.1	No	No
DB2 for OS/400, Version 3.1 and higher	Yes	Yes

Table 6. Data Sources Supporting Two-Phase Commit (continued)

Data Source	Supports two-phase commit?	Supports two-phase commit DDL?
EDA/SQL	No	No
Generic Access API data sources	No	No
Informix Version 5	Yes	Yes
Informix, Versions 7.1 and 7.2, with TP/XA library	Yes	Yes
MS SQL Server (DBLIB)	No	No
MS SQL Server 4.2 (CTLIB) with XA library	No	No
MS SQL Server 6.5 (ODBC), using DataJoiner on NT	No	No
MS SQL Server 6.5 (ODBC), using DataJoiner V2 on UNIX operating systems	No	No
Oracle V7 and higher, without the XA library distributed database option	No	No
Oracle V7 and higher, with the XA library and the distributed database option on UNIX operating systems	Yes	No
Oracle V7.3 and higher, on Windows NT 3.51 and 4.0 operating systems (requires V7.3.3 Oracle Client at DataJoiner on NT operating systems)	Yes	No
Oracle RDB (all levels)	No	No
SQL Anywhere	No	No
SQL/DS, before Version 4.1	No	No
SQL/DS, Version 4.1 and higher	Yes	Yes
Sybase SQL Server (CTLIB) with XA library	Yes	No
Sybase SQL Server (DBLIB)	No	No
Teradata	No	No

Enabling Data Sources for Two-Phase Commit

Perform the following steps for data sources that will be participating in two-phase commits:

- For all data sources participating in two-phase commit transactions, define a transaction resolution password and ID for resynch operations. This is a special user name and password with the authority to COMMIT and ROLLBACK transactions. Then, using the CREATE USER MAPPING SQL statement, map this ID to the special ID, SYSTMDB.

  DataJoiner will always use the SYSTMDB ID for the resynch agent that connects to RMs during resynch processing. This ID must be mapped to a remote user ID and password that has the authority to commit or rollback the transaction initiated on remote tables by users through DataJoiner.

For more information about CREATE USER MAPPING statements, see the *Application Programming and SQL Reference Supplement*.

- For non-DB2 data sources (Informix, Oracle, Sybase ctlib) install the required XA libraries for the data source.

  For Sybase ctlib data sources, you must update the `xa_config` file to map Sybase node names to Logical Resource Manager (LRM) names. If a DataJoiner user has write authority for the `xa_config` file, this step is performed automatically by DataJoiner when the CREATE SERVER MAPPING statement is issued for the data source. If a user does not have write authority for the `xa_config` file, this step must be performed manually by the Sybase administrator.
- For DRDA server data sources, two-phase commit transactions require a DDCS gateway, which can be either standalone DDCS or DDCS as supplied with DataJoiner for AIX.

  Restriction: Neither DDCS for Windows NT nor the DDCS functionality packaged with DataJoiner for Windows NT can be used as a DDCS gateway to DRDA server data sources using two-phase commit. Two-phase commit transactions from DataJoiner for Windows NT must go through a non-Windows NT DDCS gateway, such as DDCS for AIX or OS/2, or through DataJoiner for AIX.

  There are three possible ways that you can use DDCS to access DRDA server data:

  – Use an existing DDCS gateway that is installed as a standalone product. For customers who already have DDCS installed and configured, this is the simplest method.

  – Use a separate DataJoiner instance that is installed as a DDCS gateway. This method allows you to use DataJoiner as a DDCS gateway that can be shared among multiple DataJoiner instances for transactions involving two-phase commit.

  – Use the same DataJoiner instance as a DDCS gateway. When DataJoiner requests data from a DRDA server data source, that request is looped back to itself and handled by its DDCS gateway functionality. For customers who do not currently own standalone DDCS, this is the simplest method because it does not require a separate DataJoiner instance.

No matter which method you choose, the set up procedure for DDCS is the same. For information about configuring DDCS for two-phase commit, see the *DDCS for AIX Installation and Configuration Guide*.

Configuring DataJoiner for Two-Phase Commit

This section describes the steps you must perform at the DataJoiner instance for two-phase commit transactions.

- For non-DB2 data sources, link-edit the XA libraries with the DataJoiner libraries by adding the following lines to the appropriate section of djxlink.makefile as follows:

  Informix (7.1 and higher only)
  Add `libinfxxa.o`

  Oracle Add `-lxa`

  Sybase (ctlib)
  – Change `-l comn` to `-l comn.so`
  – Change `-l cs` to `-l cs.so`
  – Add `-lxa`

- For DRDA server data sources, perform the following steps at the DataJoiner instance:

  1. Catalog a TCP/IP node entry pointing from DataJoiner to the DDCS gateway; for example:

     ```
     CATALOG TCPIP NODE ddcs_node_name REMOTE ddcs_machine_name
     SERVER ddcs_port_name
     ```

     where *ddcs_node_name* is a unique name for the DDCS gateway, *ddcs_machine_name* is the name of the system where the data source resides, and *ddcs_port_name* is the primary port name for use by data source clients.

  2. Define the DDCS gateway to DataJoiner by issuing a CREATE SERVER MAPPING statement; for example:

     ```
     CREATE SERVER MAPPING FROM server_name TO NODE "ddcs_node_name"
     DATABASE "ddcs_dbname" TYPE DB2/MVS VERSION 3.1 PROTOCOL "db2ra"
     ```

     Notice that `db2ra` is specified as the protocol. This is because DataJoiner is accessing a DDCS instance as opposed to accessing the DB2/MVS data source directly.

  3. Enable each data source that will participate in two-phase commits by issuing a CREATE SERVER OPTION statement; for example:

     ```
     CREATE SERVER OPTION two_phase_commit FOR SERVER server_name SETTING 'Y'
     ```

     Alternately, you can modify the two-phase commit option at runtime by issuing the SET SERVER OPTION command.

  See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about the mapping statements that are described in this section.

Overview of Configuration Steps

This section describes the steps that are required to configure DataJoiner to access data sources. The steps are similar for all data sources:

1. If necessary, install client software on the DataJoiner server.
2. Configure network communications.
3. Identify the node where the data source resides to DataJoiner.
4. Update DataJoiner's catalog views with information about the data source.
5. Test the connection to the data source.

DataJoiner can also be configured to access DB2 data sources directly, without the use of nicknames or pass-through sessions. The configuration steps for this method are described in detail in "Accessing DRDA Data Sources using DataJoiner DDCS" on page 102 and "Accessing DB2RA Data Sources Outside the Heterogeneous Environment" on page 118.

Chapter 8. Accessing DRDA Data Sources

There are two ways to access data sources through the DataJoiner DRDA Application Requester function:

- DataJoiner provides access through the use of nicknames and the pass-through facility. This method is explained in “Accessing DRDA Data Sources Using Nicknames and Pass-Through Statements”.
- DataJoiner provides access for applications that use IBM’s Distributed Database Connection Services (DDCS). This method is required by data replication and applications that must access the data source directly, and is explained in “Accessing DRDA Data Sources using DataJoiner DDCS” on page 102.

Accessing DRDA Data Sources Using Nicknames and Pass-Through Statements

This section shows how to configure DataJoiner to DB2 for OS/390 and DB2 for AS/400 data sources. Other DRDA data sources are configured in a similar manner. For detailed information on other DRDA data sources, refer to the *DRDA Connectivity Guide* (SC26-4783).

The process of configuring DataJoiner and DRDA data sources for two-phase commit transactions is explained in “Configuring for Two-Phase Commit” on page 73.

The steps that are required to access a DRDA data source through the use of nicknames and the pass-through facility are:

- “Step 1: Configure Network Communications” on page 80
- “Step 2: Update the Node Directory” on page 98
- “Step 3: Update DataJoiner Catalog Views with Server and User Information” on page 99
- “Step 4: Test the Connections to the Data Source” on page 102

Figure 8 on page 80 is an overview that shows how DataJoiner’s DRDA application requester accesses a data source. Figure 8 shows the NODE directory entry, in this case TOMVS. Node names are stored both in the catalog view SYSCAT.SERVERS and in the Node directory. The Node directory then references the SNA Configuration Side Information Profile, db2cplnn. The connection is initiated from there.

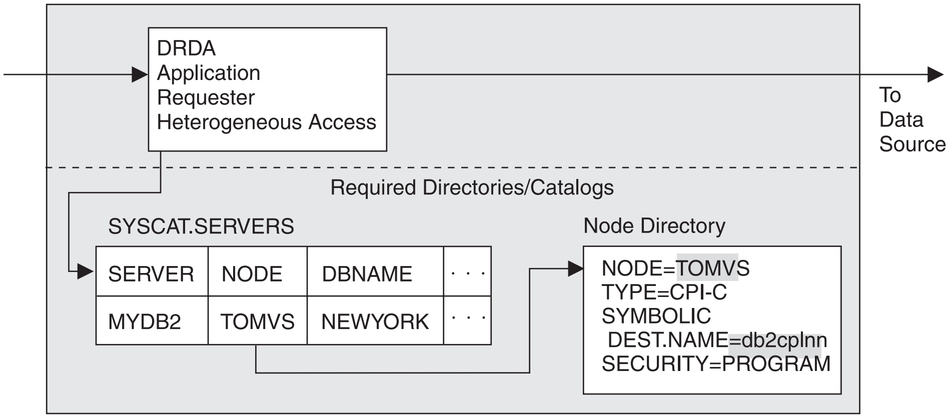


Figure 8. DataJoiner Application Requester Heterogeneous Access

The remainder of this section contains detailed instructions for the steps that were just discussed. It will be helpful to fill in the planning template that is provided as you configure your environment. These templates are designed for DB2 for OS/390 data sources, but other data sources are configured in a similar manner.

Step 1: Configure Network Communications

You can use the SNA LU 6.2 communications protocol to access DRDA data sources, and you can use either the SNA LU 6.2 or the TCP/IP protocol to access DRDA3 data sources.

- To configure TCP/IP, see “Configuring TCP/IP for DRDA3 Data Sources (DB2 for OS/390 and DB2 for AS/400)”.
- To configure SNA LU 6.2 for DataJoiner for AIX systems, see “Configuring IBM eNetwork Communication Server on AIX for DRDA Data Sources” on page 83.
- To configure SNA LU 6.2 for DataJoiner for Solaris systems, see “Configuring SunLink SNA PU 2.1” on page 91.

Configuring TCP/IP for DRDA3 Data Sources (DB2 for OS/390 and DB2 for AS/400)

This section shows you how to establish a connection from DataJoiner to a DRDA3 data source through TCP/IP. DRDA3 data sources include DB2/MVS V5.1 and later, DB2 for OS/390, and DB2 for AS/400 V4.2 and later.

Before you can configure DataJoiner to access a DRDA3 data source through TCP/IP, make sure that TCP/IP support is installed and configured for each participating DRDA3 data source. See the data source's installation guide for instructions.

Due to the characteristics of the TCP/IP protocol, the TCP/IP subsystem on one host might not be notified of the failure of its partner on a different host. As a result, a DataJoiner server accessing a data source using TCP/IP, or the corresponding process at the data source, might sometimes appear to be hung. You can use two network options, tcp_keepidle and tcp_keepintvl, to tune a specified time interval before the failure is detected.

The values for these options are system wide and apply to all TCP/IP socket applications for which the SO_KEEPALIVE socket option is enabled, not just DataJoiner.

To set or display the network options, use the `no` command.

Figure 12 on page 109 illustrates a typical TCP/IP configuration between DataJoiner and a DB2 for OS/390 data source:

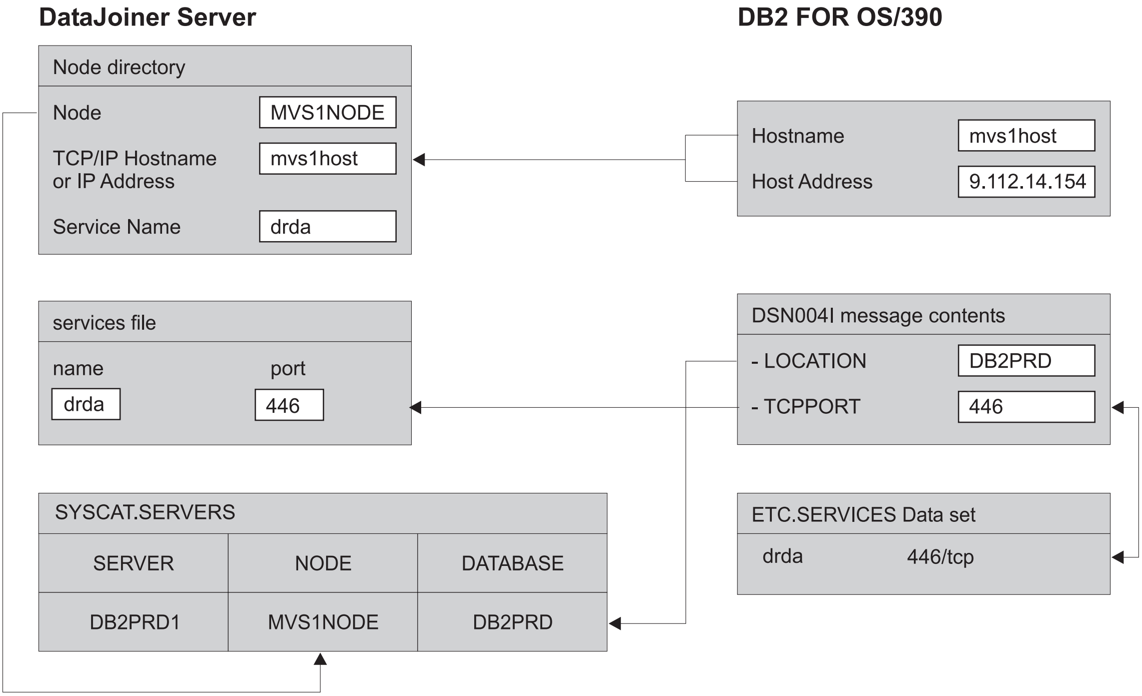


Figure 9. DB2 for OS/390 Data Source Configuration through TCP/IP

To configure TCP/IP for DRDA3 data sources (DB2 for OS/390 and DB2 for AS/400):

1. Determine the following values for the DRDA3 system:

- The TCP/IP host name or IP address of the system where the DRDA3 data source resides.
- The TCP/IP port number for the DRDA service as defined at the data source. By default, the port number for the DRDA service is 446.
- For DB2 for OS/390 data sources, the LOCATION name of the DB2 for OS/390 subsystem.

  This value can be obtained from the DDF start message (DSNL004I).
- For DB2 for AS/400 data sources, the RDB name of the AS/400 system.

  This value can be obtained from the WRKRDBDIRE display and is the relational database name of the record whose remote location is *LOCAL.
- A user ID and password for the DRDA3 system. The user ID that you use must have CONNECT, SELECT, and BINDADD privileges at the DRDA3 data source.

  Exceptions:

  – For DB2 for OS/390 data sources, a password is not required if `TCP/IP Already Verified => YES` was specified in the DDF DSNTIP5 setup panel.

  – For DB2 for AS/400 data sources, a password is not required if `DDM Password Required = NO` was specified on the CHGDDMTCPA display.

2. For AS/400 systems, verify that the following tasks have been performed:
   - The NULLID collection was created:

     ```
     CRTLIB LIB(NULLID)
     ```
   - Attributes of the DDM job are set.

     To determine if they are set, issue the **CHGDDMTCPA** command, and verify that the following values are displayed:

     ```
     Autostart server    *YES  *NO,*YES,*SAME
     Password required   *YES  *NO,*YES,*SAME
     ```

     If you receive security errors (3008N, reason code 0) when connecting to DB2 for AS/400 from DataJoiner, change the password setting for DDM and attempt reconnecting.
   - The DDM job was started with the **STRTCPSVR SERVER (*DDM)** command.

     To determine if it is running, issue the following command:

     ```
     WRKACTJOB SBS(QSYSWRK)
     ```

     and ensure that QRWTLSTN is displayed in the QSYSWRK subsystem.
   - Change the CCSID for the DataJoiner user ID to 37.

     ```
     CHGUSRPRF USRPRF(DJADM) CCSID(37)
     ```

3. Complete the following steps on the DataJoiner server workstation:
   a. Define the data source's port number at the DataJoiner server by either manually updating DataJoiner's /etc/services file, or by using the instructions in "Step 2: Update the Node Directory" on page 98.
   b. For DataJoiner for AIX V3.2 systems, synchronize the /etc/services file and the inetd daemon by issuing the following commands:

      ```
      inetimp
      refresh -s inetd
      ```

   c. If necessary, update /etc/hosts for the data source host name and IP address. This step must be performed as root.

      Whether you must update /etc/hosts depends on how TCP/IP address translation is done on your network. The remote host name that is specified on the **CATALOG TCPIP NODE** command must be translated to an IP address when it is used. If your network has a name server that recognizes the host name, the name server will perform the IP address translation, and you do not need to update /etc/hosts. Otherwise, you need an entry in /etc/hosts for the remote host name. See your network administrator to learn how your network is configured.

When you finish setting up TCP/IP network communications, continue the configuration process. If you are configuring for access through the use of nicknames and pass-through statements, go to "Step 2: Update the Node Directory" on page 98. If you are configuring for access using DataJoiner's DDCS functionality, go to "Step 2: Update the Node Directory" on page 103.

Configuring IBM eNetwork Communication Server on AIX for DRDA Data Sources

This section describes how to configure an APPC connection between your IBM eNetwork Communication Server workstation and a DRDA server. Before you begin, ensure that your workstation has IBM eNetwork Communication Server V5.0.2.5 for AIX installed. If you need further information to configure your SNA environment, refer to the online help provided with eNetwork Communication Server for AIX.

Planning for Configuring Your DRDA Servers

Figure 10 on page 84 provides a DRDA Data Source Planning Template for DB2 for OS/390 data sources for your assistance. Other data sources are configured in a similar manner.

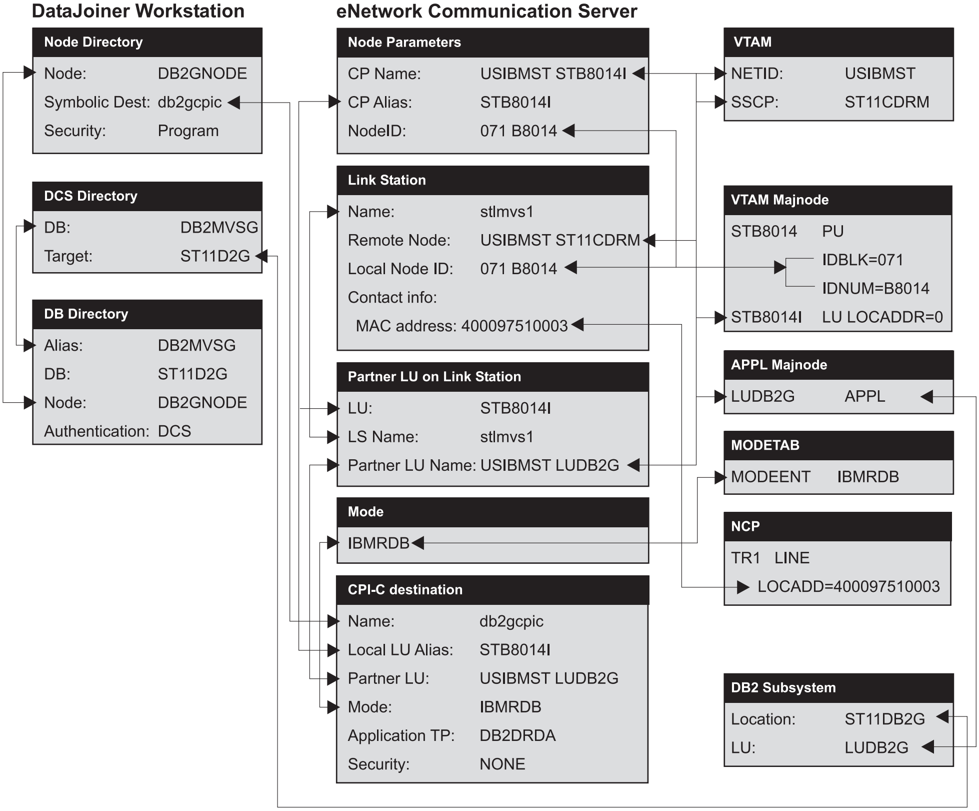


Figure 10. DRDA Data Source Planning Template

Before you configure the DataJoiner workstation, obtain the following information from your host and LAN administrators:

SNA network name

- For DB2 for OS/390, this value is the VTAM NETID.
- For DB2 for AS/400, this value is the local Network ID from the DSPNETA command.

The example value used in this section is USIBMST.

SNA control point name

- For DB2 for OS/390, this value is the SSCP name of the VTAM that loads the NCP software into the 3745 that has the token ring adapter that the DataJoiner workstation needs to communicate with to reach the DRDA server.
- For DB2 for AS/400, this value is the local Control Point (CP Name) from the DSPNETA command.

The example value used in this section is ST11CDRM.

Location or RDB name of the DRDA server

- For DB2 for OS/390, this value is the Location name on the DDF Start Message DSNL004I.
- For DB2 for AS/400, this value is the WRKRDBDIRE Relational Database name where Remote Location Name is *LOCAL.

The example value used in this section is DB2G.

LU name of the DRDA server

- For DB2 for OS/390, this value is the LU name in the DDF Start Message DSNL004I.
- For DB2 for AS/400, this value is same as the local Control Point name from the DSPNETA command.

The example value used in this section is LUDB2G.

MAC address of the DRDA server's token-ring adapter

- For DB2 for OS/390, this value is the LOCADD value of the LINE definition for the 3745 token-ring adapter in the NCP.
- For DB2 for AS/400, thi value is the local adapter address of the AS/400 token-ring adapter from the DSPLIND command.

The example value used in this section is 400009721059.

MAC address of the DataJoiner workstation's token-ring adapter
The example value used in this section is 0800207215f4.

Local LU name for the DataJoiner workstation

- For DB2 for OS/390, this value is the VTAM Independent LU (LOCADDR=0) defined for the DataJoiner workstation in a VTAM Switched Net Major Node definition.
- For DB2 for AS/400, this value is the Remote Location Name for the DataJoiner workstation.

The example value used in this section is STB9255I.

Node ID for the DataJoiner workstation
If the DRDA server is MVS, this value is the IDBLK + IDNUM parameters in the PU definition for the DataJoiner workstation in a VTAM Switched Net Major Node definition.

In the example in this section IDBLK=071 and IDNUM=B8014; therefore, the Node ID is 071B8014.

Modename IBMRDB

- For DB2 for OS/390, IBMRDB must be in a VTAM Modetab at the DRDA server system.
- For DB2 for AS/400, verify IBMRDB using the **DSPMODD IBMRDB** command.

User ID with Password at the DRDA server

The user ID that you chose must have CONNECT, SELECT, and BINDADD privileges at the DRDA server.

Configuring SNA on Your IBM eNetwork Communication Server V5 for AIX

These steps assume that:

- The basic installation of the IBM eNetwork Communication Server V5 for AIX (CS/AIX) is completed.
- DataJoiner is installed.
- The user is logged on as root.
- A Token Ring network is the communications medium.

If the DRDA server is DB2 for AS/400:

- Ensure that the collection NULLID is created.
- Change the CCSID for the DataJoiner user ID to 037:

  ```
  CHGUSRPRF  DataJoiner userid CCSIC (037)
  ```

To configure eNetwork Communication Server for AIX for use by DataJoiner, log on to the system as a user with root authority and use either the **snaadmin** program (found in `/usr/bin/`) or the **xsnaadmin** program (found in `/usr/bin/X11/`). Information about these programs can be found in the system documentation. The following steps describe how to use the **xsnaadmin** program to configure eNetwork Communication Server for AIX.

1. Enter the command **xsnaadmin**. The Node window for the server opens.
2. Define a Node.
   a. Select **Services –> Configure Node Parameters**. The Node Parameters window opens.
   b. For **APPN support**, select **End node**.
   c. In the **SNA addressing** box:
      1) **Control Point Name** should be:
         - First field: your SNA network name
         - Second field: your local LU name for the DataJoiner workstation
      2) In the **Control Point Alias** field: enter your local LU name for the DataJoiner workstation
      3) In the **Node ID** field type your Node ID for the DataJoiner workstation
   d. Click **OK**.
3. Define a port.
   a. Select the Connectivity and Dependent LUs window.

   b. Click **Add**. The Add to Node window opens.

   c. Select **Port**.

   d. In the **Port Using** box select the appropriate port type.

   e. Click **OK**. The Port window for the selected port type opens.

   f. Type a name for the port in the **SNA port name** field. A common value is PRT0.001

   g. Select the **Initially active** check box.

   h. Click **OK**.

4. Define a link station.

   a. In the Connectivity and Dependent LUs window, select the port that you defined in step 3.

   b. Click **Add**. The Add to Node window opens.

   c. Select **Add a link station to port**.

   d. Click **OK**. The Token ring link station window opens.

   e. Type a name for the link station in the **Name** field.

   f. In the **Activation** box select **On Node Start up**.

   g. In the **LU traffic box**, select **Any**.

   h. In the **Independent LU traffic** box:

      - Type the SNA network name in the first field and type the SNA control point name in the second field.
      - For **Remote Note Type**, select **Discover**.

   i. In the **Dependent LU traffic** box specify:

      - **Remote Node role**: host
      - **Local Node ID**: node ID for the DataJoiner workstation.
      - **Remote ID**: leave this blank.

   j. In the **Contact Information** box specify:

      - **MAC Address**: The MAC address of the DRDA server's token ring adapter.
      - **SAP number**: 04 is usually acceptable.

   k. Click **OK**.

5. Define a partner LU over the link station.

   a. Select **Services –> APPC –> New PLUs –> Over Link Station** from the menu bar. The Partner LU on link station window opens.

   b. Fill out the fields:

      - **LU Name**: The local LU name for the DataJoiner workstation.
      - **LS Name**: The name of the Link Station that you defined in step 4e.
      - **Partner LU name**:
        – First field: The SNA network name

    - Second field: The LU name of the DRDA server

  c. Click **OK**.

6. Define an alias for the partner LU.

   a. Select the Remote Sytems window.

   b. Click **Add**. The Add to node window opens.

   c. Select the **Define partner LU alias** radio button.

   d. Click **OK**. The Partner LU window opens.

   e. Fill out the fields:

      - **Partner LU Name**
        - First field: The SNA network name
        - Second field: The LU name of the DRDA nerver
      - **Partner LU characteristics**:
        - **Alias**: Typically the same as Partner LU name (without the network name)
        - **Uninterpreted name**: Typically the same as Partner LU Alias

   f. Select the **Supports Parallel Sessions** check box.

   g. Click **OK**.

7. Define a mode.

   a. Select **Services –> APPC –> Modes** from the menu bar. The Modes window opens.

   b. Click **New**. The Mode window opens.

   c. In the **Name** field type `IBMRDB`.

   d. Specify the configuration values. The following list shows the suggested values for each field:

      - **Initial Session limits**: 32
      - **Maximum Session limits**: 32767
      - **Min contention winner sessions**: 16
      - **Min contention loser sessions**: 16
      - **Auto-activated session**: 0
      - **Initial Receive pacing window**: 16
      - **Maximum Receive pacing window**: 63
      - **Specify timeout**: cleared
      - **Restrict max RU size**: Okay cleared; if selected:
        - Set **Maximum RU size upper bound**: 4096
        - Set **Maximum RU size lower bound**: 1024
        - then clear **Restrict Max RU Size** check box

These values are suggested because they are known to work. You need to tailor these values so that they are optimized for your particular application environment.

   e. Click **OK**.
   f. Click **Done**.

8. Define the CPI-C destination name.
   a. Select **Services –> APPC –> CPI-C** from the menu bar. The CPI-C destination names window opens.
   b. Click **New**. The CPI-C destination window opens.
   c. In the **Name** field type the symbolic destination name that you want to associate with the host or AS/400 server database. This example uses `DB2CPIC`.
   d. In the local LU box select the **Specify Local LU Alias** radio button and type the Node's Control Point alias from step 2c2 on page 86.
   e. In the **Partner LU and mode** box:
      1) Select the **Use PLU Alias** radio button and type the partner LU alias from step 6e on page 88.
      2) In the Mode field type `IBMRDB`.
   f. In the **Partner TP** box:
      - For DB2 for MVS/ESA or DB2 for OS/390, either:
        – Click **Service TP** and type `x'07f6c4c2'` in the field.
        – Click **Application TP** and type `DB2DRDA for DB2 for OS/390 or DB2/MVS` in the field.
      - For AS/400, either:
        – Click **Service TP** and type `x'07f6c4c2'` in the field.
        – Click **Application TP** and type `QCNTEDDM` in the field.
      - For DB2 for VM:
        – Click **Application TP** and type the DB2 for VM database name in the field.
      - For DB2 for VSE:
        – Click **Application TP** and type `AXE` in the field.
   g.

   In the **Security** box, select the security type. This value is typically None.

   Note: The Security method will actually be specified in "Step 2: Update the Node Directory" on page 98 for access within heterogeneous environments and in "Step 2: Update the Node Directory" on page 103 for access outside of hererogeneous environments. In the DB2 CATALOG APPC NODE step, SECURITY PROGRAM is specified if User ID and Password are to be sent and SECURITY SAME is specified if User ID and the Already-Verified Token are to be sent.

   h. Click **OK**.
   i. Click **Done**.

9. Test the APPC connection.
   a. Start the SNA subsystem by using one of these methods:
      - Enter the **/usr/bin/sna –start** command.
      - From smitty, select **Communications Applications and Services –> Communications Server for AIX –> Manage SNA Resources –> Start SNA Resources –> Start SNA**.
      - Enter `/usr/bin/X11/xsnaadmin` and select **Start**.
   b. Start a SNA session to test the APPC definitions. You can use `xsnaadmin` or `smitty`.
      - From `xsnaadmin`:
        1) On the CS/AIX Node panel start the subsystem node. Select the appropriate node icon for the node defined in Step 2 on page 86 and click **Start**.
        2) Start the link station by selecting the link station defined in step 4 on page 87 and click **Start**.
        3) Start the session.
           - – Select the Independent local LU name used in the CPI-C Destination definition from step 8 on page 89 and click **Start**. The activate session for LU window opens.
           - – In the **Use PLU Alias** field, type the alias for the partner LU specified in the CPI-C Destination definition from step 8e1 on page 89.
           - – In the **Mode** field type `IBMRDB`.
           - – In the **Polarity** field specify **Either**.
           - – Click **OK**.

           In the CS/AIX Node window, ’2 sessions’ should appear next to the selected local LU.
      - From smitty, select **Communications Applications and Services –> Communications Server for AIX –> Manage Resources –> Start Resources –> Start a Session**.

> You may also need to contact your database or network administrators to have your local LU names added to the appropriate tables in order to access the host or AS/400 server database.

When you have finished setting up the APPC network communications, continue the configuration process. If you are configuring for access within the heterogeneous environment, go to “Step 2: Update the Node Directory” on page 98. If you are configuring for access outside the heterogeneous environment, go to “Step 2: Update the Node Directory” on page 103.

Configuring SunLink SNA PU 2.1

This section describes how to configure an APPC connection between your DataJoiner for Solaris workstation and a DRDA server. Before you begin, ensure that your workstation has SunLink SNA for Solaris 9.0 or higher installed and the following communications products:

- SunLink PU2.1 9.0 or later
- SunLink P2P LU6.2 9.0 or later
- SunLink P2P CPI-C 9.0 or later

The examples in this section use a token-ring network as the communications medium.

If you need further information to set up your environment, see:

- *DB2 Connectivity Supplement*
- *DRDA Connectivity Guide*
- *SunLink SNA PU 2.1 9.0 Configuration and Administrator’s Manual*
- *SunLink P2P CPI-C 9.0 Programmer’s Manual*

Planning for Configuring Your DRDA Servers

Before you configure the DataJoiner for Solaris workstation, obtain the following information from your host and LAN administrators:

SNA network name

- For DB2 for OS/390, this value is the VTAM NETID.
- For DB2 for AS/400, this value is the local Network ID from the DSPNETA command.

The example value used in this section is USIBMST.

SNA control point name

- For DB2 for OS/390, this value is the SSCP name of the VTAM that loads the NCP software into the 3745 that has the token ring adapter that the DataJoiner workstation needs to communicate with to reach the DRDA server.
- For DB2 for AS/400, this value is the local Control Point (CP Name) from the DSPNETA command.

The example value used in this section is ST11CDRM.

Location or RDB name of the DRDA server

- For DB2 for OS/390, this value is the Location Name on the DDF Start Message DSNL004I.
- For DB2 for AS/400, this value is the WRKRDBDIRE Relational Database name where Remote Location Name is *LOCAL.

The example value used in this section is DB2G.

LU name of the DRDA server

- For DB2 for OS/390, this value is the LU Name in the DDF Start Message DSNL004I.
- For DB2 for AS/400, this value is same as the local Control Point name from the DSPNETA command.

The example value used in this section is LUDB2G.

MAC address of the DRDA server's token-ring adapter

- For DB2 for OS/390, this value is the LOCADD value of the LINE definition for the 3745 token-ring adapter in the NCP.
- For DB2 for AS/400, thi value is the local adapter address of the AS/400 token-ring adapter from the DSPLIND command.

The example value used in this section is 400009721059.

MAC address of the DataJoiner workstation's token-ring adapter

The example value used in this section is 0800207215f4.

Local LU name for the DataJoiner workstation

- For DB2 for OS/390, this value is the VTAM Independent LU (LOCADDR=0) defined for the DataJoiner workstation in a VTAM Switched Net Major Node definition.
- For DB2 for AS/400, this value is the Remote Location Name for the DataJoiner workstation.

The example value used in this section is STB9255I.

Node ID for the DataJoiner workstation

If the DRDA server is MVS, this value is the IDBLK + IDNUM parameters in the PU definition for the DataJoiner workstation in a VTAM Switched Net Major Node definition.

In the example in this section IDBLK=071 and IDNUM=B8014; therefore, the Node ID is 071B8014.

Modename IBMRDB

- For DB2 for OS/390, IBMRDB must be in a VTAM Modetab at the DRDA server system.

- For DB2 for AS/400, verify IBMRDB using the **DSPMODD IBMRDB** command.

User ID with Password at the DRDA server
The user ID that you chose must have CONNECT, SELECT, and BINDADD privileges at the DRDA server.

Configuring SunLink SNA on Your DataJoiner for Solaris Workstation

To configure your DataJoiner for Solaris workstation:

1. As root, configure the SunLink PU 2.1 configuration file `/opt/SUNWpu21/sunpu2.config`.
2. As root, start SunLink SNA with a new or updated configuration file.
3. As the DataJoiner instance owner, create a CPIC Side file in the directory `INSTHOME/sqllib/adm`.
4. If the DRDA server is DB2 for AS/400:
   - Ensure that the collection NULLID is created.
   - Change the CCSID for the DataJoiner user ID to 037:

     ```
     CHGUSRPRF  DataJoiner userid CCSIC (037)
     ```

These steps are described in detail in the following sections.

Configuring the SunLink PU 2.1 Configuration File: You must perform this step as root. For this step, you can edit the sample file that is provided with SunLink SNA. Include the Directives in the SunLink PU 2.1 configuration file:

- CP
- TRLINE
- DLC
- LU
- PTNR_LU for DRDA Server
- MODE

End each Directive definition with a semicolon (;). See the *SunLink SNA PU 2.1 9.0 Configuration and Administrator’s Manual* for more information about defining these directives.

Consider the following recommendations for configuring the SunLink SNA PU 2.1 configuration file for DataJoiner. The Sample values provided correspond to those that are used in Figure 11 on page 96.

CP Directive
Required to activate PU Type 2.1 capabilities.

- NAME= anything up to 8 characters; in the example, `STB9255`.

- NQ_CP_NAME=Network name (NETID).CP name; in the example, USIBMST.STB9255.

TRLINE Directive

Defines the token-ring adapter on the DataJoiner for Solaris workstation.

- NAME=anything; MAC1 is usually used.
- DEVICE=the UNIX path to the device special file for the TR port; in the example, /dev/tr.
- LAN_RATE=4 or 16 megabytes; in the example, RING_16Mbs.
- SOURCE_ADDRESS: the MAC address of the TR adapter; in the example, x'0800207215f4'.

DLC Directive

Defines the link to the DRDA server.

- NAME=anything up to 8 characters; DLC1 is usually used.
- LINK_NAME=Name of TRLINE directive; in the example, MAC1.
- ACT_PU_SUPPRESS=YES
- LCLLSAP=x'04' is usually used.
- RMTLSAP=x'04' is usually used.
- MODULO=8 or 128; in the example, 128.
- RMTMACADDR=the MAC address of DRDA server; in the example, x'400009721059'.
- TERMID=the Node ID of the DataJoiner for Solaris workstation. If the DRDA Server is MVS, use the IDBLK+IDNUM; in the example, x'071B9255'.

LU Directive

Defines the local LU of the DataJoiner for Solaris workstation.

- NAME=Independent LU name supplied by the DRDA Server Network Administrator; in the example, STB9255I.
- NQ_LU_NAME=Network name (NETID).LU Name; in the example, USIBMST.STB9255I.
- SESS_LMT=the maximum number of sessions that are supported. In the example, 16.
- LUTYPE=6.2 required.

PTNR_LU Directive

Defines the LU of the DRDA server (the LU of DB2 for OS/390 or AS/400).

- NAME=the LU Name of the DRDA server; in the example, LUDB2G.
- NQ_LU_NAME=Network name (NETID).LU Name; in the example, USIBMST.LUDB2G.

- LOC_LU_NAME=the name of the LU Directive; in the example, STB9255I.

MODE Directive

Defines DRDA Mode IBMRDB.

- A MODE Directive with NAME=IBMRDB must be included for each DRDA server, using the PTNR_LU_NAME of each.
- NAME=IBMRDB; IBMRDB is required.
- DLC_NAME=the name of the DLC Directive; in the example, DLC1.
- PTNR_LU_NAME=the name of the PTNR_LU Directive, in the example, LUDB2G.
- LCL_MAX_SESS_LMT=the maximum number of sessions; in the example, 30.
- MIN_CW_SESS=the minimum number of contention winners; in the example, 15.
- MIN_CL_SESS=the minimum number of contention losers; in the example, 15.

Figure 11 on page 96 contains an example /opt/SUNPU21.sunpu2.config.STB9255 file.

```
CP NAME=STB9255
NQ_CP_NAME=USIBMST.STB9255
;

TRLINE NAME=MAC1
DEVICE='/dev/tr'
LAN_RATE=RING_16Mbs
SOURCE_ADDRESS=x'0800207215f4'
;

DLC NAME=DLC1
LINK_NAME=MAC1
ACT_PU_SUPPRESS=YES
LCLLSAP=x'04'
RMTLSAP=x'04'
MODUL0=128
RMTMACADDR=x'400009721059'
TERMID=x'071B9255'
;

LU NAME=STB9255I
NQ_LU_NAME=USIBMST.STB9255I
SESS_LMT=16
LUTYPE=6.2
;

PTNR_LU NAME=LUDB2G
NQ_LU_NAME=USIBMST.LUDB2G
LOC_LU_NAME=STB9255I
;
MODE NAME=IBMRDB
DLC_NAME=DLC1
PTNR_LU_NAME=LUDB2G
LCL_MAX_SESS_LMT=30
MIN_CW_SESS=15
MIN_CL_SESS=15
;
```

Figure 11. Example SunLink Configuration File

Starting SunLink SNA: You start SunLink SNA with the new or updated sunpu2.config file. This step must be performed as root.

1. Change to the SunLink PU 2.1 directory:

   ```
   cd /opt/SUNWpu21
   ```

2. Set up environment variables for FlexLM licensing, for example:

   ```
   export LD_LIBRARY_PATH=/usr/openwin/lib:/usr/lib
   export LM_LICENSE_FILE=/etc/opt/licenses/licenses_combined
   ```

   Refer to the SunLink documentation for more information.

3. Check the status of SunLink SNA by using the sunop utility.
4. Stop SunLink if it is active. For example, enter:

```
kill -9 sunpu2.pid
kill -9 sunlu2.pid
```

5. Start SunLink by using the **sunpu2.1** command and the new or updated sunpu2.config file.

   You can start **sunpu2.1** using a file that also sets the required environment variables and specifies a particular sunpu2.config file to be used. In the example, the file name is snastart.STB9255. The contents of the file are:

   ```
   LD_LIBRARY_PATH=$(LD_LIBRARY_PATH):/usr/openwin/lib:/usr/lib
   LM_LICENSE_FILE=/etc/opt/licenses/licenses_combined:$(LM_LICENSE_FILE)
   export LD_LIBRARY_PATH LM_LICENSE_FILE
   cd /opt/SUNWpu21
   ```

   To invoke **sunpu2.1** using the sunpu2config file, enter:

   ```
   /opt/SUUNWpu21/sunpu2.1  -f /opt/SUNWpu21/sunpu2.config.STB9255
   ```

Creating a CPIC Side File: This step must be performed as the DataJoiner instance owner.

Create a CPIC side file in the sqllib/adm directory:

1. Change to the INSTHOME/sqllib/adm directory.
2. Use vi or another editor to create a CPIC side file. The file name must be eight characters or less. It can be either uppercase or lowercase.

   An example CPIC side file (INSTHOME/sqllib/adm/st11db2g) looks like this:

   ```
   PTNR_LU_NAME   = LUDB2G
   MODE_NAME      = IBMRDB
   TP_NAME        = DB2DRDA
   SECURITY       = NONE
   ```

   where:

   - PTNR_LU_NAME=the name of the PTNR_LU Directive for the DRDA server in the sunpu2.config file; in the example, LUDB2G.
   - MODE_NAME=IBMRDB. IBMRDB is a required value.
   - TP_NAME=DB2DRDA is the TP name for the DB2 for OS/390 DRDA server.
   - SECURITY=NONE or PROGRAM if the DRDA server requires passwords; Specify SECURITY=SAME if the DRDA server does not require passwords and is expecting the ″Already Verified″ indicator.

For details on creating CPIC side files, see the *SunLink P2P CPI-C 9.0 Programmer's Manual*.

Important: When updating the DataJoiner Node Directory, specify the CPIC side file name in the CATALOG APPC NODE command as the symbolic destination name after the REMOTE keyword. The value is case-sensitive. For example, if the CPIC side file name is db2cpidj, update the Node Directory by using a command similar to:

```
db2 'CATALOG APPC NODE STMVS11 REMOTE db2cpidj SECURITY PROGRAM'
```

Step 2: Update the Node Directory

The node directory contains entries for some types of remote servers that your workstation can access. Update the node directory by using the command line processor (CLP) or, if using DataJoiner for AIX systems, the Database Director utility. To catalog a node, you must have SYSADM authority.

The following examples show the proper syntax for updating the node directory from the CLP.

For TCP/IP configurations:

```
CATALOG TCPIP NODE mvs1node REMOTE mvs1host SERVER 446
```

For APPC configurations:

```
CATALOG APPC NODE mvs1node REMOTE db2cpidj SECURITY program
```

where:

mvs1node
: Is an arbitrary name for the node being cataloged.

mvs1host and db2cpidj
: For TCP/IP configurations, mvs1host is the host name of the DRDA3 data source. You can also specify the DRDA3 host subsystem's IP address as the REMOTE keyword.

  For APPC configurations, db2cpidj is the LU 6.2 Side Information Profile being used to communicate with the DRDA data source.

446
: Can be either the DRDA port number or the DRDA service name (by default, drda) of the DRDA3 subsystem. The SERVER parameter is used only for TCP/IP configurations.

program
: Controls the security information that gets passed to the DRDA AS over SNA. Valid values include PROGRAM, SAME, and NONE; however, PROGRAM, which passes both a user ID and password to the DRDA AS, is specified in most situations. The SECURITY parameter is used only for APPC configurations. See the *DataJoiner Administration Supplement* for more information about security.

For more information about the **CATALOG...NODE** command, see the *DATABASE 2 Command Reference*.

Step 3: Update DataJoiner Catalog Views with Server and User Information

To access a data source that uses DataJoiner nicknames or pass-through statements, you must update the DataJoiner catalog by using DataJoiner DDL statements. This section explains how to do so by using the following DDL statements:

- CREATE SERVER MAPPING
- CREATE USER MAPPING
- CREATE SERVER OPTION
- CREATE NICKNAME

See the *DataJoiner Application Programming and SQL Reference Supplement* for more detailed information about DataJoiner DDL statements and when to use them.

CREATE SERVER MAPPING

Use the CREATE SERVER MAPPING DDL statement to update the SYSCAT.SERVERS view. This DDL statement is required to identify the DRDA data source to DataJoiner. The following guidelines apply when issuing the CREATE SERVER MAPPING statement:

- Choose a unique server name.
- Set NODE to the node name that you cataloged in “Step 2: Update the Node Directory” on page 98. This value is case-sensitive.
- Set DATABASE to a value appropriate for the server platform:
  - For DB2 for OS/390 data sources, set DATABASE to the LOCATION of the OS/390 system. This value is obtained from the DDF start message (DSNL004I).
  - For DB2 for AS/400 data sources, set DATABASE to the local RDB name of the AS/400 system. This value is obtained from the WRKRDBDIRE display and is the relational database name of the record whose remote location is *LOCAL.

  The DATABASE parameter is case-sensitive.
- Set TYPE to the data source type you are mapping. See the *DataJoiner Application Programming and SQL Reference Supplement* for a list of valid types.
- Set VERSION to the correct version of the data source.
- Set PROTOCOL to ″drda″ if using APPC, or to ″drdaIP″ if using TCP/IP. This value is case-sensitive and needs to be enclosed in quotation marks.

- Set AUTHID to an authorization name at the data source that has BINDADD or equivalent privileges. This value will be transformed to uppercase unless enclosed in quotation marks.
- Set PASSWORD to the password that is associated with the AUTHID. This value will be transformed to uppercase unless enclosed in quotation marks.

  Restrictions:

  - For DB2 for OS/390 data sources, do not specify a password if `TCP/IP Already Verified => YES` was specified in the DDF DSNTIP5 setup panel.
  - For DB2 for AS/400 data sources, do not specify a password if `DDM Password Required = NO` was specified in CHGDDMTCPA.

A sample CREATE SERVER MAPPING statement for a DB2 for OS/390 data source accessed using TCP/IP is:

```
CREATE SERVER MAPPING FROM db2prd1 TO NODE "mvs1node"
DATABASE "db2mvsa" TYPE db2/mvs VERSION 5.1 PROTOCOL "drdaIP"
AUTHID DB2ADM PASSWORD DB2PW
```

A sample CREATE SERVER MAPPING statement for a DB2 for AS/400 data source accessed using TCP/IP is:

```
CREATE SERVER MAPPING FROM db2prd1 TO NODE "4001node"
DATABASE "as400a" TYPE db2/400 VERSION 4.31 PROTOCOL "drdaIP"
AUTHID DB4ADM PASSORD DB4PW
```

A sample statement for a DB2 for OS/390 data source that is accessed using APPC is:

```
CREATE SERVER MAPPING FROM db2mvsa TO NODE "db2node"
DATABASE "db2mvs01" TYPE db2/mvs VERSION 3.1 PROTOCOL "drda"
AUTHID DB2ADM PASSWORD DB2PW
```

Preserving case-sensitivity:

The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

CREATE USER MAPPING

Use the CREATE USER MAPPING DDL statement to update the SYSCAT.REMOTEUSERS view. This DDL statement is used to map information about authorization IDs and passwords that are used to access a data source. The following example shows the proper syntax of the CREATE USER MAPPING DDL statement:

```
CREATE USER MAPPING FROM BRODY TO SERVER db2prd1
AUTHID DB2ADM PASSWORD DB2PW
```

where:

- BRODY is a local DataJoiner authorization name.
- db2prd1 is the name of the server that contains the remote authorization ID.
- DB2ADM is the remote authorization ID.
- DB2PW is the password that is associated with the remote authorization ID.

CREATE SERVER OPTION

Use the CREATE SERVER OPTION DDL statement to update the SYSCAT.SERVER_OPTIONS view. This DDL statement is used to customize data-source server attributes. For additional information on the CREATE SERVER OPTION DDL statement, see the *DataJoiner Application Programming and SQL Reference Supplement*. The following example shows the proper syntax of the CREATE SERVER OPTION DDL statement for the pushdown option.

```
CREATE SERVER OPTION pushdown FOR SERVER db2prd1
SETTING 'Y'
```

where:

- pushdown is the option that controls whether or not DataJoiner will allow remote data sources to evaluate operations locally.
- db2prd1 is the name of the server for which this option is being implemented.
- 'Y' indicates that the option is allowed.

CREATE NICKNAME

Use the CREATE NICKNAME DDL statement to map the server name, remote authorization ID, and table name of a remote data source into a single nickname.

Attention:

> Before creating nicknames, read the considerations and restrictions associated with them as documented in the *DataJoiner Application Programming and SQL Reference Supplement*.

The following example shows the proper syntax of the CREATE NICKNAME DDL statement:

```
CREATE NICKNAME deptx.proj_info FOR db2prd1.BRODY.proj
```

where:

- `deptx` is an optional schema or qualifier. If none is specified, this value defaults to the user ID of the creator.
- `proj_info` is an arbitrary identifier.
- `db2prd1` is the name of the server where the remote view or table is located.
- `BRODY` is the schema to which the view or table belongs.
- `proj` is a remote view or table name.

Attention:

Server name, schema, and table name are case-sensitive parameters. You must enclose all case-sensitive parameters in double quotes.

Step 4: Test the Connections to the Data Source

This step is optional but recommended because it allows you to determine if your data source is configured correctly.

Prerequisite step: Before testing connections, you must perform user mapping as explained in “Step 3: Update DataJoiner Catalog Views with Server and User Information” on page 99.

To test your connections through DataJoiner, complete the following tasks:

1. Connect to the DataJoiner database.
2. Either create a nickname with DataJoiner for a table at a data source, or use the pass-through facility to directly access a table at a data source.

Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through statements.

Accessing DRDA Data Sources using DataJoiner DDCS

Applications that use IBM’s Distributed Database Connection Services (DDCS) to access DRDA data sources can also be used with DataJoiner’s DRDA application requester (AR) functionality. There are two reasons you might want to use this method:

- So applications can connect directly to DRDA databases without using DataJoiner nicknames or pass-through sessions
- As an alternative means of binding the packages that are used by DataJoiner to these data sources (DataJoiner also binds these packages when a CREATE SERVER MAPPING statement is issued for a DRDA data source)

The steps that are required to access a DRDA data source through DataJoiner DDCS functionality are:

- "Step 1: Configure Network Communications" on page 80
- "Step 2: Update the Node Directory"
- "Step 3: Catalog a DCS Directory Entry" on page 104
- "Step 4: Catalog a Database Directory Entry" on page 104
- "Step 5: Bind Packages used by DataJoiner and Other Applications" on page 105

Step 1: Configure Network Communications

See "Configuring IBM eNetwork Communication Server on AIX for DRDA Data Sources" on page 83 or "Configuring TCP/IP for DRDA3 Data Sources (DB2 for OS/390 and DB2 for AS/400)" on page 80 for instructions.

Step 2: Update the Node Directory

The node directory contains entries for some types of remote servers that your workstation can access. Update the node directory by using the command line processor (CLP) or, if using DataJoiner for AIX systems, the Database Director utility. To catalog a node, you must have SYSADM authority.

The following examples show the proper syntax for updating the node directory by using the CLP.

For TCP/IP configurations:

```
CATALOG TCPIP NODE mvs1node REMOTE mvs1host SERVER 446
```

For APPC configurations:

```
CATALOG APPC NODE mvs1node REMOTE db2cpidj SECURITY program
```

where:

`mvs1node`
Is an arbitrary name for the node that is being cataloged.

`mvs1host` and `db2cpidj`
For TCP/IP configurations, `mvs1host` is the host name of the DRDA3 data source. You can also specify the DRDA3 host subsystem's IP address as the REMOTE keyword.

For APPC configurations, `db2cpidj` is the LU 6.2 Side Information Profile being used to communicate with the DRDA data source.

`446` Can be either the DRDA port number or the DRDA service name (by default, drda) of the DRDA3 subsystem. The SERVER parameter is used only for TCP/IP configurations.

program

Controls the security information that gets passed to the DRDA AS over SNA. Valid values include PROGRAM, SAME, and NONE; however, PROGRAM, which passes both a user ID and password to the DRDA AS, is specified in most situations. The SECURITY parameter is used only for APPC configurations. See the *DataJoiner Administration Supplement* for more information about security.

For more information about the **CATALOG...NODE** command, see the *DATABASE 2 Command Reference*.

Step 3: Catalog a DCS Directory Entry

Catalog an entry in the DCS directory for the database; for example:

```
CATALOG DCS DATABASE db2db AS db2mvsa
```

where:

- db2db is the alias of the target database being cataloged.
- db2mvsa is the target database name:
  - For DB2 for OS/390 data sources, set the AS keyword to the LOCATION of the OS/390 system. This value can be obtained from the DDF start message (DSNL004I).
  - For DB2 for AS/400 data sources, the AS keyword value can be obtained from the WRKRDBDIRE display; it is the relational database name of the record whose remote location is *LOCAL.

For more information about the **CATALOG DCS DATABASE** command, see the *DATABASE 2 Command Reference*.

Step 4: Catalog a Database Directory Entry

Update the system database directory; for example:

```
CATALOG DATABASE db2db AT NODE mvs1node AUTHENTICATION SERVER
```

where:

- db2db is the database name that was specified in the **CATALOG DCS DATABASE** command in the previous step.
- mvs1node is the name of the node from “Step 2: Update the Node Directory” on page 103.

For more information about the **CATALOG DATABASE** command, see the *DATABASE 2 Command Reference*.

Step 5: Bind Packages used by DataJoiner and Other Applications

To simplify the bind procedure, bind files are grouped together in different .lst files, each one being specific to a type of DRDA server. You must first connect to the database, then issue the **bind** command against the database using the appropriate .lst file.

To bind the necessary packages for the DB2 Client Application Enabler for AIX products in use at the DRDA server, issue the following commands:

```
CONNECT TO dbname USER userid USING password
BIND path/@DRDA_server.lst BLOCKING ALL
     SQLERROR CONTINUE MESSAGES bind.msg GRANT PUBLIC
CONNECT RESET
```

where *dbname* is the database name; the *userid/password* pair are used for authentication; DRDA_server.lst is the appropriate .lst file for a specific DRDA server, as shown in the following list:

DRDA server	**DRDA_server.lst**
OS/390	ddcsmvs.lst
VSE	ddcsvse.lst
VM	ddcsvm.lst
OS/400	ddcs400.lst

The bind files and lists are located in the home/sqllib/bnd directory. The following example shows a typical command sequence for binding packages at a DB2 for OS/390 server:

```
db2 connect to db2mvsa
db2 bind HOME/sqllib/bnd/@ddcsmvs.lst blocking all grant public
db2 bind HOME/sqllib/bnd/@applycs.lst isolation cs blocking all grant public
db2 bind HOME/sqllib/bnd/@applyur.lst isolation ur blocking all grant public
```

You will also need to bind your clients to the DataJoiner database. Use the db2ubind.lst and the db2cli.lst files. The files are contained in /usr/lpp/djx_02_01_01/bnd on DataJoiner for AIX systems and /opt/IBMdjx/V2.1.1/bnd on DataJoiner for Solaris systems:

```
CONNECT TO dbname USER userid USING password
BIND path/@db2ubind.lst MESSAGES bind.msg GRANT PUBLIC
BIND path/@db2cli.lst MESSAGES clibind.msg GRANT PUBLIC
```

where:

- *dbname* is the name of the DataJoiner database to which you wish to connect
- *userid/password* are used for authentication and must have appropriate privileges to bind packages

- *path* is the full path name of the directory where the bind files are located, such as `/sqllib/bnd/`
- `bind.msg` and `clibind.msg` are the output message files
- **GRANT PUBLIC** grants EXECUTE and BIND privileges to PUBLIC.

Notes about binding:

- You must run the **BIND** command separately for each database that you wish to access in order to put the necessary packages in place in that database.
- The `db2ubind.lst` file contains the list of bind (.bnd) files that are required to create the packages for the database utilities. The `db2cli.lst` file contains the list of bind (.bnd) files that are required to create packages for the DB2 CLI and the DB2 ODBC driver.
- Binding might take a few minutes to complete.

If you configured the DRDA AR just to bind DataJoiner packages to the DRDA database, then the DRDA data source can be uncataloged after you finish binding. However, the DRDA database will need to remain in the DataJoiner database directory if the DRDA database will be accessed directly by applications such as DataJoiner Replication and DataJoiner Replication Administration. For more information about binding DRDA data sources, see *DB2 Installing and Using AIX Clients* and the *IBM DDCS User's Guide*.

Chapter 9. Accessing DB2RA Data Sources

DB2RA refers to the DB2 format and protocol between clients and servers. It is proprietary and equivalent to Oracle's SQL*Net and Sybase's Open Client/Open Server protocols, which are also proprietary. DataJoiner uses DB2RA natively to communicate between clients and DataJoiner, and between DataJoiner and DB2/CS and DB2 Universal Database data sources.

There are two ways to access DB2RA data sources:

- DataJoiner provides access through the use of nicknames and the pass-through facility. This method is explained in "Accessing DB2RA Data Sources Using Nicknames and Pass-Through Statements".
- DataJoiner provides direct access to DB2RA data sources without using nicknames or pass-through sessions. This method is explained in "Accessing DB2RA Data Sources Outside the Heterogeneous Environment" on page 118.

Supported DB2RA data sources must have client support capabilities installed. See the data source documentation for information about configuring the data source.

Accessing DB2RA Data Sources Using Nicknames and Pass-Through Statements

The configuration steps that are required to access a DB2RA data source through the use of nicknames and the pass-through facility are:

1. "Step 1: Configure Network Communications"
2. "Step 2: Update the NODE Directory" on page 114
3. "Step 3: Update DataJoiner Catalog Views with Server and User Information" on page 115
4. "Step 4: Test the Connections to the Data Source" on page 118

Step 1: Configure Network Communications

DataJoiner supports TCP/IP, IPX/SPX, and APPC. Configuration of DataJoiner to any of these is similar to configuring a client to DataJoiner. Configuration instructions for these protocols are provided in the following sections:

- TCP/IP; see "Configuring TCP/IP for DB2RA Data Sources" on page 108

- IPX/SPX; see “Configuring IPX/SPX for DB2RA Data Sources” on page 111

Configuring TCP/IP for DB2RA Data Sources

This section shows how to use TCP/IP to establish a connection from DataJoiner to a DB2/CS or DB2 UDB data source. This section can be used as a guide for configuring any DB2RA data source that is accessed through TCP/IP. The process described here is similar to configuring a client to communicate with DataJoiner. The difference is that here DataJoiner is a client to DB2, instead of a CAE or SDK being a client to DataJoiner.

Before you can configure DataJoiner to access a DB2RA data source through TCP/IP, make sure that TCP/IP support is installed and configured for each participating DB2RA data source. See the data source’s installation guide for instructions.

Due to the characteristics of the TCP/IP protocol, the TCP/IP subsystem on one host might not be notified of the failure of its partner on a different host. As a result, a DataJoiner server accessing a data source using TCP/IP, or the corresponding process at the data source, might sometimes appear to be hung. You can use two network options, tcp_keepidle and tcp_keepintvl, to tune a specified time interval before the failure is detected.

The values for these options are system wide and apply to all TCP/IP socket applications for which the SO_KEEPALIVE socket option is enabled, not just DataJoiner.

To set or display the network options, use the **no** command.

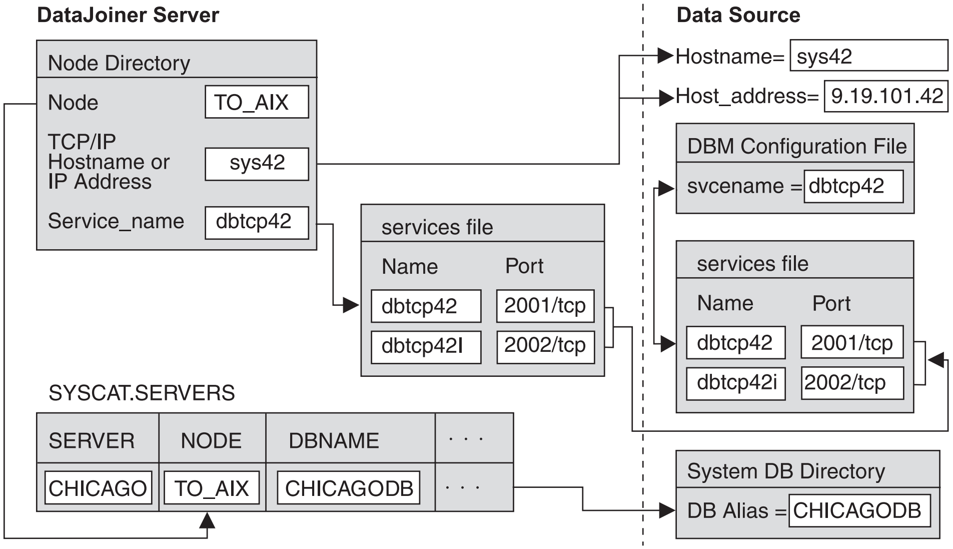


Figure 12. DB2RA Data Source through TCP/IP

The values that are used in this section are in Figure 12. A blank template is also provided in Figure 13 on page 110. You can fill in the blank template with your system's information. Then complete the instructions in this section by plugging your own values in where appropriate.

1. Determine the following values for the DB2RA system:
   - The database name or alias that DataJoiner will access.
   - The TCP/IP port numbers for the data source as they are stored in the `/etc/services` file on the data source's system. In the example, the entries are:

     ```
     dbtcp42     3700/tcp         # Data source primary connect port
     dbtcp42i    3701/tcp         # Data source interrupt port
     ```

   - The host name of the system where the data source resides. To get this name, issue the **hostname** command from a command prompt on the data source's system. In the example, the host name is sys42.
2. Complete the following steps on the DataJoiner server workstation:
   a. Define the data source's port numbers at the DataJoiner server by either manually updating DataJoiner's `/etc/services` file, or by using the instructions in "Step 2: Update the NODE Directory" on page 114. All DB2RA data sources communicate with TCP/IP clients through ports that are defined in `/etc/services`. If DataJoiner is installed on the same system as the data source, the ports are probably already defined as needed. If DataJoiner is installed on a different system than the data source, use the data source's port numbers that are defined in its `/etc/services` file.

b. For DataJoiner for AIX systems V3.2, synchronize the /etc/services file and the inetd daemon by issuing the following AIX commands:

```
inetimp
refresh -s inetd
```

c. If necessary, update /etc/hosts for the data source host name and IP address. This step must be performed as root.

   Whether you must update /etc/hosts depends on how TCP/IP address translation is done on your network. The remote host name that is specified on the **CATALOG TCPIP NODE** command (sys42 in the example) must be translated to an IP address when it is used. If your network has a name server that recognizes the host name, the name server will perform the IP address translation, and you do not need to update /etc/hosts. Otherwise, you need an entry in /etc/hosts is for the remote host name. See your network administrator to learn how your network is configured.

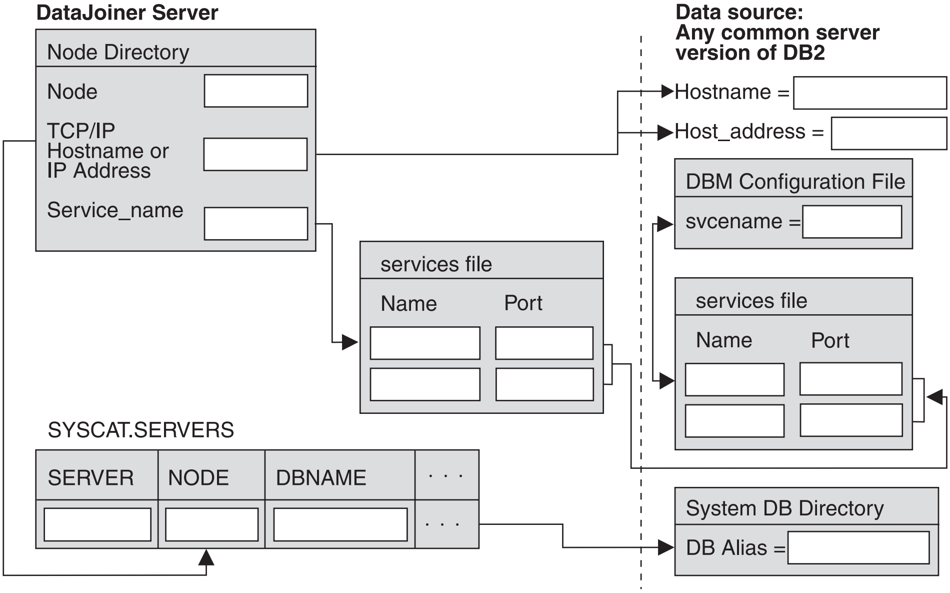


Figure 13. DB2RA Data Source Planning Template for TCP/IP

When you finish setting up TCP/IP network communications, continue the configuration process. If you are configuring for access through the use of nicknames and pass-through statements, go to “Step 2: Update the NODE Directory” on page 114. If you are configuring for access outside the heterogeneous environment, go to “Step 2: Update the NODE Directory” on page 119.

Configuring IPX/SPX for DB2RA Data Sources

DataJoiner provides IPX/SPX communication support through the IPX/SPX support option (`db2_02_01.cs.ipx` on DataJoiner for AIX systems).

To enable IPX/SPX support for a DataJoiner server you must:

- Participate in a Novell NetWare LAN environment.
- Update the database manager configuration file.
- Register the DataJoiner server at the NetWare file server (file server addressing only).
- Set the DB2COMM environment variable.
- Start the database manager (**db2start**).

Configuring IPX/SPX for DataJoiner

IPX/SPX support is provided by the following products:

- IBM Netware for AIX Version 3.11B (this is the complete product which includes a file server)
- IBM AIX NetBIOS and IPX Support/6000

NetWare 4.x Directory Services is not supported in this release of DataJoiner. You should use the bindery emulation feature of NetWare 4.x.

You can access DataJoiner through IPX/SPX as follows:

- File Server Addressing

  The client connects to DataJoiner by retrieving the server's address from a NetWare file server.
- Direct Addressing

  The client connects to DataJoiner by directly specifying the NetWare internetwork address of the server (bypassing the NetWare file server). No NetWare file server needs to reside on the network with the client and server.

The NetWare information you need to update the database manager configuration file is as follows:

Fileserver Specifies the name of the NetWare file server where the internetwork address of the database manager will be registered.

The internetwork address of the database manager is stored in the bindery at the NetWare file server.

Objectname Represents a particular database manager in the network, and should be unique for each DataJoiner instance that is registered at a NetWare file server.

Socket number

Specifies a well-known static socket number and represents the connection endpoint in a DataJoiner server's internetwork address. It must be entered in the database manager configuration file in hexadecimal.

The socket number defaults to 0x879E (change this value if running more than 1 DataJoiner instance). DataJoiner has registered well-known sockets with Novell in the range 0x879E to 0x87A2 for the customer's use. If you run more than 5 instances on the server machine, you must prevent socket collisions for instances 6 and higher by choosing a socket number that is *not* 0x0000 or in the dynamic socket range 0x4000 to 0x7FFF.

The number 0x0000 is reserved for allocating dynamic sockets, and the range of numbers from 0x4000 to 0x7FFF is used for dynamic sockets.

The socket number (*ipx_socket*) must be unique for every DataJoiner instance on a machine. It must also be unique among all IPX/SPX applications running on the DataJoiner machine to ensure that DataJoiner is able to listen to incoming IPX/SPX connections using this socket number.

Updating the Database Manager Configuration File for IPX/SPX

As described above, the database manager configuration file at the server contains the following parameters that are specific for IPX/SPX support:

- fileserver
- objectname
- ipx_socket

File Server Addressing: Configure this type of addressing if the DataJoiner server will be accessed by clients that use either of the following addressing schemes:

- File server addressing and direct addressing
- File server addressing only

Note: For file server addressing, the following characters are not valid for the fileserver parameters or the objectname parameters: `/ \ : ; , * ?`

These configuration parameters can be entered or modified in the database manager configuration file by using one of the following choices:

- The Command Line Processor, documented in the *DB2 Command Reference*

  For example:

  ```
  db2 update database manager configuration using fileserver netwsrv
  db2 update database manager configuration using objectname db2inst1
  db2 update database manager configuration using ipx_socket 879F
  ```
- The Configuration API, documented in the *DB2 API Reference*
- The DB2 Database Director

Direct Addressing: Configure this type of addressing if DataJoiner will be accessed by clients that use only direct addressing.

If you are using direct addressing only, you might specify the fileserver and object name parameters as '*'. For example:

```
  db2 update database manager configuration using fileserver * objectname *
```

Note: To make this change to the configuration file effective, you must stop and start the database manager.

Registering the DB2 Server

It is not necessary to register the DataJoiner server if you use direct addressing only.

After the IPX/SPX parameters have been configured, the administrator must register the DataJoiner server in the bindery at the NetWare file server before starting the database manager. To register the DataJoiner server, issue the following command:

```
db2 register db2 server in nwbindery user <userid> password <password>
```

where <userid> and <password> are your Novell file server login user ID and password which must have Supervisor or Workgroup Manager security equivalence.

Notes:

1. This command must be issued locally from a DataJoiner server (it is not supported remotely) *once* before the first **db2start** for each DataJoiner instance.
2. If you move DataJoiner on your network, change its address, or if other IPX/SPX configuration changes are necessary, you must first deregister the DataJoiner server in the bindery at the NetWare file server, make the change, then re-register the DataJoiner server.

When registering at a NetWare 4.x file server (which uses directory services and provides bindery emulation capability), the userid used to register/deregister must be created with the same context as the ″current bindery context used by Directory Services when it does bindery emulation″ (as well as having supervisory or equivalent authority).

You can find the bindery emulation context currently in use by checking the bindery emulation setting on the NetWare 4.x file server (with the SERVMAN utility).

Setting DB2COMM for IPX/SPX

You must set a value of `ipxspx` in the DB2COMM environment variable to enable IPX/SPX support when a **db2start** command is issued.

When you have finished setting up IPX/SPX network communications, continue the configuration process. If you are configuring for access within the heterogeneous environment, go to “Step 2: Update the NODE Directory”. If you are configuring for access outside the heterogeneous environment, go to “Step 2: Update the NODE Directory” on page 119.

Step 2: Update the NODE Directory

The NODE directory contains entries for some types of remote servers that your workstation can access. Update the NODE directory by using the command line processor (CLP) or, if using DataJoiner for AIX systems, the Database Director utility. To catalog a node, you must have SYSADM authority.

The following examples show the proper syntax for updating the NODE directory from the CLP.

For TCP/IP configurations:

```
CATALOG TCPIP NODE db2node REMOTE sys42 SERVER dbtcp42
```

For APPC configurations:

```
CATALOG APPC NODE db2node REMOTE db2cpidj SECURITY program
```

where:

`db2node`
    Is an arbitrary name for the node being cataloged.

`sys42` and `db2cpidj`
    For TCP/IP configurations, `sys42` is the host name of the DRDA3 data source. You can also specify the DB2RA system’s IP address as the REMOTE keyword.

For APPC configurations, `db2cpidj` is the LU 6.2 Side Information Profile being used to communicate with the DB2RA data source.

`dbtcp42`
: Can be either the port number or the service name of the DB2RA system. The SERVER parameter is used only for TCP/IP configurations.

`program`
: Controls the security information that gets passed to the DB2RA system over SNA. Valid values include PROGRAM, SAME, and NONE; however, PROGRAM, which passes both a user ID and password to the DB2RA system, is specified in most situations. The SECURITY parameter is used only for APPC configurations. See the *DataJoiner Administration Supplement* for more information about security.

For more information about the **CATALOG...NODE** command, see the *DATABASE 2 Command Reference*.

Step 3: Update DataJoiner Catalog Views with Server and User Information

To access a DB2RA data source using DataJoiner nicknames or pass-through statements, you must update the DataJoiner catalog by using DataJoiner DDL statements. This section explains how to do so by using the following DDL statements:

- CREATE SERVER MAPPING
- CREATE USER MAPPING
- CREATE SERVER OPTION
- CREATE NICKNAME

See the *DataJoiner Application Programming and SQL Reference Supplement* for more detailed information about DataJoiner DDL statements and when to use them.

CREATE SERVER MAPPING

Use the CREATE SERVER MAPPING DDL statement to update the SYSCAT.SERVERS view. This DDL statement is required to identify the DB2RA data source to DataJoiner. The following guidelines apply when issuing the CREATE SERVER MAPPING statement:

- Choose a unique server name.
- Set NODE to the node you cataloged in “Step 2: Update the NODE Directory” on page 114. This value is case-sensitive.

- Set DATABASE to the name of a database at the server that you are mapping. This value is case-sensitive.
- Set TYPE to the data source type you are mapping. See the *DataJoiner Application Programming and SQL Reference Supplement* for a list of valid types.
- Set VERSION to the correct version of the data source.
- Set PROTOCOL to ″db2ra″. This value is case-sensitive and must be enclosed in delimiters.
- Set AUTHID to an authorization name at the data source that has SYSADM or equivalent authority. This value will be transformed to uppercase unless enclosed in quotation marks.
- Set PASSWORD to the password that is associated with the AUTHID. This value will be transformed to uppercase unless enclosed in quotation marks.

A sample CREATE SERVER MAPPING statement for a DB2/CS data source is:

```
CREATE SERVER MAPPING FROM db2v2cs1 TO NODE "db2node"
DATABASE "invoices" TYPE DB2/CS VERSION 2.1 PROTOCOL "db2ra"
AUTHID DB2ADM PASSWORD DB2PW
```

Preserving case-sensitivity:

The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

CREATE USER MAPPING

Use the CREATE USER MAPPING DDL statement to update the SYSCAT.REMOTEUSERS view. This DDL statement is used to map information about authorization IDs and passwords that are used to access a data source. The following example shows the proper syntax of the CREATE USER MAPPING DDL statement:

```
CREATE USER MAPPING FROM BRODY TO SERVER db2v2cs1
AUTHID DB2ADM PASSWORD DB2PW
```

where:

- BRODY is a local DataJoiner authorization name.
- db2v2cs1 is the name of the server that contains the remote authorization ID, which, in this example, is DB2ADM.

- DB2PW is the password that is associated with the remote authorization ID.

CREATE SERVER OPTION

Use the CREATE SERVER OPTION DDL statement to update the SYSCAT.SERVER_OPTIONS view. This DDL statement is used to customize your DataJoiner configuration. The following example shows the proper syntax of the CREATE SERVER OPTION DDL statement for the pushdown option.

```
CREATE SERVER OPTION pushdown FOR SERVER db2v2cs1
SETTING 'Y'
```

where:

- pushdown is the option that controls whether or not DataJoiner will allow remote data sources to evaluate operations locally.
- db2v2cs1 is the name of the server for which this option is being implemented.
- 'Y' indicates that the option is allowed.

CREATE NICKNAME

Use the CREATE NICKNAME DDL statement to map the server name, remote authorization ID, and table name of a remote data source into a single nickname.

Attention:

> Before creating nicknames, you should understand the considerations and restrictions associated with them as documented in the *DataJoiner Application Programming and SQL Reference Supplement*.

The following example shows the proper syntax of the CREATE NICKNAME DDL statement:

```
CREATE NICKNAME deptx.proj_info FOR db2v2cs1.BRODY.proj
```

where:

- deptx is an optional schema or qualifier. If none is specified, this value defaults to the user ID of the creator.
- proj_info is an arbitrary identifier.
- db2v2cs1 is the name of the server where the remote view or table is located.
- BRODY is the schema to which the view or table belongs.
- proj is a remote view or table name.

Step 4: Test the Connections to the Data Source

This step is optional but recommended because it allows you to determine if your data source is configured correctly.

Prerequisite step: Before testing, you must perform user mapping as explained in "Step 3: Update DataJoiner Catalog Views with Server and User Information" on page 115.

To test your connections through DataJoiner, complete the following tasks:

1. Connect to the DataJoiner database.
2. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through statements.

Accessing DB2RA Data Sources Outside the Heterogeneous Environment

You might want to configure DataJoiner to access DB2RA data sources outside the heterogeneous environment for two reasons:

- So applications such as Replication and Replication Administration can connect directly to DB2RA databases without using DataJoiner nicknames or pass-through sessions.
- As an alternative means of binding the packages used by DataJoiner to these data sources (DataJoiner also binds these packages when a CREATE SERVER MAPPING statement is issued for a DB2RA data source).

The steps to configure DataJoiner to access a DB2RA data source without using nicknames or pass-through sessions are:

1. Configure network communications
2. Update the NODE directory
3. Catalog a database
4. Bind packages that are used by DataJoiner and other applications

Step 1: Configure Network Communications

You can use the following communications protocols to access DB2RA data sources:

- TCP/IP; see "Configuring TCP/IP for DB2RA Data Sources" on page 108
- IPX/SPX; see "Configuring IPX/SPX for DB2RA Data Sources" on page 111

Step 2: Update the NODE Directory

The NODE directory contains entries for some types of remote servers that your workstation can access. Update the NODE directory by using the command line processor (CLP) or, if using DataJoiner for AIX systems, the Database Director utility. To catalog a node, you must have SYSADM authority.

The following example shows the proper syntax for updating the NODE directory from the CLP. Use the first command for TCP/IP configurations; use the second command for APPC configurations. For example:

```
CATALOG TCPIP NODE db2node REMOTE sys42 SERVER dbtcp42
```

or

```
CATALOG APPC NODE db2node REMOTE db2cpidj SECURITY program
```

where:

`db2node`
: Is an arbitrary name for the node being cataloged.

`sys42` and `db2cpidj`
: For TCP/IP configurations, sys42 is the host name of the DRDA3 data source. You can also specify the DB2RA system's IP address as the REMOTE keyword.

  For APPC configurations, db2cpidj is the LU 6.2 Side Information Profile being used to communicate with the DB2RA data source.

`dbtcp42`
: Can be either the port number or the service name of the DB2RA system. The SERVER parameter is used only for TCP/IP configurations.

`program`
: Controls the security information that gets passed to the DB2RA system over SNA. Valid values include PROGRAM, SAME, and NONE; however, PROGRAM, which passes both a user ID and password to the DB2RA system, is specified in most situations. The SECURITY parameter is used only for APPC configurations. See the *DataJoiner Administration Supplement* for more information about security.

For more information about the **CATALOG...NODE** command, see the *DATABASE 2 Command Reference*.

Step 3: Catalog a Database Directory Entry

Catalog a database directory entry for the database; for example:

```
CATALOG DATABASE db2db AT NODE db2node AUTHENTICATION server
```

where:

- db2db is the name of the database entry in the system database directory.
- db2node is the name of the node from "Step 2: Update the NODE Directory" on page 119.

For a complete discussion of the **CATALOG DATABASE** command, see the *DATABASE 2 Command Reference*.

Step 4: Bind Packages used by DataJoiner and Other Applications

DataJoiner requires certain packages to be bound at the data source.

To simplify the bind procedure, bind files are grouped together in different .lst files, each one being specific to a type of DB2RA server. You must first connect to the database, then issue the **bind** command against the database using the appropriate .lst file. When binding against a DB2 for OS/2, DB2 for AIX, DB2 for HP-UX, or DB2 Solaris server, use the db2ubind.lst and the db2cli.lst files, which are contained in /usr/lpp/djx_02_01_01/bnd. Issue the following commands:

```
CONNECT TO dbname USER userid USING password
BIND <path>/@db2ubind.lst MESSAGES bind.msg GRANT PUBLIC
BIND <path>/@db2cli.lst MESSAGES clibind.msg GRANT PUBLIC
```

where:

- *dbname* is the name of the database to which you want to connect
- *userid/password* are used for authentication
- *<path>* is the full path name of the directory where the bind files are located, such as /sqllib/bnd/
- bind.msg and clibind.msg are the output message files
- **GRANT PUBLIC** grants EXECUTE and BIND privileges to all users.

The following example shows a typical command sequence for binding packages at a DB2/UDB database:

```
connect to db2udba
bind HOME/sqllib/bnd/@db2cli.lst blocking all grant public
bind HOME/sqllib/bnd/@db2ubind.lst blocking all grant public
bind HOME/sqllib/bnd/@applycs.lst isolation cs blocking all grant public
bind HOME/sqllib/bnd/applyur.lst isolation ur blocking all grant public
```

Notes about binding:

- The **bind** command must be run separately for each database that you wish to access.
- The `db2ubind.lst` file contains the list of bind (.bnd) files that are required to create the packages for the database utilities. The `db2cli.lst` file contains the list of bind (.bnd) files that are required to create packages for the DB2 CLI and the DB2 ODBC driver.
- Binding might take a few minutes to complete.

When the bind is finished, you can uncatalog the database.

Chapter 10. Accessing Informix Data Sources

For DataJoiner to access Informix databases, it needs a data access module built by link-editing DataJoiner libraries with Informix libraries. To make this job easier, DataJoiner provides a sample makefile, `djxlink.makefile`, that contains examples of link-editing DataJoiner with Informix.

Note: DataJoiner for Solaris systems supports Informix OnLine V7.1 or later (non-GLS, -XPS, and -SE versions only) using the informix7 data access module. Informix V5 or V6 data sources are not supported.

Two types of Informix data access modules can be built with djxlink.makefile: informix and informix7. Use the informix data access module to access Informix databases that use Informix-Net Version 5. Use the informix7 data access module to access Informix databases that use informix Version 7.

Restriction:

You can access both an Informix Version 5 and an Informix Version 7 database through the Informix-Net Version 5 data access module. However, accessing a Version 5 database through the Version 7 data access module is not supported.

The following two sections assume that the Informix server is installed on a different machine than DataJoiner and the Informix client code. If the Informix server and DataJoiner are installed on the same machine, follow the same instructions but do not set the SQLRM environment variable.

Requirements for Informix transaction logging:

To maintain transaction integrity, DataJoiner requires that transaction logging be enabled for your Informix sources. If transaction logging is disabled, DataJoiner is unable to rollback units of work.

Using the Informix-Net Version 5 Data Access Module

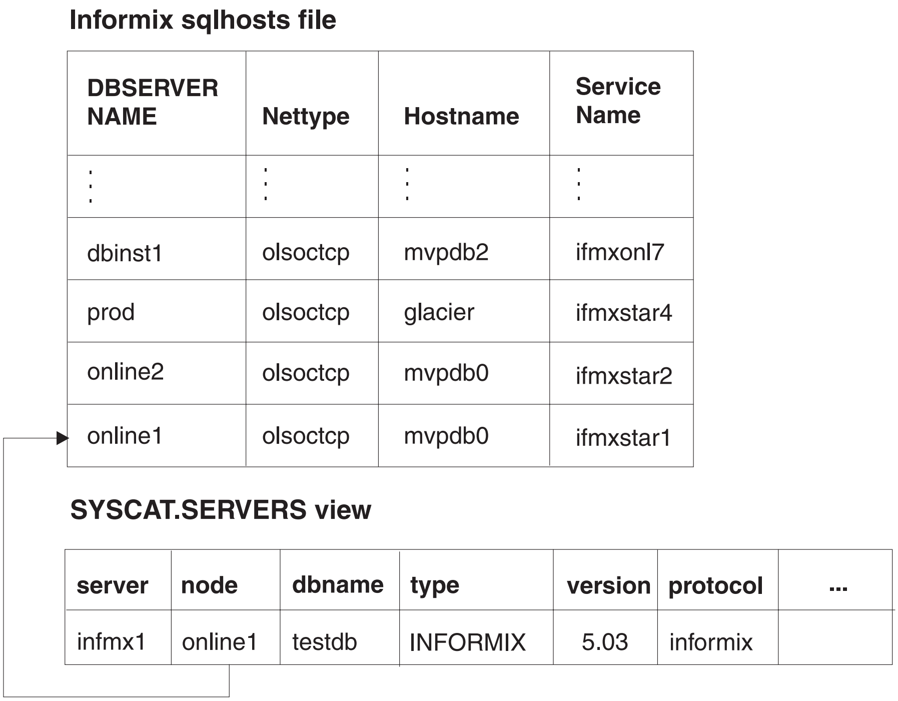
Follow these steps if you are using Informix-Net Version 5 to access Informix databases.

Preqrequisite step:

With Informix Version 5, either the .netrc file or the `/etc/hosts` and `/etc/hosts.equiv` files need to be set up before a client can successfully request a connection to a remote server.

These steps provide examples for using the .netrc file. If you want to use the `/etc/hosts` and `/etc/hosts.equiv` files, see the *Informix-Net and Informix-Star Installation and Configuration Guide.*

1. If you have not already done so, install Informix-Net Version 5 and the Informix esql/c libraries on the system where DataJoiner is installed.
2. Follow the directions in the *Informix-Net/Informix-Star Installation and Configuration Guide* for creating a sqlhosts file under the `$INFORMIXDIR/etc` directory. Figure 14 shows the general structure of an `sqlhosts` file.



Informix sqlhosts file

DBSERVER NAME	Nettype	Hostname	Service Name
⋮	⋮	⋮	⋮
dbinst1	olsoctcp	mvpdb2	ifmxonl7
prod	olsoctcp	glacier	ifmxstar4
online2	olsoctcp	mvpdb0	ifmxstar2
online1	olsoctcp	mvpdb0	ifmxstar1

SYSCAT.SERVERS view

server	node	dbname	type	version	protocol	...
infmx1	online1	testdb	INFORMIX	5.03	informix	

Figure 14. Informix V5 Data Access Module

3. If you have not already done so, build an Informix version 5 data access module by running `djxlink.sh`, as described in “Step 3: Link DataJoiner to Data Source Client Software” on page 46.

   For DataJoiner to send requests to Informix, it needs a data access module (a file) that has been built by link-editing DataJoiner libraries with Informix libraries. If you are accessing multiple Informix databases using the Informix-Net Version 5 data access module, each database that you access requires a data access module.

   When you run `djxlink.sh`, if you receive a message that indicates that the Informix 5 data access module could not be built, you will need to edit the sample makefile so that the makefile works with your particular setup. The sample makefile is /usr/lpp/djx_02_01_01/lib/djxlink.makefile. Editing and running djxlink.makefile are also described in “Step 3: Link DataJoiner to Data Source Client Software” on page 46.

If djxlink.sh fails to build the Informix version 5 data access module, login as root and edit djxlink.makefile. Find the section in the file that is titled “Accessing Data Sources Through the Informix Data Access Module (Informix Version 5)”.

a. In the makefile, respecify the DJX_INFORMIX_LIBRARY_PATH variable to be the directory specified in step 1 on page 124. This directory should be the location of the libraries for your Informix Version 5 driver.

b. The name of the Informix 5 database access module that is built by the makefile is informix. If you do not want to change this name, skip to the next step.

   If you want to change the name, you need to change two lines in the makefile:

   - Change the target name on the dependency line. For example, if you want the data access module to be named ifmx5, change the dependency line from this:

     ```
     informix: libinformix.a informixEP.o informix.exp informix.imp
     ```

     To this:

     ```
     ifmx5: libinformix.a informixEP.o informix.exp informix.imp
     ```

   - Change the output name of the link-edit statement. For example, change this:

     ```
     ld -o informix \
     ```

     To this:

     ```
     ld -o ifmx5 \
     ```

c. As root, link-edit the informix data access module library by using the supplied makefile. To complete this task, you need the following:

   - The informix data access module library that is provided with DataJoiner
   - The informix.exp file that is provided with DataJoiner
   - The Informix esql/c libraries that are provided with the Informix esql product

   Run the makefile by entering:

   ```
   cd /usr/lpp/djx_02_01_01/lib
   make -f djxlink.makefile informix
   ```

If you changed the name of the data access module in the makefile, then use that name with the **make** command. For example:

```
make -f djxlink.makefile ifmx5
```

For AIX 3.2.5 users:

If you are running AIX 3.2.5, and you are unable to create the data access module because of errors from unresolved symbols (mulh, divss and divus), then add the following line to your makefile in the Informix 5 data access module section:

```
-bI:/usr/lib/FCM/lowsys.exp
```

Run the makefile again.

Whenever you rerun `djxlink.makefile` for the informix data access module, you must stop and restart all DataJoiner instances that use the informix data access module.

4. For each of the remaining Informix databases you wish to access through Informix-Net Version 5, copy the newly created data access module to a new file (where each file has a unique name). The unique file name is the value you will specify in the PROTOCOL column of SYSCAT.SERVERS for that Informix database. To copy the data access module from the `/usr/lpp/djx_02_01_01/lib` directory, enter:

   ```
   cp original-data-access-module-name new-data-access-module-name
   ```

   Where *original-data-access-module-name* is the unique name from step 125, and *new-data-access-module-name* is the name of the new data access module. For example:

   ```
   cp informix informixb
   ```

   Do steps 5 through 10 on page 128 for each DataJoiner instance that will use the informix data access module.

5. Update the `.profile` file of the DataJoiner instance to include the following lines:

   ```
   export INFORMIXDIR="informix-home-directory"
   export SQLRMDIR="relay-module-directory"
   export SQLRM="relay-module-name"
   ```

   where:

   informix-home-directory
   Is the directory in which you installed the e/sqlc library in step 1

   relay-module-directory
   Is the directory containing the relay module, typically `informix-home-directory/lib`

relay-module-name

Is the name of the relay module to use for the Informix-Net communication protocol.

Restrictions:

- Be sure there are no spaces on either side of the equal sign (=) in these export statements.
- If DataJoiner and the Informix server are installed on the same machine, do not set the SQLRM environment variable.

See the *Informix-Net/Informix-Star Installation and Configuration Guide* for more information on these environment variables.

6. Use the CREATE SERVER MAPPING DDL statement to define each Informix database you expect to access via Informix-Net Version 5. Specify a different protocol for each database. When performing server mapping:

- Choose a unique server name.
- Set NODE to the appropriate dbserver name from the Informix sqlhosts file. This value is case-sensitive. For an example, see Figure 14 on page 124.

  Note that you do not need to use the DataJoiner **CATALOG NODE** command for any data sources that are accessed through the Informix-Net Version 5 data access module.
- Set DATABASE to the name of an Informix database at the Informix server. This value is case-sensitive.
- Set TYPE to `INFORMIX`.
- Set VERSION to the version of Informix that the Informix server is running. For example, `5.0` for Informix 5.03 (only one-digit numbers and two-digit numbers separated by a decimal point are supported)
- Set PROTOCOL to the name of a data access module you built in step 4 on page 126. For example, `"informix"`. (See Figure 14 on page 124). This value is case-sensitive.

A sample entry for Informix Version 5 is:

```
CREATE SERVER MAPPING FROM infrmxa TO NODE "infrmxa1" DATABASE "test_db"
TYPE informix VERSION 5.0 PROTOCOL "informix"
```

Preserving case-sensitivity:

The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.

- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

7. Update the `.netrc` file of the DataJoiner instance with the host name, userid, and password that DataJoiner will use to access the dbserver name you specified in SYSCAT.SERVERS; for example:

   ```
   machine infmxa1 login joeuser password brody
   ```

   where:

   infmxa1
   Is the dbserver name that is specified in the Informix sqlhosts file

   joeuser
   Is the login-id that is to be sent to the Informix server

   brody Is the password for the specified login-id

   If `.netrc` already contains an entry for this host, you do not need to add another. DataJoiner uses the same .netrc entry to access all Informix databases on a given system.

8. If you want the Informix data access module to start dynamically when the DataJoiner instance is started, update the db2profile of the DataJoiner instance to include an entry for the DJXCOMM variable. If you do not set the DJXCOMM variable, you will need to manually start the Informix data access module.

   The name of each unique Informix data access module you created should be specified in the DJXCOMM variable. For example, if you are using the data access module that is named `informix`, the entry would be:

   ```
   export DJXCOMM=informix
   ```

   Ensure that there are no spaces on either side of the equal sign (=).

   See “Step 3: Set Environment Variables” on page 58 for more information.

9. Invoke **db2profile** through either of the following methods:

   - From the DataJoiner instance login, run .profile. For example:

     ```
     . .profile
     ```

   - Invoke db2profile directly. For example:

     ```
     . /home/djinst1/sqllib/db2profile
     ```

10. Recycle (**db2stop** followed by **db2start**) the DataJoiner instance.

11. Use additional DataJoiner DDL statements to refine access to the data sources you defined using CREATE SERVER MAPPING statements. Other steps, such as adding server options and creating a user mapping are not required at this time; but, they are suggested. See the *DataJoiner*

Application Programming and SQL Reference Supplement for more information about DDL statements and when to use them.

Sample DDL statements for an Informix data source are:

```
CREATE SERVER OPTION colseq FOR SERVER infrmxa
SETTING 'I'
```

```
CREATE USER MAPPING FROM marlow TO SERVER infrmxa
AUTHID "bigdog" PASSWORD "growler"
```

12. Create nicknames. When you create nicknames, remember to put double quotes around the Informix schema and table name if they are case sensitive. For example:

    ```
    CREATE NICKNAME sanjose.employees FOR infmx."stl"."employees"
    ```

    See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about the CREATE NICKNAME statement.

13. Test the connections to your data sources. This step is optional but recommended because it allows you to determine if your data source is configured correctly.

    Prerequisite step:

    Before testing, you must perform user mapping as mentioned in step 11 on page 128. See the *DataJoiner Application Programing and SQL Reference* for more information.

    To test your location transparency connections through DataJoiner, complete the following tasks:

    a. Connect to the DataJoiner database.
    b. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

    Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Using the Informix Version 7 Data Access Module

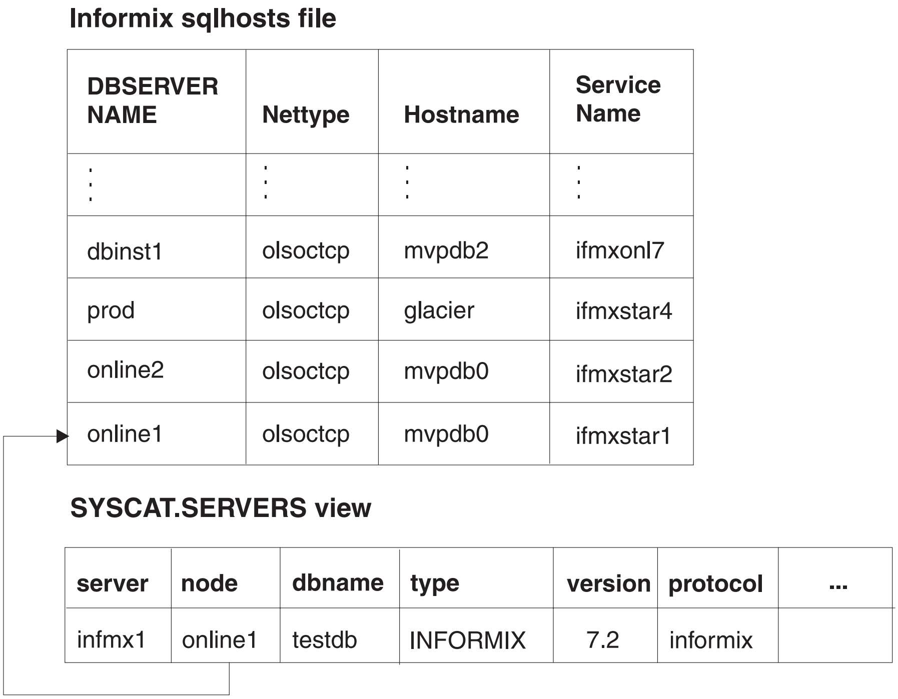
Notes:

- AIX/6000 requires that the Portable Streams Environment be loaded in order to access Informix databases through Informix Version 7.
- DataJoiner supports Informix ALS Versions 5 and 6.

To configure DataJoiner access to Informix databases through Informix Version 7:

1. If you have not already done so, install the Informix esql/rt or equivalent libraries on the DataJoiner system.

2. Follow the directions in the *Informix—OnLine Administrator's Guide* to create an `sqlhosts` file under the `$INFORMIXDIR/etc` directory. Figure 15 shows the general structure of an `sqlhosts` file.



Informix sqlhosts file

DBSERVER NAME	Nettype	Hostname	Service Name
⋮	⋮	⋮	⋮
dbinst1	olsoctcp	mvpdb2	ifmxonl7
prod	olsoctcp	glacier	ifmxstar4
online2	olsoctcp	mvpdb0	ifmxstar2
online1	olsoctcp	mvpdb0	ifmxstar1

SYSCAT.SERVERS view

server	node	dbname	type	version	protocol	...
infmx1	online1	testdb	INFORMIX	7.2	informix	

Figure 15. Informix V7 Data Access Module

3. If you have not already done so, build an Informix version 7 data access module by running `djxlink.sh`, as described in "Step 3: Link DataJoiner to Data Source Client Software" on page 46.

   For DataJoiner to send requests to Informix, it needs a data access module (a file) that has been built by link-editing DataJoiner libraries with Informix libraries. For Informix Version 7 databases, you need only one data access module regardless of the number of databases you will be accessing.

   When you run `djxlink.sh`, if you receive a message that indicates that the Informix 7 data access module could not be built, you will need to edit the sample makefile so that the makefile works with your particular setup. The makefile is:

   - `/usr/lpp/djx_02_01_01/lib/djxlink.makefile` on AIX systems.
   - `/opt/IBMdjx/V2.1.1/lib/djxlink.makefile` on Solaris systems.

   Editing and running `djxlink.makefile` are also described in "Step 3: Link DataJoiner to Data Source Client Software" on page 46.

If `djxlink.sh` fails to build the Informix version 7 data access module, login as root and edit `djxlink.makefile`. Find the section in the file that is titled ‘Accessing Data Sources Through the Informix Data Access Module (Informix Version 7)’.

a. In the makefile, respecify the DJX_INFORMIX7_LIBRARY_PATH variable to be the directory specified in step 1 on page 129. This directory should be the location of the libraries for your Informix Version 7 driver.

   Attention:
   If you are using the ESQL/C library, see the informix7c sample in the `djxlink.makefile`.

   The unique file name is the value you specify as the PROTOCOL value in the CREATE SERVER MAPPING DDL statement.

b. The name of the Informix 7 database access module that is built by the makefile is `informix7`. If you do not want to change this name, skip to the next step.

   If you want to change the name, you need to change two lines in the makefile:

   - Change the target name on the dependency line. For example, if you want the data access module to be named `ifmx7`, change the dependency line from this:

     ```
     informix7: libinformix7.a informixEP.o informix.exp informix.imp
     ```

     To this:

     ```
     ifmx7: libinformix7.a informixEP.o informix.exp informix.imp
     ```

   - Change the output name of the link-edit statement. For example, change this:

     ```
     ld -o informix7 \
     ```

     To this:

     ```
     ld -o ifmx7 \
     ```

   The name will be used in the SERVER_PROTOCOL column of SYSCAT.SERVERS for the first database that is accessed using Informix Version 7. The SERVER_PROTOCOL column has a restriction that the filename cannot be an existing data access module name unless you are replacing it.

c. As root, link-edit the informix7 data access module library using the supplied makefile. To complete this task, you need the following:
   - The informix7 data access module library that is provided with DataJoiner
   - The informix.exp file that is provided with DataJoiner

- The Informix esql/rt libraries that are provided with the Informix esql product

From the /usr/lpp/djx_02_01_01/lib directory on AIX systems or the /opt/IBMdjx/V2.1.1/lib directory on Solaris systems, execute the makefile by entering one of the following commands:

- If using ESQL/RT, enter:

  ```
  make -f djxlink.makefile informix7
  ```
- If using ESQL/C, enter:

  ```
  make -f djxlink.makefile informix7c
  ```

If you changed the name of the data access module in the makefile, then use that name with the **make** command.

Whenever you re-run djxlink.makefile for the informix data access module, you must stop and restart all DataJoiner instances that use the informix7 data access module.

Complete steps 4 through 7 on page 133 for each DataJoiner instance that will use the informix data access module.

4. Use the CREATE SERVER MAPPING DDL statement to define each Informix database you expect to access via Informix Version 7. Specify the same protocol for each database. When performing server mapping:
   - Choose a unique server name.
   - Set NODE to the appropriate Informix dbserver name. This value is case sensitive.

     Note that you do not need to use the **CATALOG NODE** command for the data sources that are accessed through the Informix Version 7 data access module.
   - Set DATABASE to the name of an Informix database at the Informix server that is listed in the NODE column. This value is case-sensitive.
   - Set TYPE to informix.
   - Set VERSION to the version of Informix that the Informix server is running. For example, 7.2 for Informix 7.2.
   - Set PROTOCOL to "informix7". This value is case sensitive.

   A sample entry for Informix Version 7 is:

   ```
   CREATE SERVER MAPPING FROM infrmxb TO NODE "infrmxb1" DATABASE "test_db"
   TYPE informix VERSION 7.2 PROTOCOL "informix7"
   ```

   Preserving case-sensitivity:
   The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

5. Update the `.profile` file of the DataJoiner instance to include the following lines:

   ```
   export INFORMIXDIR="informix-home-directory"
   export INFORMIXSERVER="dbserver-name"
   ```

   Where *informix-home-directory* is the directory in which you installed the e/sql library in step 1 on page 129, and *dbserver-name* is the name that is entered in the first column of the sqlhosts file.

   Some releases of Informix require you to set INFORMIXSERVER with the name of a default server. However, this value is not used by DataJoiner. Instead, DataJoiner uses the dbserver-name set with the NODE parameter of the CREATE SERVER MAPPING statement.

   Be sure there are no spaces on either side of the equal sign (=) in these export statements. See the *Informix—OnLine Administrator's Guide* for more information on these environment variables.

6. If you want the Informix data access module to start dynamically when the DataJoiner instance is started, update the db2profile of the DataJoiner instance to include an entry for the DJXCOMM variable. The name of the Informix7 data access module should be specified. If you do not set the DJXCOMM variable, you will need to manually start the Informix data access module.

   For example, if you are using a data access module that is named `informix7`, the entry would be:

   ```
   export DJXCOMM=informix7
   ```

   Ensure that there are no spaces on either side of the equal sign (=). See “Step 3: Set Environment Variables” on page 58 for more information.

7. Invoke **db2profile** through either of the following methods:
   - From the DataJoiner instance login, run `.profile`. For example:

     ```
     . .profile
     ```

   - Run it directly. For example:

     ```
     . /home/djinst1/sqllib/db2profile
     ```

8. Use additional DataJoiner DDL statements to refine access to the data sources you defined using CREATE SERVER MAPPING statements. Other steps, such as adding server options and creating a user mapping are not

required at this time; but, they are suggested. See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about DDL statements and when to use them.

Sample DDL statements for an Informix data source are:

```
CREATE SERVER OPTION colseq FOR SERVER infrmxb
SETTING 'I'

CREATE USER MAPPING FROM marlow TO SERVER infrmxb
AUTHID "bigdog" PASSWORD "growler"
```

9. Create nicknames. When you create nicknames, remember to put double quotes around the Informix schema and table name if they are case sensitive. For example:

   ```
   CREATE NICKNAME sanjose.employees FOR infmx."stl"."employees"
   ```

   See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about the CREATE NICKNAME statement.

10. Test the connections to your data sources. This step is optional but recommended because it allows you to determine if your data source is configured correctly.

    Prerequisite step:
    Before testing, you must perform user mapping as mentioned in step 8 on page 133. See the *DataJoiner Application Programing and SQL Reference* for more information.

    To test your location transparency connections through DataJoiner, complete the following tasks:

    a. Connect to the DataJoiner database.
    b. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

    Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Chapter 11. Accessing Oracle Data Sources Using SQL*Net or Net8

Two different Oracle data access modules are included with DataJoiner: one for use with the SQL*Net V1 or V2 client software, and one for use with the Net8 client software. The client software you use dictates which data access module you will use. If you use SQL*Net, you must use the sqlnet data access module, and if you use Net8, you must use the net8 data access module. Regardless of the client software you use, however, you can access both Oracle Version 7 and Oracle Version 8 data sources.

Restriction:

Oracle's Net8 client software is supported on AIX 4.1.5 or later and Solaris 2.5.1 or later only.

To set up access to Oracle databases:

1. Install Oracle SQL*Net or Net8 on the DataJoiner system.

   It is recommended that you request a relink of SQL*Net or Net8 during the install process. This information will be used when DataJoiner is linked to this software.
2. If not already completed as part of install, use djxlink.sh as described in "Step 3: Link DataJoiner to Data Source Client Software" on page 46 to link-edit Oracle SQL*Net or Net8 libraries to DataJoiner.
3. Configure SQL*Net or Net8. See the Oracle documentation for configuration instructions.

Complete Step 4 through Step 9 for each DataJoiner instance that will use the Oracle data access module.

4. If you will be accessing Oracle 7.3 or later databases, enable the Asynchronous I/O extension.
5. If not already done, Set the ORACLE_HOME environment variable. SQL*Net and Net8 require that this variable be set prior to starting your DataJoiner instance. If this variable is changed, the DataJoiner instance must be stopped and restarted for the new ORACLE_HOME value to take effect.

   The value of ORACLE_HOME that you set for the DataJoiner instance is the one that is used by DataJoiner. If an individual user of DataJoiner has the ORACLE_HOME environment variable set, it is not used by DataJoiner.

   Set environment variables in the DataJoiner instance owner's profile by entering the following commands:

```
export ORACLE_HOME=oracle-home-directory
export ORACLE_BASE=oracle-home-directory
```

Ensure that there are no spaces on either side of the equal sign (=).

ORACLE_BASE is optional. It is used to locate the tnsnames.ora file.

6. Ensure that the SQL*Net or Net8 `tnsnames.ora` file (usually found in the `/etc` directory) is updated for each Oracle server to which communications are configured. The SID is the name of the Oracle instance. The HOST is the host name where the Oracle server is located.
7. If the TNS_ADMIN environment variable is set in the environment of the DataJoiner instance, it will be used to locate the tnsnames.ora file. TNS_ADMIN is optional.

   Set the TNS_ADMIN environment variable in the DataJoiner instance owner's profile by entering the following command:

   ```
   export TNS_ADMIN=$ORACLE_BASE/admin/util/network
   ```

   Ensure that there are no spaces on either side of the equal sign (=).
8. If you are using Oracle 7.2, set the ORA_NLS environment variable in the DataJoiner instance owner's profile by entering the following command:

   ```
   export ORA_NLS=$ORACLE_HOME/ocommon/nls/admin/data
   ```
9. Recycle (**db2stop**, then **db2start**) the DataJoiner instance.
10. Use the CREATE SERVER MAPPING DDL statement to define each Oracle server to which communications are configured. When performing server mapping:
    - Choose a unique server name.
    - Set NODE to the appropriate database server name from the tnsnames.ora file. This value is case sensitive. For an example, see Figure 16 on page 138.

      Note that you do not need to use the **CATALOG NODE** command for Oracle data sources.
    - Set TYPE to `oracle`.
    - Set VERSION to the data source version that the server is running. For example, `7.2` for Oracle 7.2.
    - Set PROTOCOL to `"sqlnet"` if using Oracle's SQL*Net client software, or to `"net8"` if using Oracle's Net8 client software. This value is case sensitive.

    A sample entry for an Oracle data source is:

    ```
    CREATE SERVER MAPPING FROM oraclela TO NODE "orala" TYPE oracle
    VERSION 7.2 PROTOCOL "sqlnet"
    ```

A DATABASE value is not required; Oracle does not have multiple databases per instance to which you must connect to access data.

Preserving case-sensitivity:

The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

11. If you want the Oracle data access module to start dynamically when the DataJoiner instance is started, update the db2profile of the DataJoiner instance to include an entry for the DJXCOMM variable. For example, if you are using the data access module named sqlnet, the entry would be:

    ```
    export DJXCOMM=sqlnet
    ```

    Ensure that there are no spaces on either side of the equal sign (=).

    See “Step 3: Set Environment Variables” on page 58 for more information.

12. Use additional DataJoiner DDL statements to refine access to the data sources you defined using CREATE SERVER MAPPING statements. Other steps (adding server options, creating a user mapping, creating a nickname) are not required at this time; but, they are suggested. See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about DDL statements and when to use them.

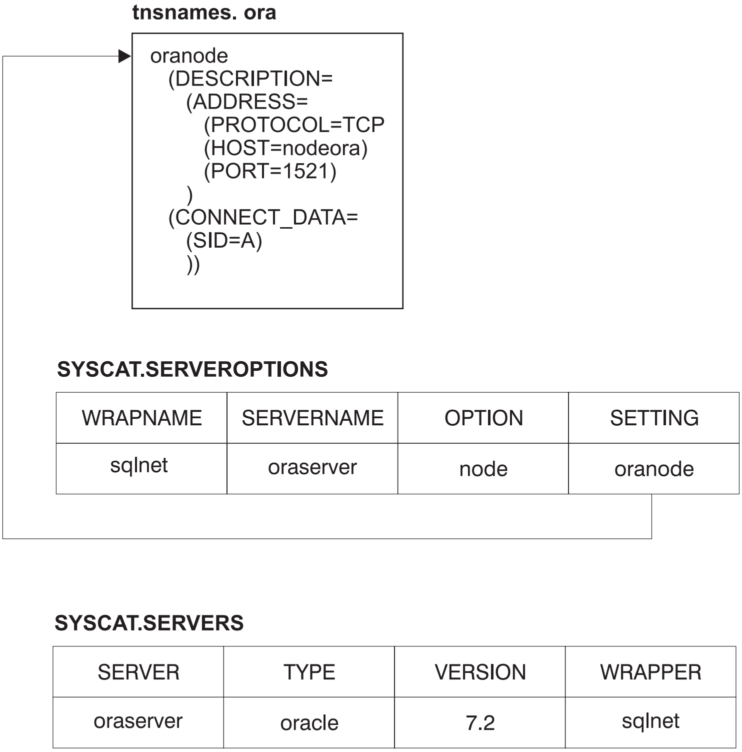
    Sample DDL statements are:

    ```
    CREATE SERVER OPTION colseq FOR SERVER oraclela
    SETTING 'N'

    CREATE USER MAPPING FROM marlow TO SERVER oraclela
    AUTHID "bigdog" PASSWORD "growler"

    CREATE NICKNAME dogpound FOR oraclela.bigdog.dogpound
    ```

    Attention:

    The Oracle user (at the Oracle data source, not DataJoiner), must have been created using the Oracle **create user** command with the ‘identified by’ clause rather than the ‘identified externally’ clause.

Figure 16 on page 138 shows the information in the tnsnames.ora file and SYSCAT.SERVERS view. For Oracle, the value of dbname is a null string (“”) because Oracle does not have multiple databases in one instance of its database manager.



tnsnames. ora

```
oranode
  (DESCRIPTION=
    (ADDRESS=
      (PROTOCOL=TCP
      (HOST=nodeora)
      (PORT=1521)
    )
  (CONNECT_DATA=
    (SID=A)
    ))
```

SYSCAT.SERVEROPTIONS

WRAPNAME	SERVERNAME	OPTION	SETTING
sqlnet	oraserver	node	oranode

SYSCAT.SERVERS

SERVER	TYPE	VERSION	WRAPPER
oraserver	oracle	7.2	sqlnet

Figure 16. ORACLE Data Access Module

13. For each ’HOST’ in the DESCRIPTION section of the tnsnames.ora file, update /etc/hosts if necessary.

    Whether you must update /etc/hosts depends on how TCP/IP is configured on your network. Part of the network must translate the remote hostname specified in the DESCRIPTION section in the tnsnames.ora file (″nodeora″ in our example) to an address. If your network has a name server that recognizes the hostname, you do not need to update /etc/hosts. Otherwise, you need an entry for the remote host. See your network administrator to learn how your network is configured.

14. Test the connections to your data sources. This step is optional but recommended because it allows you to determine if your data source is configured correctly.

    Prerequisite step:
    Before testing, you must perform user mapping as mentioned in step 12 on page 137. See the *DataJoiner Application Programing and SQL Reference* for more information.

To test your location transparency connections through DataJoiner, complete the following tasks:

a. Connect to the DataJoiner database.

b. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Chapter 12. Accessing Microsoft SQL Server and Sybase Data Sources Using Sybase Open Client

| The Sybase data access module is included with DataJoiner for AIX and DataJoiner for Solaris systems and supports both the ctlib and dblib access mechanisms.

| DataJoiner for AIX systems can also use the Sybase Open Client dblib API to access pre-Version 7 Microsoft SQL Server data sources.

To set up access to Sybase and Microsoft SQL Server databases:

1. Install catalog stored procedures on the Sybase or Microsoft SQL Server node. In the Sybase documentation, see the *Command Reference Manual for Sybase SQL Server* for details.
2. Install the Sybase Open Client libraries on the DataJoiner system.
3. If not already completed as part of install, use `djxlink.sh`, as described in "Step 3: Link DataJoiner to Data Source Client Software" on page 46, to link-edit Sybase Open Client libraries with DataJoiner.

Complete steps 4 through 7 for each DataJoiner instance that will use the Sybase data access module.

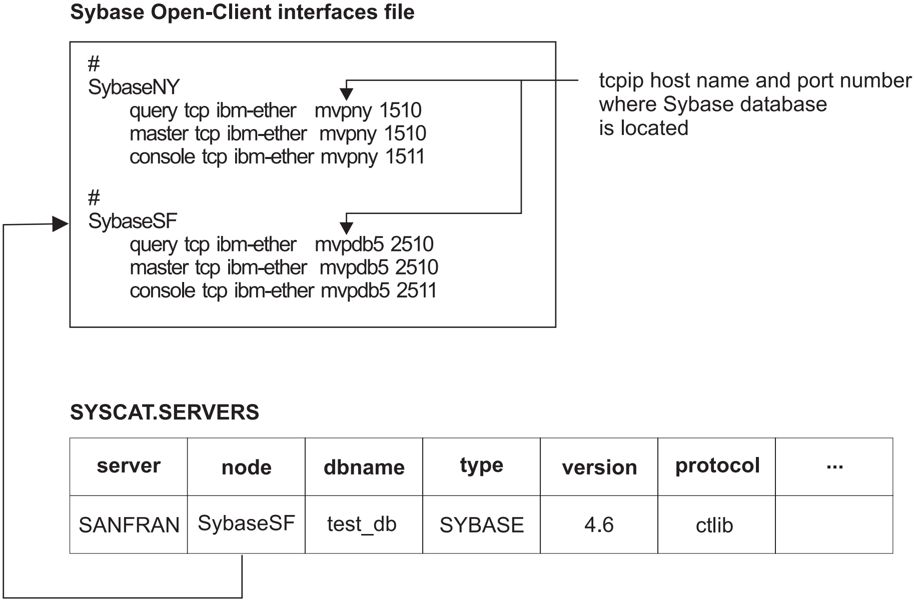
4. Update the `.profile` file of the DataJoiner instance with the SYBASE environment variable. Perform an export with the following commands:

   ```
   export SYBASE="sybase home directory"
   export PATH="$SYBASE/bin:$PATH"
   ```

   Make sure that there are no spaces on either side of the equal sign (=).
5. Execute the DataJoiner instance ·profile by entering:

   ```
   . .profile
   ```
6. Recycle (**db2stop**, then **db2start**) the DataJoiner instance.
7. Use the Sybase-supplied utility to create an interfaces file that includes the data for all the Sybase Open Servers and Microsoft SQL Servers you want to access. See the *Sybase System Administration Guide* for more information on this procedure.

   Figure 17 on page 142 contains an example of a Sybase Open Client interfaces file:



Sybase Open-Client interfaces file

```
#
SybaseNY
      query tcp ibm-ether   mvpny 1510
      master tcp ibm-ether  mvpny 1510
      console tcp ibm-ether mvpny 1511

#
SybaseSF
      query tcp ibm-ether   mvpdb5 2510
      master tcp ibm-ether  mvpdb5 2510
      console tcp ibm-ether mvpdb5 2511
```

SYSCAT.SERVERS

server	node	dbname	type	version	protocol	...
SANFRAN	SybaseSF	test_db	SYBASE	4.6	ctlib	

Figure 17. Sybase Data Access Module. The database server name in the Open Client interfaces file corresponds to the NODE column in SYSCAT.SERVERS; the host machine name and port number correspond to the TCP/IP host machine name and port address for the Sybase data source.

8. Make the interfaces file accessible by doing *one* of the following tasks:
   - Put the file in the DataJoiner instance's `$HOME/sqllib` directory.
   - Using the **ln** command, link to the file from the DataJoiner instance's `$HOME/sqllib` directory. For example,

     ```
     ln -s -f /home/sybase/interfaces  /home/djinst1/sqllib
     ```

   - Specify the path and name of the interfaces file by using CONNECTSTRING in SYSCAT.SERVER_OPTIONS. For information about CONNECTSTRING, see "Specifying a Connect String for Open Client" on page 144.
9. Use the CREATE SERVER MAPPING DDL statement to define each database you want to access using Open Server Continue. When performing server mapping:
   - Choose a unique server name.
   - Set NODE to the appropriate database server name from the Open Client interfaces file. This value is case sensitive. For an example, see Figure 17.

     Note that you do not need to use the **CATALOG NODE** command for data sources that are accessed through Open Client.
   - Set DATABASE to the name of a database at the server that is listed as the NODE value. This value is case sensitive.

- Set TYPE:
  - For Sybase data sources, specify `sybase`.
  - For Microsoft SQL Server data sources, specify `MSSQLSERVER`.
- Set VERSION to the data source version that the server is running. For example, `4.3` for Sybase 4.3.
- Set PROTOCOL:
  - For Sybase data sources, specify either `"ctlib"` or `"dblib"`
  - For Microsoft SQL Server data sources, specify `"dblib"`

  This value is case sensitive.

A sample entry for a Sybase data source is:

```
CREATE SERVER MAPPING FROM sybase1 TO NODE "sybase1" DATABASE "test_db"
TYPE sybase VERSION 4.3 PROTOCOL "dblib"
```

A sample entry for Microsoft SQL Server Version 6 is:

```
CREATE SERVER MAPPING FROM MsoftV6 TO NODE "MsoftV6" DATABASE "test_db"
TYPE MSSQLServer VERSION 6.0 PROTOCOL "dblib"
```

Notice that the server value is the same as the node value. The NODE value must match the server name in the Open Client interfaces file.

Preserving case-sensitivity:
The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

10. If you want the ctlib or dblib data access module to start dynamically when the DataJoiner instance is started, update the db2profile of the DataJoiner instance to include an entry for the DJXCOMM variable. For example, if you are using both the ctlib and dblib data access modules, the entry would be:

    ```
    export DJXCOMM=ctlib,dblib
    ```

    Ensure that there are no spaces on either side of the equal sign (=).

    See “Step 3: Set Environment Variables” on page 58 for more information.

11. Use additional DataJoiner DDL statements to refine access to the data sources you defined using CREATE SERVER MAPPING statements. Other steps (adding server options, creating a user mapping, creating a

nickname) are not required at this time; but, they are suggested. See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about DDL statements and when to use them.

Sample DDL statements for a Sybase data source are:

```
CREATE SERVER OPTION colseq FOR SERVER sybase1
SETTING 'I'
```

```
CREATE USER MAPPING FROM marlow TO SERVER sybase1
AUTHID "bigdog" PASSWORD "growler"
```

```
CREATE NICKNAME dbo.sysobjects FOR mssoft.\"dbo\".\"sysobjects\"
```

12. Test the connections to your data sources. This step is optional but recommended because it allows you to determine if your data source is configured correctly.

    Prerequisite step:

    Before testing, you must perform user mapping as mentioned in the previous step. See the *DataJoiner Application Programing and SQL Reference* for more information.

    To test your location transparency connections through DataJoiner, complete the following tasks:

    a. Connect to the DataJoiner database.
    b. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

    Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Specifying a Connect String for Open Client

Open Client uses timeout thresholds to interrupt queries that run for too long a period of time. You can set these thresholds in DataJoiner by using the CONNECTSTRING option of the CREATE SERVER OPTION DDL statement.

Use the CONNECTSTRING option to specify:

- Timeout duration for SQL queries
- Timeout duration for login response
- Path and name of an interfaces file to use
- Packet size

Syntax

The syntax of CONNECTSTRING entries is:

```
    ┌─;──────────────────────────────────┐
    ▼                                    │
►►─────┬─TIMEOUT── = ──seconds───────────┬┴───────────────────────────────────────►◄
       ├─LOGIN_TIMEOUT── = ──seconds─────┤
       ├─IFILE── = ──"string"────────────┤
       ├─PACKET_SIZE── = ──bytes─────────┤
       └─;───────────────────────────────┘
```

TIMEOUT
Specifies the number of seconds for DataJoiner to wait for a response from Open Client for any SQL statement. The value of *seconds* is a positive whole number in DataJoiner's integer range or the default value, which for dblib is 0, and for ctlib is -9999. Both of these default values cause DataJoiner to wait indefinitely for a response.

LOGIN_TIMEOUT
Specifies the number of seconds for DataJoiner to wait for a response from Open Client to the login request.

IFILE
Specifies the path and name for the Open Client interfaces file to use. The path that is identified in *string* must be enclosed in double quotes (″). The default is `sqllib/interfaces` in the home directory of your DataJoiner instance.

PACKET_SIZE
Specifies the packet size in bytes. If the data source does not support the specified packet size, the connection will fail. Increasing the packet-size when each record is very large (for example, when inserting rows into large tables) significantly increases performance. The byte size is a numeric value. See the Sybase reference manuals for more information.

These keywords are not case sensitive. The parameters can be specified in any order, but can be specified only once. The semicolon (;) must be present for each parameter and its value. The equal sign (=) must be present. There can be any number of tabs or spaces that separate the parameter from the equal sign, the equal sign from the value, and the value from the semicolon.

Examples

To set the timeout value to 60 seconds, set the connect string to:

```
TIMEOUT=60;
```

To set the timeout value to 60 seconds and the interfaces file to /etc/interfaces, set the connect string using the following example:

```
CREATE SERVER OPTION connectstring FOR SERVER sybase1
SETTING 'TIMEOUT=60;IFILE="/etc/interfaces";'
```

Chapter 13. Accessing Microsoft SQL Server Sources through MERANT DataDirect Drivers

DataJoiner can access Microsoft SQL Server 6.5 and 7.0 through the MERANT DataDirect Drivers. DataJoiner for AIX systems can also use the Sybase Open Client dblib API to access pre-Version 7 Microsoft SQL Server data sources. (See “Chapter 12. Accessing Microsoft SQL Server and Sybase Data Sources Using Sybase Open Client” on page 141.)

This chapter explains how to configure DataJoiner to access SQL Server 6.5 through MERANT DataDirect Connect ODBC Version 3.5 and SQL Server 7 through MERANT DataDirect SequeLink ODBC Edition Version 4.51.

Restriction:
The MERANT DataDirect drivers are supported on AIX 4.2.1.5 and later only.

Accessing Microsoft SQL Server Data Sources through the MERANT DataDirect Connect ODBC Driver

The protocol is named `mssqlodbc`. The server option IGNORE_UDT is supported for user-defined data types. See the CREATE SERVER OPTION in the *DataJoiner Application Programming and SQL Reference Supplement* for more information.

Setting up the MERANT DataDirect Connect ODBC 3.5 Driver

To access a Microsoft SQL Server V6.5 data source through the MERANT Data Direct Connect ODBC driver:

1. Install the nothread version of the libraries supplied by MERANT. Please see the MERANT `READ.ME` file that is supplied with DataDirect Connect ODBC for instructions on how to install the non-reentrant AIX driver. DataJoiner will not work with the other driver.
2. Set up the environment variables for all accessed data sources. Set `DJX_ODBC_LIBRARY_PATH` to the installation path of the MERANT DataDirect Connect ODBC Version 3.5 for AIX libraries; for example:

   ```
   export DJX_ODBC_LIBRARY_PATH=/merant/lib
   ```
3. Specify the MERANT library directory as the first entry in the LIBPATH; for example:

   ```
   export LIBPATH=/merant/lib:other_libpaths
   ```

where *other_libpaths* is your system’s previous library path information. Then add the following line to the bottom of the odbc.sh script:

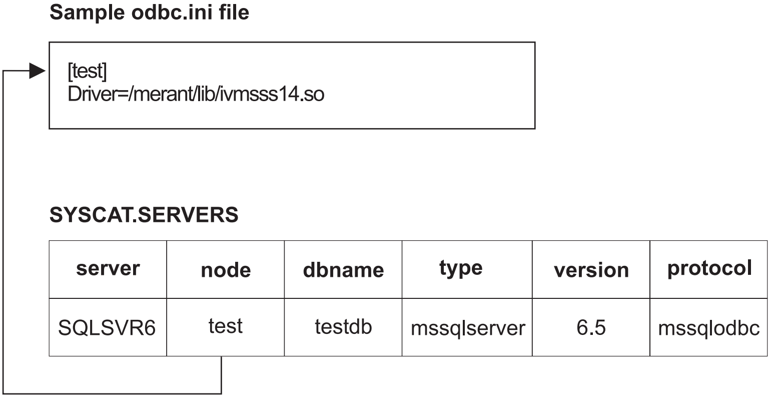
```
DB2ENVLIST=LIBPATH
export DB2ENVLIST
```

The odbc.sh script is supplied with the MERANT drivers and must be called from the .profile of the DataJoiner instance.

4. Run djxlink.sh. The djxlink.sh shell script creates a data access module named mssqlodbc. See “Running djxlink.makefile” on page 50.
5. Use the CREATE SERVER MAPPING DDL statement to define each SQL server that you expect to access via DataJoiner. For each CREATE SERVER MAPPING statement:
   - Choose a unique server name.
   - Set NODE to the appropriate database server name from the obdc.ini file. This value is case-sensitive. For an example, see Figure 18.
   - Set DATABASE to the name of a database at the server that is listed as the NODE value. This value is case-sensitive.
   - Set TYPE to mssqlserver.
   - Set VERSION to the data source version that the server is running. For example, 6.5 for SQL Server Version 6.5.
   - Set PROTOCOL to "mssqlodbc". This value is case-sensitive.

   A sample entry for a Microsoft SQL Server 6.5 data source is:

   ```
   CREATE SERVER MAPPING FROM SQLSVR6 TO NODE "test" DATABASE "testdb"
   TYPE "mssqlserver" VERSION 6.5 PROTOCOL "mssqlodbc"
   ```



```
[test]
Driver=/merant/lib/ivmsss14.so
```

server	node	dbname	type	version	protocol
SQLSVR6	test	testdb	mssqlserver	6.5	mssqlodbc

Figure 18. MERANT DataDirect Connect ODBC Driver Data Access Module

Preserving case-sensitivity:

The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes. You need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

Obtaining ODBC Traces

Use the information in this section to obtain ODBC tracing information, which is used to analyze and resolve problems.

- ODBC tracing will work properly with DataJoiner only if the MERANT libaries are located in /usr/lib or /lib. Assuming that the MERANT drivers are installed in the /merant directory, set up symbolic links from /usr/lib to /merant, as follows:

```
cd /usr/lib
ln -s /merant/lib/libivbas14.a
ln -s /merant/lib/libivflt14.a
ln -s /merant/lib/libivmback.so
ln -s /merant/lib/libivmfront.a
ln -s /merant/lib/libivutl14.a
ln -s /merant/lib/libodbc.a
ln -s /merant/lib/libodbcinst.a
ln -s /merant/lib/liboraclemap.a
ln -s /merant/lib/odbc.so
ln -s /merant/lib/odbccurs.so
ln -s /merant/lib/odbcinst.so
ln -s /merant/lib/odbctrac.so
ln -s /merant/lib/vscnctdlg.so
ln -s /merant/lib/ivdb214.so
ln -s /merant/lib/ivdbf14.so
ln -s /merant/lib/ivinf914.so
ln -s /merant/lib/ivmsss14.so
ln -s /merant/lib/ivoi214.so
ln -s /merant/lib/ivoing14.so
ln -s /merant/lib/ivor714.so
ln -s /merant/lib/ivor814.so
ln -s /merant/lib/ivsyb1114.so
ln -s /merant/lib/ivtxt14.so
```

- ODBC tracing will work properly with DataJoiner only if default tracing is disabled, and the entry in odbc.ini (in the home directory of the DB2 instance) specifies:

```
[ODBC]
Trace=0
TraceDll=/usr/lib/odbctrac.so
InstallDir=/merant
```

DataJoiner will turn on ODBC tracing during connect if the DJXODBCTRACE environment variable is defined. The variable must specify the name of an existing directory, with write access, for example:

```
export DJXODBCTRACE=/tmp/odbc_trace_directory_name
```

ODBC trace file names are generated from the DataJoiner process ID and connection handle, for example:

```
/tmp/odbc_trace_directory_name/djxodbc.trace.60932.20058a78
```

Accessing Microsoft SQL Server Data Sources through the MERANT DataDirect SequeLink ODBC Edition Driver

The protocol is named "djxsqlnk". The server option IGNORE_UDT is supported for user-defined data types. See CREATE SERVER OPTION in the *DataJoiner Application Programming and SQL Reference Supplement* for more information.

Setting up MERANT DataDirect SequeLink ODBC Edition 3.5 Driver

The following instructions explain how to access a Microsoft SQL Server Version 7 data source through the MERANT SequeLink ODBC Edition 3.5 Driver.

The MERANT driver requires AIX 4.2.1.5 or later.

1. Install the nothread version of the libraries supplied by MERANT. See the MERANT READ.ME file supplied with MERANT DataDirect SequeLink ODBC Edition Driver for instructions on how to install the non-reentrant AIX driver. DataJoiner will not work with the other driver.
2. Set up the environment variables for all accessed data sources. Specify the variables in the .profile of the instance owner.

   Set SQLNK_HOME to the installation path of the MERANT DataDirect SequeLink ODBC Edition Driver for AIX libraries; for example:

   ```
   SQLNK_HOME=/sequelink/4_51_00/lib
   ```
3. Specify the MERANT library directory as the first entry in the LIBPATH; for example:

   ```
   export LIBPATH=/sequelink/4_51_00/lib:other_libpaths
   ```

   where *other_libpaths* is your system’s previous library path information. Then add the following line to the bottom of the odbc.sh script:

   ```
   DB2ENVLIST=LIBPATH
   export DB2ENVLIST
   ```

   The odbc.sh script is supplied with the MERANT drivers and must be called from the .profile of the DataJoiner instance.

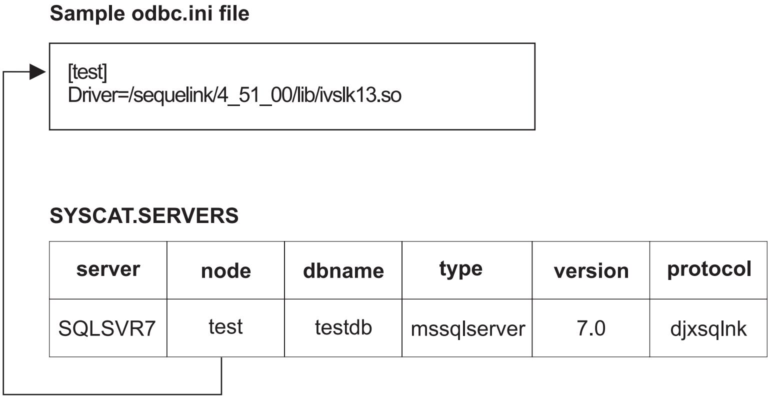
4. Run `djxlink.sh`. The `djxlink.sh` shell script creates two DataJoiner protocol modules for Microsoft SQL Server:

   ```
   mssqlodbc  -- DataDirect Connect for SQL Server Version 6.5 data sources
   djxsqlnk   -- DataDirect SequeLink for SQL Server Version 7 data sources
   ```

   MERANT supports SQL Server Version 7 data sources via DataDirect SequeLink. See “Running djxlink.makefile” on page 50.
5. Use the CREATE SERVER MAPPING DDL statement to define each SQL server that you expect to access via DataJoiner. For each CREATE SERVER MAPPING statement:
   - Choose a unique server name.
   - Set NODE to the appropriate database server name from the obdc.ini file. This value is case-sensitive. For an example, see Figure 19.
   - Set DATABASE to the name of a database at the server that is listed as the NODE value. This value is case-sensitive.
   - Set TYPE to `mssqlserver`.
   - Set VERSION to the data source version that the server is running. For example, `7.0` for SQL server 7.
   - Set PROTOCOL to "`djxsqlnk`". This value is case-sensitive.

   A sample entry for a SQL Server Version 7.0 data source is:

   ```
   CREATE SERVER MAPPING FROM SQLSVR7 TO NODE "test" DATABASE "testdb"
     TYPE "mssqlserver" VERSION 7.0 PROTOCOL "djxsqlnk"
   ```



Sample odbc.ini file

```
[test]
Driver=/sequelink/4_51_00/lib/ivslk13.so
```

SYSCAT.SERVERS

server	node	dbname	type	version	protocol
SQLSVR7	test	testdb	mssqlserver	7.0	djxsqlnk

Figure 19. MERANT DataDirect SequeLink ODBC Edition Driver Data Access Module

Preserving case-sensitivity:
The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes. You need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

6. Copy /usr/lpp/djx_02_01/cfg/.envfile (with ODBCINI and SQLNK_HOME defined), to ~/sqllib/.envfile for the each DataJoiner instance, and ensure that each instance defines these two environment variables. This can be done by calling odbc.sh from the .profile of each DataJoiner instance.
7. To use the client user ID as the default high-level qualifier for table names and views, edit /sequelink/4_51/00/ini/sqlnkdsn.ini and set SUID and SPWD to blank:

```
SUID =
SPWD =
```

Obtaining ODBC Traces

Use the information in this section to obtain ODBC tracing information, which is used to analyze and resolve problems.

- ODBC tracing will work properly with DataJoiner only if the MERANT libaries are located in /usr/lib or /lib. Assuming that the MERANT drivers are installed in the /sequelink/4_51_00 directory, set up symbolic links from /usr/lib to the directory, as follows:

```
cd /usr/lib
ln -s /sequelink/4_51_00/lib/ivslk13.so
ln -s /sequelink/4_51_00/lib/libodbc.a
ln -s /sequelink/4_51_00/lib/libodbcinst.a
ln -s /sequelink/4_51_00/lib/liboraclemap.a
ln -s /sequelink/4_51_00/lib/odbccurs.so
ln -s /sequelink/4_51_00/lib/odbcinst.so
ln -s /sequelink/4_51_00/lib/odbctrac.so
ln -s /sequelink/4_51_00/lib/vscnctdlg.so
```

- ODBC tracing will work properly with DataJoiner only if default tracing is disabled, and the entry in odbc.ini (in the home directory of the DB2 instance) specifies:

```
[ODBC]
Trace=0
TraceDll=/usr/lib/odbctrac.so
InstallDir=/sequelink/4_51_00
```

  DataJoiner will turn on ODBC tracing during connect if the DJXODBCTRACE environment variable is defined. The variable must specify the name of an existing directory, with write access, for example:

```
export DJXODBCTRACE=/tmp/odbc_trace_directory_name
```

ODBC trace file names are generated from the DataJoiner process ID and connection handle, for example:

```
/tmp/odbc_trace_directory_name/djxodbc.trace.60932.20058a78
```

Chapter 14. Accessing Teradata Data Sources

DataJoiner for AIX systems can access Teradata Version 2.0.1 or later databases through the Teradata data access module. DataJoiner's Teradata support requires AIX Version 4.2.1 or higher.

Notes:

- DataJoiner for Solaris systems does not support Teradata data sources.
- Due to Teradata restrictions, DataJoiner automatically issues a COMMIT WORK after every transparent DDL statement (for example, CREATE TABLE, ALTER TABLE, DROP TABLE), even when AUTOCOMMIT is OFF.

To access Teradata databases:

1. Set up the Teradata TCP/IP `hosts` file in your `/etc` directory. You need to set up this file on each client before you can successfully request a connection to a remote Teradata server. For example, the `hosts` file should contain a line similar to this:

   ```
   192.168.100.100 tdatsvr.companyname.com  tdatsvrCOP1  # Remote Teradata Server
   ```

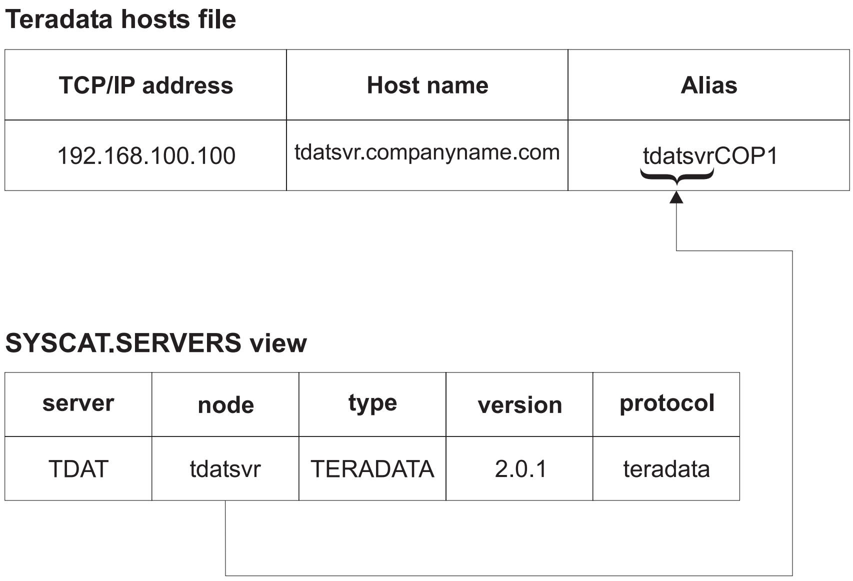
   where:

   `192.168.100.100`
   Is the TCP/IP address of the remote server.

   `tdatsvr.companyname.com`
   Is a symbolic name associated with the remote server.

   `tdatsvrCOP1`
   Is the alias for the remote server. The alias must begin with an alphabetic string and end with the `COP`*n* suffix, where *n* is a number between 1 and the total number of applications processors (APs) associated with the Teradata communications processor (COP).

   Figure 20 on page 156 shows the general structure of a hosts file.



Teradata hosts file

TCP/IP address	Host name	Alias
192.168.100.100	tdatsvr.companyname.com	tdatsvrCOP1

SYSCAT.SERVERS view

server	node	type	version	protocol
TDAT	tdatsvr	TERADATA	2.0.1	teradata

Figure 20. Teradata Data Access Module

See the *Teradata Client for Unix Installation Guide* for more information about the `hosts` file.

2. Install the Teradata client library (`libcliv2.a`) on the system where DataJoiner is installed.
3. If you did not create the Teradata data access module during installation, create it now:
   a. Before running `djxlink.sh` to create the Teradata data access module, set the TERADATA_HOME environment variable to the directory where the Teradata client library resides. This step must be performed as root.

      For example, if the path name where the Teradata client library resides is `/teradata/lib/libcliv2.a`, you would issue:

      ```
      export TERADATA_HOME=/teradata
      ```
   b. The `djxlink.sh` script automatically appends `/lib` to the value that you specify.
   c. Run the `djxlink.sh` script as described in "Running djxlink.sh" on page 46.

If `djxlink.sh` fails to successfully build the Teradata data access module, you need to edit and run djxlink.makefile to build it. See "Editing djxlink.makefile" on page 49 and "Running djxlink.makefile" on page 50 for instructions.

Complete steps 4 through 9 for each DataJoiner instance that will use the Teradata data access module.

4. Update the `.profile` file of the DataJoiner instance to include the Teradata COPLIB and COPERR environment variables; for example:

   ```
   export COPLIB=teradata-lib-directory
   export COPERR=teradata-lib-directory
   ```

   where *teradata-lib-directory* is the directory in which you installed the Teradata client library in step 2 on page 156.

   Restriction: Be sure there are no spaces on either side of the equal sign (=) in these export statements.

   See the *Teradata Client for Unix Installation Guide* for more information on the COPLIB and COPERR environment variables.

5. Use the CREATE SERVER MAPPING DDL statement to define each Teradata server that you expect to access via DataJoiner. For the CREATE SERVER MAPPING statement:

   - Choose a unique server name.
   - Set NODE to the appropriate server alias from the `hosts` file (excluding the suffix `COPn`). For an example, see Figure 20 on page 156.

     You do not need to use the DataJoiner **CATALOG NODE** command for any data sources that are accessed through the Teradata data access module.
   - Set TYPE to `teradata`.
   - Set VERSION to the version of Teradata that the Teradata server is running. For example, use `2.0.1` for Teradata 2.0.1 (only one-digit numbers and two-digit numbers separated by a decimal point are supported).
   - Set PROTOCOL to `"teradata"`. (See Figure 20 on page 156). This value is case-sensitive.

   A sample CREATE SERVER MAPPING statement for Teradata is:

   ```
   CREATE SERVER MAPPING FROM tdat TO NODE "tdatsvr"
   TYPE teradata VERSION 2.0.1 PROTOCOL "teradata"
   ```

   Preserving case-sensitivity:

   The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

   - From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
   - From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

6. Use additional DataJoiner DDL statements to refine access to the data sources that you defined with the CREATE SERVER MAPPING statement. Other steps (adding server options, creating user mappings, creating nicknames) are not required at this time, but they are suggested. See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about DDL statements and when to use them.
7. If you want the Teradata data access module to be loaded automatically when the DataJoiner instance is started, update the db2profile of the DataJoiner instance to include an entry for the DJXCOMM environment variable.

   The name of the Teradata data access module that you created should be specified in the DJXCOMM variable. For example, if you are using the data access module that is named `teradata`, the entry would be:

   ```
   export DJXCOMM=teradata
   ```

   Ensure that there are no spaces on either side of the equal sign (=).

   See "Step 3: Set Environment Variables" on page 58 for more information.
8. Invoke **db2profile** through either of the following methods:
   - From the DataJoiner instance login, run `.profile`. For example:

     ```
     . .profile
     ```
   - Invoke **db2profile** directly. For example:

     ```
     . /home/djinst1/sqllib/db2profile
     ```
9. Recycle (**db2stop** followed by **db2start**) the DataJoiner instance.
10. Optional: Test the connections to your data sources. This step is recommended because it allows you to determine if your data source is configured correctly.
    a. If the user ID and password used to access the Teradata data source are different from the user ID and password used at the DataJoiner instance, you must create a user mapping as mentioned in step 6 on page 158, for example:

       ```
       CREATE USER MAPPING FROM USER TO SERVER TDAT AUTHID WILSON
       PASSWORD SECRET
       ```

       See the *DataJoiner Application Programing and SQL Reference* for more information.

    b. Connect to the DataJoiner database that you created in "Step 6: Create a DataJoiner Database" on page 62.

    c. Either create a nickname for a table at a Teradata server, or use pass-through to directly access a Teradata table.

Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Default Type Mappings from Teradata Data Sources to DataJoiner

This section shows default forward mappings between DB2 for CS data types defined to DataJoiner and Teradata data types. In these mappings, the REMOTE_TYPENAMEs defined in the Teradata database are mapped to the corresponding TYPENAMEs in the DataJoiner database.

Table 7. Teradata Default Data Type Mappings in SYSCAT.SERVER_DATATYPES (Not All Columns Shown)

REMOTE_TYPENAME	REMOTE_META_TYPE	REMOTE_BIT_DATA	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_S_OPR_P	TYPENAME	BIT_DATA	LENGTH	SCALE
BYTEINT	S	-	-	-	-	-	-	SMALLINT	Y	2	-
SMALLINT	S	N	-	-	-	-	-	SMALLINT	Y	2	-
INTEGER	S	N	-	-	-	-	-	INTEGER	Y	4	-
DECIMAL	S	N	-	-	-	-	-	DECIMAL	Y	-	-
FLOAT	S	N	-	-	-	-	-	DOUBLE	Y	8	-
CHAR	S	N	1	254	-	-	-	CHARACTER	N	-	-
CHAR	S	N	255	4000	-	-	-	VARCHAR	N	-	-
CHAR	S	N	4001	32000	-	-	-	LONG VARCHAR	N	32700	-
VARCHAR	S	N	1	4000	-	-	-	VARCHAR	N	-	-
VARCHAR	S	N	4001	32000	-	-	-	LONG VARCHAR	N	32700	-
BYTE	S	N	1	254	-	-	-	CHARACTER	Y	-	-
BYTE	S	N	255	4000	-	-	-	VARCHAR	Y	-	-
BYTE	S	N	4001	32000	-	-	-	LONG VARCHAR	Y	32700	-
VARBYTE	S	N	1	32000	-	-	-	VARCHAR	Y	-	-

Table 7. Teradata Default Data Type Mappings in SYSCAT.SERVER_DATATYPES (Not All Columns Shown) (continued)

REMOTE_TYPENAME	REMOTE_META_TYPE	REMOTE_BIT_DATA	REMOTE_LOWER_LEN	REMOTE_UPPER_LEN	REMOTE_LOWER_SCALE	REMOTE_UPPER_SCALE	REMOTE_S_OPR_P	TYPENAME	BIT_DATA	LENGTH	SCALE
VARBYTE	S	N	4001	32000	-	-	-	LONG VARCHAR	Y	32700	-
DATE	S	N	-	-	-	-	-	DATE	Y	4	-
GRAPHIC	S	N	1	127	-	-	-	GRAPHIC	N	-	-
GRAPHIC	S	N	128	2000	-	-	-	VARGRAPHIC	Y	-	-
GRAPHIC	S	N	2001	16000	-	-	-	LONG VARGRAPHIC	Y	16350	-
VARGRAPHIC	S	N	1	2000	-	-	-	VARGRAPHIC	Y	-	-
VARGRAPHIC	S	N	2001	16000	-	-	-	LONG VARGRAPHIC	Y	16350	-

Default Type Mappings from DataJoiner to Teradata Data Sources

This section shows default reverse mappings between DB2 for CS data types defined to DataJoiner and Teradata data types. In these mappings, the TYPENAMEs defined to DataJoiner create the corresponding REMOTE_TYPENAMEs defined in the Teradata database.

Table 8. Teradata Default Data Type Mappings in SYSCAT.REVTYPEMAPPINGS (Not All Columns Shown)

REMOTE_TYPENAME	REMOTE_META_TYPE	REMOTE_BIT_DATA	REMOTE_LENGTH	REMOTE_SCALE	TYPENAME	BIT_DATA	LOCAL_LOWER_LEN	LOCAL_UPPER_LEN	LOCAL_LOWER_SCALE	LOCAL_UPPER_SCALE	LOCAL_S_OPR_P
SMALLINT	S	N	-	-	SMALLINT	Y	-	2	-	-	-
INTEGER	S	N	-	-	INTEGER	Y	-	4	-	-	-
DECIMAL	S	N	-	-	DECIMAL	Y	1	18	0	18	-
FLOAT	S	N	-	-	DECIMAL	Y	19	31	0	31	-
FLOAT	S	N	-	-	DOUBLE	Y	-	8	-	-	-
CHAR	S	N	-	-	CHARACTER	N	-	-	-	-	-
VARCHAR	S	N	-	-	VARCHAR	N	-	-	-	-	-
LONG VARCHAR	S	N	-	-	LONG VARCHAR	N	-	32700	-	-	-
BYTE	S	N	-	-	CHARACTER	Y	-	-	-	-	-
VARBYTE	S	N	-	-	VARCHAR	Y	-	-	-	-	-
VARBYTE	S	N	32000	-	LONG VARCHAR	Y	-	32700	-	-	-
DATE	S	N	-	-	DATE	Y	-	4	-	-	-
FLOAT	S	N	-	-	TIME	Y	-	3	-	-	-
DECIMAL	S	N	18	4	TIMESTAMP	Y	-	10	-	-	-
GRAPHIC	S	N	-	-	GRAPHIC	N	-	-	-	-	-
VARGRAPHIC	S	N	-	-	VARGRAPHIC	Y	-	-	-	-	-
LONG VARGRAPHIC	S	N	-	-	LONG VARGRAPHIC	Y	-	16350	-	-	-

Chapter 15. Accessing IMS and VSAM Data Sources Using Classic Connect

Classic Connect is a separately-orderable component of DataJoiner that provides read access to nonrelational data stored in Information Management Systems (IMS) databases and Virtual Storage Access Method (VSAM) data sets.

For information about configuring DataJoiner and Classic Connect to access IMS and VSAM data, see the *DB2 DataJoiner Classic Connect Installation, Configuration, and Reference Guide* V2.1.1.

Chapter 16. Accessing Data Sources through CrossAccess

DataJoiner can access IMS, VSAM, or other data sources that are supported by CrossAccess (DataJoiner can also access IMS and VSAM data through Classic Connect, a separately orderable component of DataJoiner; refer to the *DataJoiner Classic Connect Installation, Configuration, and Reference Guide* for more information). This section explains how to configure DataJoiner to access IMS and VSAM through CrossAccess. Examples are specific for CrossAccess Version 2 Release 1. Refer to the CrossAccess documentation for detailed information on configuring CrossAccess for all data sources.

With DataJoiner already installed, follow these steps to configure a CrossAccess data access module. These instructions reference Figure 21 on page 169.

1. Install and configure the CrossAccess AIX components on the same system where DataJoiner is installed. After configuration is complete, note the path name of the CrossAccess dynamic link library directory and the name (complete path and filename) of the CrossAccess application configuration file. For example, if you modified the sample application configuration file that came with CrossAccess and plan to use it with DataJoiner, the Version 2 names you need to note are:

   ```
   /usr/CrossAccess/UAV2R1M00/lib
   /usr/CrossAccess/UAV2R1M00/samples/cxa-run.cfg
   ```

   The first name is the CrossAccess dynamic link directory; the second is the sample application configuration file.
2. Before using CrossAccess with DataJoiner, use CrossAccess's sample programs to test your configuration to CrossAccess. See the CrossAccess publications for information about running CrossAccess applications.
3. If you have not already done so, build a CrossAccess data access module by running `djxlink.sh`, as described in "Step 3: Link DataJoiner to Data Source Client Software" on page 46.

   For DataJoiner to send requests to CrossAccess, it needs a data access module, which essentially is a file that has been built by link-editing DataJoiner libraries with CrossAccess libraries.

   After running `djxlink.sh`, you will receive a message indicating whether or not the CrossAccess data access module was successfully built. If the data access module could not be built, you will need to edit the sample makefile named `/usr/lpp/djx_02_01_01/lib/djxlink.makefile` so that the makefile works with your particular setup. Editing and running `djxlink.makefile` are also described in "Step 3: Link DataJoiner to Data Source Client Software" on page 46.

If djxlink.sh fails to build a CrossAccess data access module, login as root and edit djxlink.makefile. Find the section of the file entitled ″Accessing Data Sources Through CrossAccess.″

a. ″Xaccess″ is the name of the CrossAccess data access module that is built by the makefile. If you do not want to change this name, skip to the next step.

   If you want to change the name, you need to change two lines in the makefile:

   - Change the target name on the dependency line. For example, if you want the data access module to be named “cxav1r2”, change the dependency line from this:

     ```
     Xaccess: djxxaccess.a Xaccess.exp Xaccess.imp
     ```

     to this:

     ```
     cxav1r2: djxxaccess.a Xaccess.exp Xaccess.imp
     ```

   - Change the output name of the link-edit statement. For example, change this:

     ```
     ld -o Xaccess \
     ```

     to this:

     ```
     ld -o cxav1r2 \
     ```

     This name will be used in the SERVER_PROTOCOL column of SYSCAT.SERVERS in a later step. The SERVER_PROTOCOL column has a restriction that the filename cannot be an existing data access module name unless you are replacing it.

b. If the CrossAccess dynamic link library directory on your system is not the same as the one specified with the -L option in the makefile, replace the -L path with your own. For example, if your path is /usr/lpp/CrossAccess/lib, change this:

   ```
   -L/usr/CrossAccess/UAV2R1M00/lib
   ```

   to this:

   ```
   -L/usr/lpp/CrossAccess/lib
   ```

c. Ensure that the reference to the apif.imp includes the CrossAccess directory where the file is located. This will ensure that the proper level is referenced during the link-edit.

   ```
   -bI:/usr/CrossAccess/UAV2R1M00/samples/apif.imp
   ```

d. As root, link-edit the CrossAccess data access module library with the CrossAccess client libraries, using the supplied makefile.

   Ensure that you are in the DataJoiner “lib” directory and execute the makefile as follows:

```
cd /usr/lpp/djx_02_01_01/lib
make -f djxlink.makefile Xaccess
```

If you changed the name of the data access module in the makefile, you need to use that name with the **make** command. For example:

```
make -f djxlink.makefile cxav1r2
```

Whenever you rerun djxlink.makefile for the CrossAccess data access module, you must recycle all DataJoiner instances that use the CrossAccess data access module.

4. For each DataJoiner instance that needs to work with CrossAccess, set the CrossAccess CXA_CONFIG environment variable in the DataJoiner instance's `.profile` file. Set it to the name of the CrossAccess application configuration file you made when you installed and configured CrossAccess. Using the example filename from Step 1, add the following line to each applicable DataJoiner instance `.profile` file:

   ```
   export CXA_CONFIG=/usr/CrossAccess/UAV2R1M00/samples/cxa-run.cfg
   ```

   Note: If you have more than one CrossAccess instance on a given system, make sure that CXA_CONFIG points to the correct application configuration file for the CrossAccess instance the DataJoiner instance should use.

5. Rerun the DataJoiner instance owner's profile. Then recycle the DataJoiner instance. From the instance owner's home directory:

   ```
   . .profile
   db2stop
   db2start
   ```

6. Update the catalog views by using the CREATE SERVER MAPPING statement to define each IMS or VSAM data source. Each statement corresponds to a CrossAccess SYSLOCATION configuration parameter. When performing server mapping:

   - Choose a unique server name. It is DataJoiner's name for the particular IMS or VSAM data source. In this example `XACCESS1` and `XACCESS2` will be used for the VSAM and IMS data sources respectively.
   - Set the NODE to the location name in the CrossAccess SYSLOCATION configuration parameter. In this example, `SFIMS` and `SFVSAM` are used for the IMS and VSAM data sources respectively. This value is case sensitive.
   - Always use `xaccess` or `crossaccess` for the TYPE.
   - Set VERSION to the version of CrossAccess, `2.1`.
   - The unique filename specified in Step 3, `Xaccess`, is used as the PROTOCOL. This value is case sensitive

A DATABASE value is not required; neither IMS nor VSAM has multiple databases per instance to which you must connect to access data.

A sample entry for an IMS data source is:

```
CREATE SERVER MAPPING FROM imsprod TO NODE "sfims"
TYPE xaccess VERSION 2.1 PROTOCOL "Xaccess"
```

Preserving case-sensitivity:

The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

If desired, perform user mapping. Specify a remote authid and password for any of the data sources by using the CREATE USER MAPPING statement. This is only necessary to send values that differ from the DataJoiner client userid.

After updating the catalog views, all users must reconnect to the DataJoiner database to access the new servers.

7. When creating nicknames for tables to be accessed in VSAM or IMS, use the following values for the remote-object-name:
   - The *data-source-name* corresponds to the SERVER column in the entry in SYSCAT.SERVERS that is specified in Step 6.
   - The **authorization-name** and the *remote-table-name* correspond to the table name parameter in the CrossAccess USE TABLE statement.
8. Test the connections to your data sources. This step is optional but recommended because it allows you to determine if your data source is configured correctly.

   Prerequisite step:

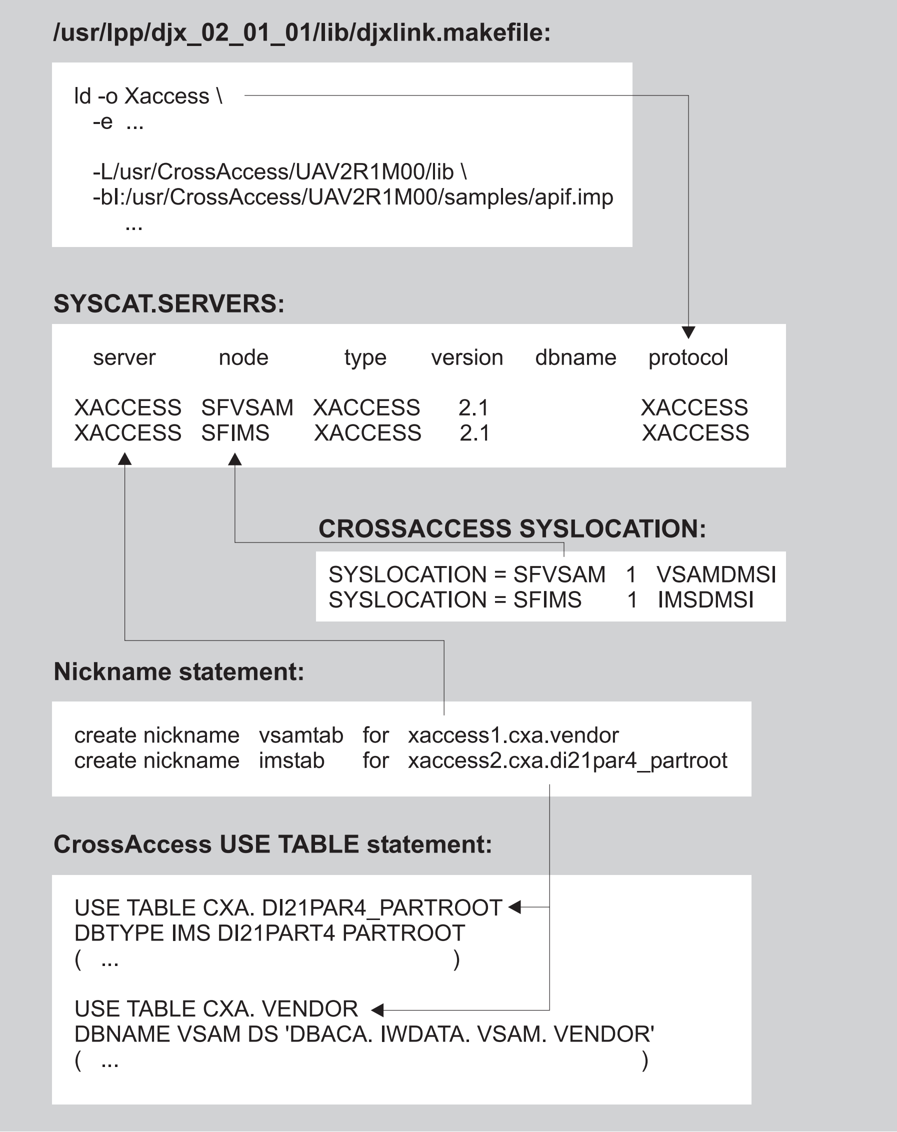
   Before testing, you must perform user mapping as mentioned in 6 on page 167. See the *DataJoiner Application Programing and SQL Reference* for more information.

   To test your location transparency connections through DataJoiner, complete the following tasks:

   a. Connect to the DataJoiner database.
   b. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Figure 21 shows the configuration for the CrossAccess data access module configuration within DataJoiner.



/usr/lpp/djx_02_01_01/lib/djxlink.makefile:

```
ld -o Xaccess \
  -e  ...

  -L/usr/CrossAccess/UAV2R1M00/lib \
  -bI:/usr/CrossAccess/UAV2R1M00/samples/apif.imp
     ...
```

SYSCAT.SERVERS:

```
  server      node       type     version   dbname    protocol

XACCESS   SFVSAM   XACCESS     2.1                   XACCESS
XACCESS   SFIMS    XACCESS     2.1                   XACCESS
```

CROSSACCESS SYSLOCATION:

```
SYSLOCATION = SFVSAM   1   VSAMDMSI
SYSLOCATION = SFIMS    1   IMSDMSI
```

Nickname statement:

```
create nickname   vsamtab   for   xaccess1.cxa.vendor
create nickname   imstab    for   xaccess2.cxa.di21par4_partroot
```

CrossAccess USE TABLE statement:

```
USE TABLE CXA. DI21PAR4_PARTROOT
DBTYPE IMS DI21PART4 PARTROOT
(   ...                                      )

USE TABLE CXA. VENDOR
DBNAME VSAM DS 'DBACA. IWDATA. VSAM. VENDOR'
(   ...                                                        )
```

Figure 21. System Catalog Tables

Chapter 17. Accessing Data Sources Using a Generic Data Access Module

Each data source that uses the generic access API appears to be a DBMS to DataJoiner, whether it is actually a DBMS or not. See the *DataJoiner Generic Access API Reference* for information on generic data access modules.

Restriction:

> The generic access API supports ODBC and X/Open CLI compliant drivers. If the driver you are using does not meet ODBC or X/Open CLI specifications for API calls, SQL dialect, and SQLSTATEs, the generic access data access module might not function properly.

With DataJoiner installed, complete the following steps:

1. Install the code that contains the ODBC or X/OPEN-compliant driver on the DataJoiner system.
2. As root, edit the supplied makefile (`/usr/lpp/djx_02_01_01/lib/djxlink.makefile`) to specify the library directory of the driver you installed in Step 1 and a unique file name for the output of the link-edit. The unique file name will be used in the SERVER_PROTOCOL column of SYSCAT.SERVERS. The filename cannot be an existing data access module name unless you are replacing it.

   To link-edit the custom data access module, you need:

   - The custom data access module library that is provided with DataJoiner
   - The custom data access module exports file that is provided with DataJoiner
   - The custom data access module imports file that is provided with DataJoiner
   - The CLI libraries that are provided by CLI vendor

   You need to perform the link-edit for each CLI driver (each distinct access protocol) you want to use. For example, if you have an X/OPEN-compliant driver to get to an RDB, you need to perform the link-edit once regardless of the number of RDBs you want to access. If you have both an X/OPEN-compliant driver to get to the RDB and a different X/OPEN-compliant driver to get to Supra, you need to link-edit twice—once for the RDB driver and once for the Supra driver. You can then configure access to as many RDB or Supra systems as you want without additional link-editing.

To execute the makefile, enter:

```
cd /usr/lpp/djx_02_01_01/lib
make -f djxlink.makefile Generic
```

Whenever you rerun djxlink.makefile for the generic data access module, you must recycle all DataJoiner instances that use the generic data access module.

Complete the remaining steps for each DataJoiner instance that will use the generic data access module.

3. If your driver needs specific environment variables set, identify the variables to a given DataJoiner instance before you use the driver with that instance. To accomplish this task, you must add an entry to the sqllib/.envfile of the instance. Each entry in sqllib/.envfile, except the last entry, has the following format:

   ```
   variable_name+
   ```

   Do not put the plus sign (+) on the last entry. For example, the contents of the sqllib/.envfile might look like this:

   ```
   HOME+
   SYBASE+
   CXA_CONFIG+
   ORACLE_HOME+
   TNS_ADMIN
   ```

   If you have a driver that needs an environment variable different from these, such as QE_ODBC_INI, you can add it. The file might look like this:

   ```
   HOME+
   SYBASE+
   CXA_CONFIG+
   ORACLE_HOME+
   TNS_ADMIN+
   QE_ODBC_INI
   ```
4. Perform any configuration that is required by the data access module provider. This task might include defining communication paths or database aliases. See the documentation provided by the data access module provider.
5. Recycle (**db2stop**, then **db2start**) the DataJoiner instance.
6. Issue a CREATE SERVER MAPPING DDL statement for each data source you added in step 2 on page 171.
   - Choose a unique server name.
   - Set NODE to the ODBC data source name. This value is case sensitive.
   - Set DATABASE to the name of a database at the server that is listed in the NODE column. This value is case sensitive.

- For TYPE, specify `generic`.
- For VERSION, specify `2.1`.
- For PROTOCOL, specify `generic`. This value is case sensitive.

A sample entry for a generic data source looks like this:

```
CREATE SERVER MAPPING FROM bradgn1 TO NODE "odbcds1" DATABASE "test_db"
TYPE generic VERSION 2.1 PROTOCOL "generic"
```

Preserving case-sensitivity:
The syntax required to preserve case-sensitivity depends on where you enter the CREATE SERVER MAPPING statement.

- From the DB2 interactive mode command prompt, you need to enclose all case-sensitive parameters in double quotes.
- From a UNIX operating system command prompt, you need to enclose all case-sensitive parameters in double quotes, and you need to enclose the entire CREATE SERVER MAPPING statement in single quotes.

7. If you want the generic data access module to start dynamically when the DataJoiner instance is started, update the db2profile of the DataJoiner instance to include an entry for the DJXCOMM variable. For example:

   ```
   export DJXCOMM=generic
   ```

   Ensure that there are no spaces on either side of the equal sign (=).

   See “Step 3: Set Environment Variables” on page 58 for more information.

8. Use additional DataJoiner DDL statements to refine access to the data sources you defined using CREATE SERVER MAPPING statements. Other steps, such as adding server options and creating a user mapping, are not required at this time, but they are suggested. See the *DataJoiner Application Programming and SQL Reference Supplement* and the *DataJoiner Administration Supplement* for more information about DDL statements and when to use them.

   Sample DDL statements are:

   ```
   CREATE SERVER OPTION colseq FOR SERVER bradgn1
   SETTING 'I'
   ```

   ```
   CREATE USER MAPPING FROM brad TO SERVER bradgn1
   AUTHID "bhauth1" PASSWORD "power2u"
   ```

9. Create nicknames. For example:

   ```
   CREATE NICKNAME dept FOR bradgn1.bhauth1.dept
   ```

   See the *DataJoiner Application Programming and SQL Reference Supplement* for more information about the CREATE NICKNAME statement.

10. Test the connections to your data sources. This step is optional but recommended because it allows you to determine if your data source is configured correctly.

    Prerequisite step:

    Before testing, you must perform user mapping as mentioned in step 8 on page 173.

    To test your location transparency connections through DataJoiner, complete the following tasks:

    a. Connect to the DataJoiner database.
    b. Either create a nickname with DataJoiner for a table at a data source, or use pass-through to directly access a table at a data source.

    Refer to the *DataJoiner Application Programming and SQL Reference Supplement* for more information about creating nicknames and using pass-through.

Part 4. Configuring DataJoiner as a DRDA AS

Chapter 18. Configuring the DataJoiner DRDA Application Server and Clients

This chapter describes how to configure DataJoiner for AIX as a DRDA Application Server (DRDA AS) and points you to the relevant resource for configuring clients as DRDA Application Requesters (DRDA ARs). A sample on how to configure a DB2 for MVS DRDA AR to DataJoiner is provided in "DB2 for MVS DRDA AR Sample" on page 186.

APPC Support

DataJoiner DRDA Application Server requires APPC support, which DataJoiner provides through the SNA Clients option (`djx_02_01_01.cs.sna`) during DataJoiner installation. To use APPC, IBM eNetwork Communications Server for AIX Version 5 must be installed and configured on the DataJoiner server workstation.

Configuring DataJoiner as a DRDA AS

To configure DataJoiner as a DRDA AS:

Step 1. Obtain information for configuring DataJoiner to receive connections from DRDA Application Requestors.

Step 2. Configure eNetwork Communications Server for AIX Version 5.

Step 3. Configure the DataJoiner Instance to Receive Connections from SNA Clients

Step 4. Restart the DataJoiner instance.

Step 5. Configure the DRDA Application Requestor to connect to DataJoiner.

Step 6. Bind the DRDA AR applications to the DataJoiner database.

Step 7. Connect to the DataJoiner database from the DRDA Application Requestor.

Much of the eNetwork Communications Server for AIX Version 5 definitions and input information that need to be entered for DataJoiner Version 2 on AIX to act as a DRDA Application Server are also needed for DataJoiner to connect to DRDA Application Server. Therefore, many of definitions that are created by using "Configuring IBM eNetwork Communication Server on AIX for DRDA Data Sources" on page 83 are re-usable. Please refer to these instructions as necessary.

The following definitions in eNetwork Communications Server for AIX Version 5 are used by DataJoiner as a DRDA AR and DataJoiner as a DRDA AS:

- Node Definition
- Port Definition
- Mode Definition

The following definitions in eNetwork Communications Server for AIX Version 5 are used only by DataJoiner as a DRDA AR:

- CPIC Side definition

The following definitions in eNetwork Communications Server for AIX Version 5 are used only by DataJoiner as a DRDA AS:

- Transaction Program definition
- define_trusted_groups

The following definitions in eNetwork Communications Server for AIX Version 5 are used only by DataJoiner as a DRDA AS to receive connections from a specific DRDA AR:

- Link Station definition
- Partner LU definition

You will not need the following definition for DataJoiner to receive connections from DRDA ARs:

- CPIC Symbollic Destination definition

If a CPIC Symbollic Destination definition exists for DataJoiner to connect to the DRDA server that you now want to configure as the DRDA Application Requestor, there is no conflict. You do not need to remove the existing CPIC Symbollic Destination.

Step 1: Obtain Information for Configuring DataJoiner to Receive Connections from DRDA Application Requestors

You will need the following information for DataJoiner to be either DRDA AR or DRDA AS:

SNA network name
See page 84 for explanation.

SNA control point name
See page 84 for explanation.

LU name of the DRDA application requestor
Same as if the DRDA AR were a DRDA server. See page 85 for explanation.

MAC address of the DRDA AR token-ring adapter
Same as if the DRDA AR were a DRDA server. See page 85.

Local LU name for the DataJoiner workstation
See page 85 for explanation.

Node ID for the DataJoiner workstation
See page 85 for explanation.

Modename IBMRDB
See page 85 for explanation.

Transaction program name for the DataJoiner instance

TPNAME is a Database Manager Configuration parameter for the DataJoiner instance. For inbound APPC connections, TPNAME differentiates one DataJoiner instance from other DataJoiner and DB2 instances on the same AIX system. For more information about the TPNAME parameter, see the GET DATABASE MANAGER CONFIGURATION command in the *DATABASE 2 Command Reference* and the configuring chapter in the *DATABASE 2 Administration Guide*.

To see if TPNAME is already configured, login to AIX as the DataJoiner instance owner and enter:

```
db2 get dbm cfg
```

If TPNAME is not already configured, you can pick a value. It will be used:

- In your DataJoiner Instance’s Database Manager Configuration
- In your eNetwork Communication Server for AIX Transaction Program Definition
- In your DRDA Application Requestor definitions to connect to DataJoiner

DataJoiner database’s alias name
A database must already be created in the DataJoiner instance. If its alias is different than the database name, use the alias name when configuring communications from the DRDA Application Requestor. To obtain the alias name for the DataJoiner database, login to AIX as the DataJoiner instance owner and enter:

```
db2 list db directory
```

The alias name will be indicated in each of the Database Directory entries. For instance, for a DataJoiner database whose alias name is DJDB1AL, the entry will look like this:

```
Databaes alias = DJDB1AL
Database name = DJDB1
Local database directory = /home/dj1
Database release level = 6.01
Directory entry type = Indirect
```

DataJoiner instance owner user name
AIX user name of the DataJoiner instance.

DataJoiner instance owner's primary group name
The AIX primary group for the DataJoiner instance owner. To learn the value, login to AIX as the DataJoiner instance owner and enter the AIX command **id**. In the response, the name in parentheses that follows `gid=` is the DataJoiner instance owner's group name.

DataJoiner instance owner's AIX group number
Number of the primary AIX group for the DataJoiner instance owner. To learn the value, login to AIX as the DataJoiner instance owner and enter AIX command **id**. In the response, the number that follows `gid=` is the DataJoiner instance owner's group name.

Step 2: Configure eNetwork Communications Server for AIX Version 5

The following assumptions are made:

- IBM eNetwork Communications Server Version 5 for AIX is installed.
- DataJoiner is installed.
- The user is logged on as "root".
- A token-ring network is the communications medium.

If Communications Server has not yet been configured for DataJoiner to access DRDA servers, complete the following instructions by using **xnaadmin** from "Configuring IBM eNetwork Communication Server on AIX for DRDA Data Sources" on page 83:

1. Define a Node. See step 2 on page 86
2. Define a Port. See step 3 on page 86
3. Define a Mode. See step 7 on page 88
4. Define a Transaction Program for the DataJoiner instance by using **xsnaadmin**.
   a. Select **Services –> APPC –> Transaction Programs**. The Transaction Programs window opens.
   b. Select **Add+**. The TP invocation window opens.
   c. In the **TP name** box select **Application TP** and type the transaction program name for the DataJoiner instance
   d. In the **LU** box select **Parameters are for invocation on a specific LU** and type the control point alias value from the node defintion parameters (see step 1)
   e. In the **TP invocation** box:
      - Select the **Queue incoming allocates** check box.

    - In the **Full path to TP executable** field, type the full path and file name to DataJoiner's APPC transaction program. This file name is `db2acntp`. It will be found for execution in the `sqllib/bin` subdirectory of the DataJoiner instance owner's home directory. For example, if the DataJoiner instance owner's home directory is `/home/dj1`, type:

      ```
      /home/dj1/sqllib/bin/db2acntp
      ```

    - In the **User ID** field, type the DataJoiner instance owner's ID.
    - In the **Group ID** field, type the DataJoiner instance owner's primary group name.

   f. Click **OK**.

   g. Close the Transaction Programs window.

5. Define a Trusted Group for the DataJoiner instance's Transaction Program. At the AIX command prompt, enter the following **snaadmin** command:

   ```
   snaadmin define_trusted_groups, group_id=DataJoiner_instance_owner's_AIX_Group_Numbe
   ```

   For example, if the DataJoiner instance owner's AIX Group Number is 201, you would enter:

   ```
   snaadmin define_trusted_groups, group_id=201
   ```

   The response should be:

   ```
   define_trusted_groups command successfully
   ```

   Alternatively, you can also create a file to hold the input to the **snaadmin** command. Use an AIX editor (such as vi) to create a file with contents like this:

   ```
   [define_trusted_groups]
   group_id=201
   ```

   Save and close the file and enter the following command:

   ```
   snaadmin -i filename
   ```

   For more information on **snaadmin** commands, see the *IBM eNetwork Communications Server for AIX Administration Command Reference*

If the system that you want to connect to DataJoiner is not configured for DataJoiner to access it as a DRDA server, return to **xsnaadmin** and complete following steps:

1. Define a link station. See step 4 on page 87.
2. Define a Partner LU over the link station. See step 5 on page 87.

Step 3: Configure the DataJoiner Instance to Receive Connections from SNA Clients

Login to AIX with the DataJoiner instance owner's ID and do the following steps:

1. Update the Database Manager Configuration with the Transaction Program Name for the DataJoiner instance. In the following example, the TPNAME is DJ1TP:

   ```
   db2 update dbm cfg using tpname "DJ1TP"
   ```

   For case-sensitive parameters:
   You must enclose all case-sensitive parameters in double quotes.

2. Update the DataJoiner instance owner's environment variable DB2COMM to APPC.

   a. Determine if DB2COMM is already specified:

      ```
      echo $DB2COMM
      ```

   b. Add APPC:

      - If DB2COMM is already defined, and APPC is not already included, the response to `echo $DB2COMM` will probably be `DB2COMM=TCPIP`. You need to add APPC.
      - If DB2COMM is not already defined, the response to `echo $DB2COMM` will be a blank line. You need to define DB2COMM. See "Step 3: Set Environment Variables" on page 58.
      - In the file that you use to define DB2 environment variables for the DataJoiner instance (probably in `/sqllib/db2profile`), DB2COMM should look like either:
        - `DB2COMM=APPC` to support just SNA APPC inbound connections.

        or

        - `DB2COMM=APPC,TCPIP` to support both SNA APPC and TCP/IP inbound connections.

After you update your `.profile` or `db2profile` file to add `DB2COMM=APPC`, be sure to re-run `.profile` to set the variable and put it into effect.

Step 4: Restart the DataJoiner Instance

To put the Database Manager Configuration TPNAME parameter into effect and make DataJoiner aware that DB2COMM includes APPC, restart the DataJoiner instance (**db2stop** then **db2start**). Several APPC Listener processes will be started.

Step 5: Configure the DRDA Application Requestor to Connect to DataJoiner

You configure the system that you want to connect to DataJoiner in the same way that you would configure it to connect it to any DB2 using DRDA over SNA APPC. The instructions here are for configuring a DB2 for OS/390 subsystem to connect to DataJoiner.

1. Make sure that the same SNA definitions are in place that would be used for DataJoiner to try to connect to the DRDA AR as a DRDA server.
2. Update the DB2 communications database for connections to the DataJoiner database As a minimum, records need to be inserted into SYSIBM.LOCATIONS, SYSIBM.LUNAMES, and SYSIBM.USERNAMES. See the *DRDA Connectivity Guide*, *DRDA Certification Guide*, *DB2 for OS/390 Administration Guide*, and *DB2 for OS/390 SQL Reference* for more information:
   a. Insert a record into SYSIBM.LOCATIONS with values for:

      LOCATION:
      DataJoiner database’s alias name

      LINKNAME:
      Local LU name for the DataJoiner workstation

      TPNAME:
      Transaction program name for the DataJoiner instance

      Example:

      ```
      INSERT INTO SYSIBM.LOCATIONS (LOCATION, LINKNAME, TPNAME)
      VALUES ('DJDB1AL','STB8014I','DJ1TP')
      ```

   b. Insert a record into SYSIBM.LUNAMES with values for:

      LUNAME:
      Local LU name for the DataJoiner workstation. Match LINKNAME value in SYSIBM.LOCATIONS.

      SYSMODENAME:
      IBMRDB

      SECURITY_OUT:
      ’P’

      USERNAMES:
      ’O’

      If the DataJoiner instance’s Database Manager Configuration specifies `Authentication=Server`, use ’P’ for SECURITY OUT. If the DataJoiner instance's Database Manager Configuration specifies `Authentication=Client`, see the *DRDA Connectivity Guide* and *DRDA Certification Guide*.

Example:

```
INSERT INTO SYSIBM.LUNAMES (LUNAME,SYSMODENAME,SECURITY_OUT,USERNAMES)
VALUES ('STB8014I','IBMRDB','P','O')
```

c. Insert a record into SYSIBM.USERNAMES with values for:

LINKNAME:
Local LU name for the DataJoiner workstation. Match the LINKNAME value to the SYSIBM.LOCATIONS value.

TYPE: 'O'

AUTHID:
MVS ID to be translated by this record. If AUTHID is not specified, this record will be used by all MVS IDs that do not have a record in USERNAMES with their MVS ID specified in the AUTHID column.

NEWAUTHID:
AIX ID that can connect to DataJoiner and has privileges in DataJoiner database. DataJoiner can translate IDs received in uppercase to lowercase for validation in AIX User Management, but it is better to specify NEWAUTHID as it is (case-sensitive) in AIX.

PASSWORD:
Password for the AIX ID specified in NEWAUTHID. AIX passwords are case-sensitive.

Example: To translate all MVS IDs to the same DataJoiner user ID:

```
INSERT INTO SYSIBM.USERNAMES (LINKNAME,TYPE,NEWAUTHID,PASSWORD)
VALUES ('STB8014I,'O','dj1','dj1pw')
```

Step 6: Bind the DRDA AR Applications to the DataJoiner Database

Before MVS or AS/400 applications can connect to DataJoiner, their packages must be bound to the DataJoiner database. For binding from MVS, see the *DB2 for OS/390 Command Reference*. The **bind** command behaves differently, depending on what precompile options were chosen. However, the bind options supported by DataJoiner are not the same as the bind options supported by DB2 for OS/390 and AS/400. For specifics on bind options that are supported by DataJoiner, see "Bind Options Supported by the DataJoiner DRDA Application Server" on page 298.

Tip:

Another good reference is IBM Redbook *DB2 for MVS Connections with AIX and OS/2* (SG24-4558). The chapter on binding contains:

- A table of bind options that are supported when binding to DB2 CS V2 — which is the same level as DataJoiner Vesion 2.
- An example of binding SPUFI packages and plans to DB2 CS V2.

Step 7: Connect to the DataJoiner Database from the DRDA Application Requestor

AS/400 and DB2 for OS/390 Version 4 and 5 applications must explicitly use the CONNECT SQL statement to establish connections to the DataJoiner database, and must use only one or two part names when referring to tables, views, and nicknames.

For example, when using SPUFI to test your connection to DataJoiner, in the SPUFI main panel, in the **10 CONNECT LOCATION** field, you must specify the DataJoiner database alias name (that is the LOCATION value for the DataJoiner database in the SYSIBM.LOCATIONS table).

For examples of how to write programs that access data via DataJoiner, refer to using three-part names to access DataJoiner tables and nicknames in the *DB2 for OS/390 Application Programming and SQL Guide*. The instructions and examples for programs using DRDA access in SPUFI or in your MVS programs will result in a COMMUNICATIONS ERROR because three-part names invoke the DB2 Private Protocol, which is not supported by DataJoiner.

Exception: DB2 for OS/390 Version 6 will permit tables and nicknames in DataJoiner to be accessed by three-part names.

References:

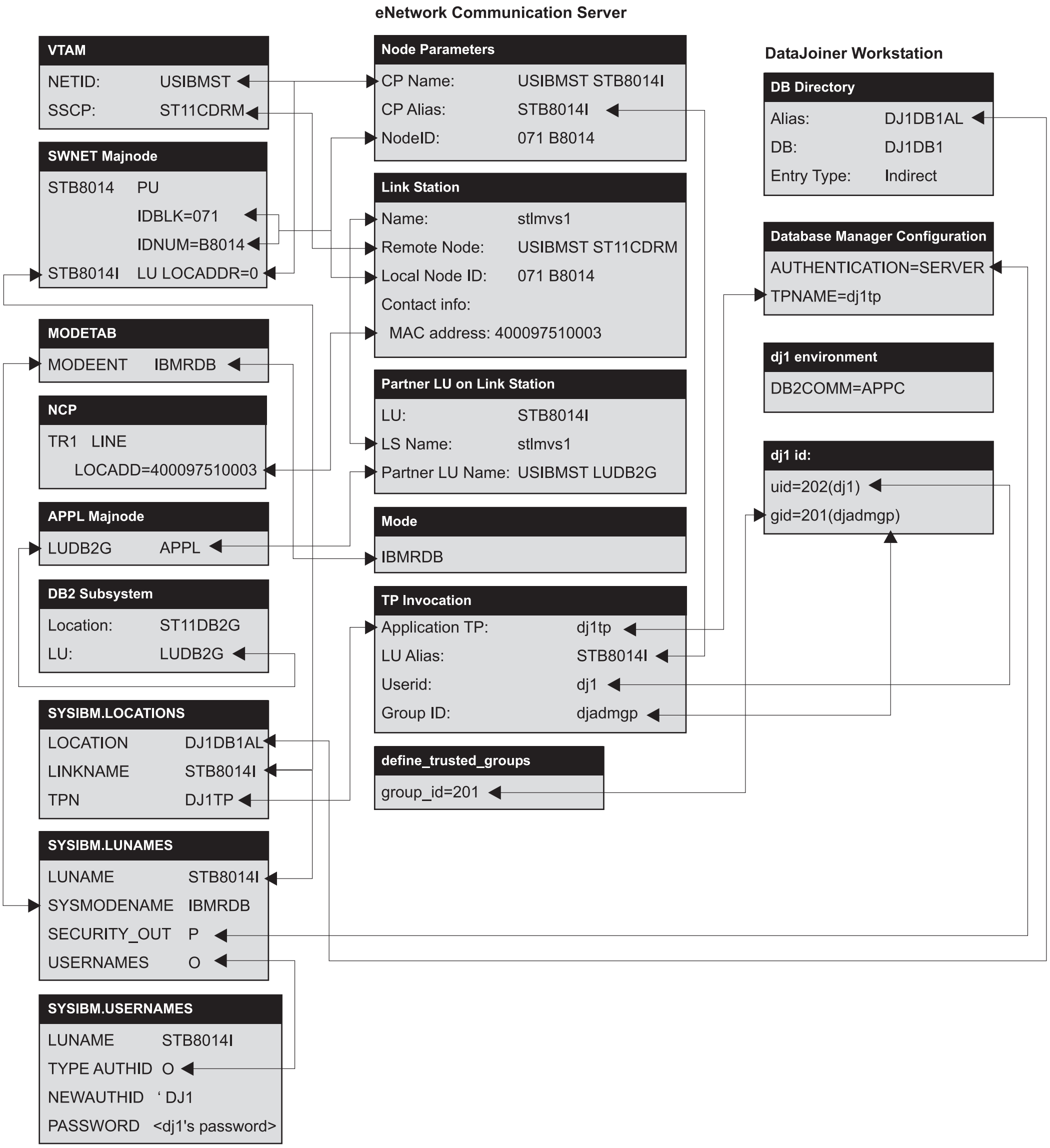
- *DRDA Connectivity Guide*
- *DRDA Certification Guide*
- *DB2 for MVS Connections with AIX and OS/2*
- “Appendix A. DRDA Application Server” on page 297

Configuring Clients as DRDA ARs

To configure clients as DRDA ARs, refer to ″Setting Up the Application Requester″ for your specific client and platform in the *Distributed Relational Database Architecture Connectivity Guide*, (SC26-4783-03).

DB2 for MVS DRDA AR Sample

This section gives an example for configuring a DB2 for MVS DRDA AR to DataJoiner. Figure 22 is provided for your use in defining your own configuration. The following steps take you through the process. For more information, refer to the *DRDA Connectivity Guide*.



VTAM
NETID: USIBMST
SSCP: ST11CDRM

SWNET Majnode
STB8014 PU
IDBLK=071
IDNUM=B8014
STB8014I LU LOCADDR=0

MODETAB
MODEENT IBMRDB

NCP
TR1 LINE
LOCADD=400097510003

APPL Majnode
LUDB2G APPL

DB2 Subsystem
Location: ST11DB2G
LU: LUDB2G

SYSIBM.LOCATIONS
LOCATION DJ1DB1AL
LINKNAME STB8014I
TPN DJ1TP

SYSIBM.LUNAMES
LUNAME STB8014I
SYSMODENAME IBMRDB
SECURITY_OUT P
USERNAMES O

SYSIBM.USERNAMES
LUNAME STB8014I
TYPE AUTHID O
NEWAUTHID ‘ DJ1
PASSWORD <dj1's password>

eNetwork Communication Server

Node Parameters
CP Name: USIBMST STB8014I
CP Alias: STB8014I
NodeID: 071 B8014

Link Station
Name: stlmvs1
Remote Node: USIBMST ST11CDRM
Local Node ID: 071 B8014
Contact info:
MAC address: 400097510003

Partner LU on Link Station
LU: STB8014I
LS Name: stlmvs1
Partner LU Name: USIBMST LUDB2G

Mode
IBMRDB

TP Invocation
Application TP: dj1tp
LU Alias: STB8014I
Userid: dj1
Group ID: djadmgp

define_trusted_groups
group_id=201

DataJoiner Workstation

DB Directory
Alias: DJ1DB1AL
DB: DJ1DB1
Entry Type: Indirect

Database Manager Configuration
AUTHENTICATION=SERVER
TPNAME=dj1tp

dj1 environment
DB2COMM=APPC

dj1 id:
uid=202(dj1)
gid=201(djadmgp)

Figure 22. DB2 for MVS DRDA AR Configuration Template

1. Perform the steps that are described in “Configuring DataJoiner as a DRDA AS” on page 177. It is assumed the VTAM definitions supporting the AIX workstation have already been made.
2. It is also assumed the following has been completed on the DB2 for MVS system:
   - DDF is configured.
   - The DB2 for MVS system has already been defined to VTAM, including the NETID for DB2 and the VTAM APPL definition.
3. Update the DB2 for MVS communications database (CDB) to define each remote DataJoiner DRDA AS. The CDB is described in detail in the *Database 2 for MVS/ESA SQL Reference*, SC26-3270.
   a. Update SYSIBM.SYSLOCATIONS with each DataJoiner database, for example:

      ```
      insert into SYSIBM.SYSLOCATIONS (location,loctype,linkname,
      linkattr) values('dbdb1',' ','STB004I','ZZSERVERTP')
      ```

      location
      The remote database name or alias.

      loctype
      Leave blank.

      linkname
      The workstation’s independent LU as it is defined to VTAM, which must match the LUNAME in SYSLUNAMES and SYSUSERNAMES. Note that it can be the control point name or the LU name that is defined in SNA on the DataJoiner workstation.

      linkattr
      The TPN profile, which must match the DataJoiner tpname in SNA Server, and in the DataJoiner database manager configuration file.

   b. Update SYSIBM.SYSLUNAMES for each DataJoiner instance. The minimun entry that is required is:

      ```
      insert into SYSIBM.SYSLUNAMES(luname,sysmodename,usersecurity,
      encryptpswds,modeselect,usernames) values('STB8004I',' ','A',
      'N','N',' ')
      ```

      The columns of particular interest in an AR configuration are these:

      luname The workstation’s independent LU as it is defined to VTAM, which must match the LINKNAME column of SYSLOCATIONS.

      usersecurity The security acceptance option for *inbound* requests from the remote LU.

- ’A’ – verification already performed (AUTHID required, password optional)
- ’C’ – all conversations need verification (both AUTHID and password are required)

usernames The type of translation performed for IDs

- ’ ’ – No translation, Security Same assumed. Only the AUTHID is sent with an already verified indicator. Use when the DataJoiner database is cataloged with AUTHENTICATION CLIENT.
- ’O’ – Outbound translation, Security Program assumed. Use when theDataJoiner database is cataloged with AUTHENTICATION SERVER or DCS. The AUTHID and Password that are located in SYSUSERNAMES are sent, so ensure that SYSUSERNAMES contains an entry for each MVS user ID that will connect to the DataJoiner database. DB2 for MVS uses SYSUSERNAMES because the MVS logon and password cannot be determined when the CONNECT statement is issued.
- ’B’ – Translation occurs both inbound and outbound.
- ’I’ – Inbound translation. This value is not used in a DB2 for MVS AR configuration.

c. Update SYSIBM.SYSUSERNAMES if USERNAMES has been specified as ’O’ or ’B’ in SYSLUNAMES as described above. All passwords in this table are not encrypted because DB2 for MVS can only accept encrypted passwords from another DB2 for MVS AS. The administrator should thus consider doing two things:

   1) Since passwords are not encrypted, restrict access to this table.

   2) Create a view that will allow each user to update their own password through the view.

d. Update SYSIBM.SYSMODESELECT with the value ’Y’ to associate a mode with an outgoing SQL request.

4. To execute applications from MVS, they must first be bound to the DataJoiner database. Refer to *DB2 for MVS Connections with AIX and OS/2*, (SG24-4558-00) for information about binding applications to DRDA servers.

Part 5. Maintaining DataJoiner

Chapter 19. Maintaining the DataJoiner Environment

This chapter contains information on possible considerations when maintaining a DataJoiner environment.

DB2 and DDCS Coexistence with DataJoiner

DataJoiner instances can coexist on the same workstation as DB2, DB2/6000, DDCS, and DDCS/6000 instances. However, symbolic links that are created by the DB2 and DDCS **db2ln** command affect the DataJoiner application programming environment on the same workstation where DataJoiner is installed.

You can use the **db2ln** command to create symbolic links between DB2 and DDCS product directories and /usr/include and /usr/lib for certain header files, such as .h, .cbl, and .etc, and for libraries, such as libdb2.a. This command is optional; you can use it to compile and link-edit applications without specifying options on the compile and link-edit commands; for example, -I (uppercase i), -L, and -l (lowercase L) on the **cc** command, and -L and -l (lowercase L) on the **ld** command.

If DDCS or DB2 already exists on the same system where DataJoiner is being installed, these symbolic links might already exist. DataJoiner uses some of the same header file and library names as DB2 and DDCS. However, the files are not compatible between DataJoiner and the other products. Therefore, when compiling a DataJoiner application, you must use the -I (uppercase i) option to specify the full path for header files. To identify the correct libraries you can either:

- Use the -L compiler/link-editor options to specify the full path for libraries:
  - On DataJoiner for AIX systems:

    ```
    -L/usr/lpp/djx_02_01_01/lib
    ```

  - On DataJoiner for Solaris systems:

    ```
    -L/opt/IBMdjx/V2.1.1/lib
    ```

  However, this option leaves your applications dependent on that particular version and release of DataJoiner. If you upgrade DataJoiner to a new version or release, the directory of the new version or release will be different from the old version or release, and you must relink-edit all of your applications.
- Specify the full path for the libraries in your LIBPATH environment variable. LIBPATH is used by DataJoiner to locate the required libraries. The search order for libraries is:

1. Directories in LIBPATH (if any)
2. Directories specified using the compile and link-edit time option -L
3. Default libraries (/lib and /usr/lib)

This option does not make your application dependent on a particular version or release of DataJoiner, but it might affect other applications because many programs can use LIBPATH to locate libraries.

Password Considerations for OS/2 Clients

If you plan to have a network that includes OS/2 database clients, specify logins in lowercase and passwords in uppercase on the UNIX platform.

Maintaining DataJoiner

To ensure that your version of DataJoiner remains current, you can install fixes as they are provided, and delete uncommitted options that are not necessary for your system.

Installing DataJoiner Program Temporary Fixes (PTFs)

To install DataJoiner for AIX PTFs:

1. Stop all DataJoiner instances.
2. Apply the PTF by using smitty or installp.
3. Read the file, /usr/lpp/djx_02_01_01/lpp.README, and pay particular attention to section 1.0 ″User Impact.″

   The first two steps can be performed in either order, but they *must* be performed before doing the remaining steps.
4. From the root login, run `/usr/lpp/djx_02_01_01_0000/instance/db2instupdt` for each DataJoiner instance you have. For example:

   ```
   /usr/lpp/djx_02_01_01_0000/instance/db2instupdt djinst1
   ```
5. Restart each DataJoiner instance.
6. Make any necessary changes as described in section 1.0 of the README file.

Deleting Options on AIX 3.2

In AIX 3.2, the procedure you follow to remove an option depends on whether the option is in the applied or committed state.

- If the option is in the applied state, use either the Remove Applied Software Product menu in smitty or the **installp** command with the reject flag. For example:

```
installp -r djx_02_01_01.dd
```

You cannot remove an option if it is a prerequisite for other options that are installed on the system unless you also remove the other options at the same time. To remove all the DataJoiner options for a given version, enter the **installp** command as follows:

```
installp -r djx_02_01_01
```

This command:

- Updates the SWVPD database.
- Deletes all the installed files and directories.
- Undoes all pre-installation and post-installation processing.

The `/usr/lpp/djx_02_01_01` directory is not removed when all the DataJoiner options are deleted. It is, however, emptied.

- If the option is in the committed state, the only way to remove it is to reinstall it using the force option and then to reject it. To perform this procedure:
  1. Login as root.
  2. Execute **installp** to do a force reinstallation that will leave the option in applied state.
  3. Execute **installp** with the reject flag, as described earlier in this section.

Deleting Options on AIX 4.1

On AIX 4.1, options that are successfully installed are always in the committed state. To remove options, use the deinstall (-u) switch. For example:

```
installp -u djx_02_01_01
```

This command removes all product options that match the djx_02_01_01 prefix from the system. You can use the deinstall switch to remove product options that are in the broken state.

Chapter 20. Maintaining Instances

This chapter covers issues that pertain to instances that are already installed.

Starting and Stopping a DataJoiner Instance

You can use DataJoiner or operating system commands to control instances. Use the operating system **kill** command only as a last resort.

Using DataJoiner Commands

You must start a DataJoiner instance before you can connect to a database, precompile an application, or bind a package to a database. The commands to start and stop an instance are, respectively:

db2start
db2stop

When you want to terminate an instance, use the **db2stop** command, and, if necessary, the **force** command to terminate all active applications that connect to the database.

See the *DB2 Command and API Reference* for more information about DataJoiner commands.

Using the kill Command

Use the operating system **kill** command to terminate a DataJoiner instance only if the DataJoiner commands do not work. Using the operating system **kill** command abruptly ends DataJoiner processes without the appropriate termination and clean-up processing. If the **kill** command is used, use the **ipcs** and **ipcrm** commands to identify and remove message queues and some memory sets.

When to Stop and Restart Instances

Certain changes to your system require you to stop and restart (recycle) a given DataJoiner instance before the changes become effective for that instance. These changes are:

- When you modify any environment variable that is used by the instance, such as those listed in “Step 3: Set Environment Variables” on page 58. The environment variables that a DataJoiner instance can use are:
  - COBLIB (for Teradata data sources)

- COPERR (for Teradata data sources)
- CXA_CONFIG (for the Classic Connect interface)
- DJX_ASYNC_APPLY
- DJX_CC2_CONFIG (for the Classic Connect interface)
- DJX_NR_CONFIG
- DJX_NR_START
- DJXCOMM
- DJXODBCTRACE
- HOME
- INFORMIXDIR (for Informix V5 and V7 data sources)
- INFORMIXSERVER (for Informix V7 data sources)
- INFORMIXSQLHOSTS
- LANG
- ORA_NLS (for Oracle data sources)
- ORACLE_HOME (for Oracle data sources)
- QE_ODBC_INI
- SQLRM (for Informix V5 data sources)
- SYBASE (for Sybase data sources)
- TNS_ADMIN (for Oracle data sources)

Some of these variables are specific to a data source. If you do not have the client libraries for a data source link-edited to DataJoiner, the environment variables that are associated with the data source do not affect DataJoiner.

The instance might reference additional environment variables if you set up a Generic data access module as described in "Chapter 17. Accessing Data Sources Using a Generic Data Access Module" on page 171.

- When you modify any of the tuning parameters and the configuration parameters in the instance's database manager configuration file. The database manager configuration file is discussed in the *DB2 Administration Guide*.
- When you modify any of the configuration parameters in the Classic Connect interface configuration file *except* for the following:

  AF TRACE LEVEL
  AF WAIT TIME
  EXTERNAL TASK WAIT TIME
  FETCH BUFFER SIZE
  TASK CONNECT WAIT TIME

Though these parameters do not require that you recycle the instance, users currently accessing Classic Connect must issue the **connect reset** command and reconnect to the database before seeing the change.

The following situations require you to stop and restart all DataJoiner instances:

- When you run djxlink.makefile, and instances already exist. See “Step 3: Link DataJoiner to Data Source Client Software” on page 46 for more information.
- When DataJoiner service is applied.
- When you create a data access module.
- When you update an instance by using the **db2iupdt** command.

Updating Instances

Existing instances are designed to be as independent as possible from the effects of subsequent installation and removal of product options. In general, the installation or removal of product options does not require any changes to existing instances. In most cases, existing instances will automatically inherit or lose access to the function of the option that is being installed or removed. For example, if an option such as a message catalog is installed or removed, the new function is automatically inherited or lost by all existing instances.

However, when the following options:

- djx_02_01_01.db2.rte (DataJoiner Executables)
- djx_02_01_01.cs.rte (DataJoiner Communication Support - Base with TCP/IP)
- djx_02_01_01.dd (Database Director)

are installed or removed, existing instances do not automatically inherit the new system configuration parameters or gain access to all the additional function.

To access new configuration and function, existing instances must be updated using the **db2iupdt** command in `/usr/lpp/djx_02_01_01/instance`.

The db2iupdt script can be invoked as follows:

```
►►──db2iupdt──instanceName──-u──UdfName──────────────────────────────────────────►◄
```

where

instanceName
is the name of the instance to be updated

-u is an optional parameter that specifies the AIX user and group under which Fenced UDFs and DARIs will execute.

Restriction:
UdfName may not be root or bin.

If UdfName is unspecified when you update an instance, one of two actions will occur:

- The user and group names for UDFs will remain unchanged if they were already defined.
- The user and group names for UDFs default to nobody:nobody; this means that there are no file access permissions defined for UDFs or DARIs.

Running the db2iupdt script updates the specified instance by performing the following processing:

1. Replace the files in INSTHOME/sqllib/adm with the files in `/usr/lpp/djx_02_01_01/adm`.
2. If the node type is changed, then a new database manager configuration file is created by merging relevant values from the existing database manager configuration file with the default database manager configuration file for the new node type. If a new database manager configuration file is created, the old file is backed up to `INSTHOME/sqllib/db2systm.bak`.

Starting a DataJoiner Instance at System Boot

To start a DataJoiner instance when the AIX system is booted:

1. Login as root.
2. Use an editor to create a file named `/etc/rc.djx`. Enter the following line for each instance that is to be started at system boot:

   ```
   su - instname "-c db2start > /dev/console 2>&1"
   ```

   where *instname* represents the instance name.
3. Set the permissions on the `/etc/rc.djx` file to 744 by entering:

   ```
   chmod 744 /etc/rc.djx
   ```
4. Add an entry, `/etc/rc.djx`, to the `/etc/rc.djx` file.

Multiple Instances

DataJoiner allows you to run multiple instances on the same operating system. Each instance must have a unique name. For a local client to use a particular instance, you must set an environment variable to the name of the instance, and you must have certain directories in your path. For more information, see “Step 3: Set Environment Variables” on page 58.

Instances are virtual copies of the multidatabase server that are designed to isolate one environment from another. An installation can isolate its production environment from any effects of its test environment by defining a distinct instance for each environment. Use multiple instances when load requirements necessitate multiple DataJoiner servers running on the same machine and sharing the same set of data sources. In such cases, you can configure each instance differently with respect to allocation of system resources. Each such DataJoiner instance is isolated from the failure of other DataJoiner instances.

An instance has its own owner, system administrator, databases, and users. Only the root administrator can create an instance, and only the root administrator or the instance owner can remove an instance. The SYSADM authority is controlled at the instance level; other privileges are controlled at the database and object level. Refer to the *DataJoiner Administration Supplement* for more information. The security mechanisms of each data source control data source access.

There are several of reasons that you might want to create multiple instances of DataJoiner. Some possibilities are:

- To isolate a development and testing environment from production
- To tune the database manager configuration file for a particular database environment
- To restrict sysadm access to certain databases

You can create additional instances by following the instructions in “Step 2: Create a DataJoiner Instance” on page 57. There must be a unique correspondence between an instance and an instance owner. Also, each instance owner must have a unique home directory.

To separate sysadm authority between instances, ensure that each instance owner belongs to a different primary group before creating instances. Otherwise a single user could have sysadm authority over multiple instances. See Figure 23 on page 200.

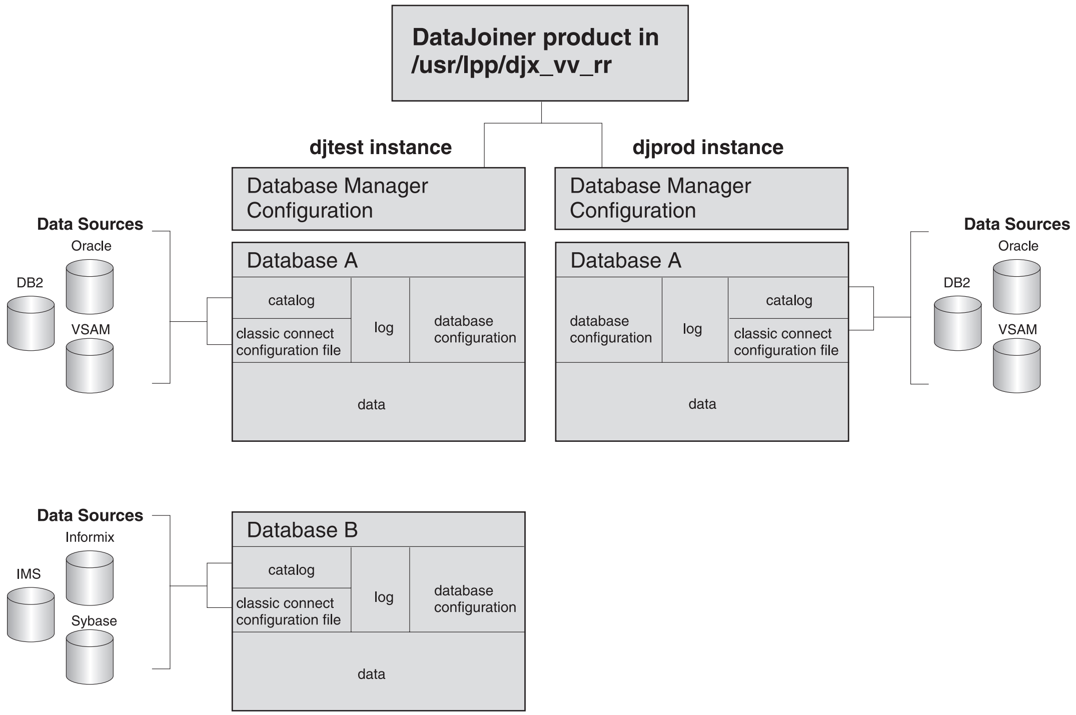


Figure 23. DataJoiner Instances

This figure shows a test instance and a production instance. In this configuration:

- The two instances represent production and test environments that are based on the same release of DataJoiner (02_01_01). They could, however, represent test environments for two different releases of DataJoiner. For example, they could represent a migration from djx_02_01 to djx_02_02 (that is, release 1 to release 2).
- sysadm access is controlled by membership in the primary group that is associated with the djtest and djprod user IDs.
- Database manager configuration affects all databases in a given instance. Service names for remote TCP/IP clients, tpnames for remote APPC clients, and specific tuning parameters for that instance are contained in the Database manager configuration. This configuration separates users who are accessing DataJoiner as remote clients, using the test instance, from users who are using the production instance.
- The test instance has both a database A and B. Database A is for an application in production so it is also in the production instance. This configuration provides a migration path from a test environment to a production environment.

- Each instance has its own log.
- Each instance has its own catalog and Classic Connect configuration file for configuring to data sources that are accessed through DataJoiner. The test and production instances can reference the same data sources or different data sources.
- The test and production instances are separated: djtest can be stopped and started, databases updated, and applications changed without fear of affecting production users.

Removing Instances

A DataJoiner instance that was created following the procedure that is described in “Step 2: Create a DataJoiner Instance” on page 57 can be removed as follows:

1. End all applications that are currently using the instance.
2. Stop the database manager and the command line processor.

   Issue the **db2stop** and **db2 terminate** commands.
3. Backup files in the `INSTHOME/sqllib` directory if needed. Backup candidates include your db2systm file and DARI applications kept in `INSTHOME/sqllib/function`, where `INSTHOME` is the home directory of the instance owner.
4. As root, execute the **db2idrop** command in `/usr/lpp/djx_02_01_01/instance` for AIX systems and `/opt/IBMdjx/V2.1.1/instance` for Solaris systems as follows:

   ```
   db2idrop instanceName
   ```

   where *instanceName* is the name of the instance being dropped.

   The **db2idrop** command removes the instance entry from the list of instances and erases the `INSTHOME/sqllib` directory.
5. As root, remove the instance owner’s user ID and group if used only for that instance. Do not remove these if you are planning to recreate the instance.

To recreate an instance, see “Step 2: Create a DataJoiner Instance” on page 57.

Authorizing Access within an Instance

DataJoiner provides optional group level authorization by using the underlying group capabilities of the operating system. Group authorization must be explicitly enabled by the system administrator, who sets the environment variable DB2GROUPS to ON. (DB2GROUPS can be set in db2profile as described in “Step 3: Set Environment Variables” on page 58.)

If this variable is defined, the system enables group support when the instance is started. Follow these steps to enable group authorization:

1. Log in as the instance owner of DataJoiner.
2. Set DB2GROUPS to ON by using whichever method is appropriate for your shell environment.
3. Recycle (stop and restart) the DataJoiner instance.

After these steps, DataJoiner will run with group authorizations enabled. For more information see the *DB2 Administration Guide*.

Chapter 21. Migrating From Previous Releases

This section shows you how to migrate a previous release of DataJoiner to Version 2.1.1.

CAUTION!

> All migrations inherently involve a certain amount of risk, and DataJoiner is no exception. Therefore, proceed with caution. Read this entire section before starting the migration process, and make appropriate backups.
>
> **Do not uninstall an existing version of DataJoiner until your migration work is completed and tested.**

The migration process consists of two distinct, but interrelated, steps:

1. Instance Migration
2. Database Migration

Instance Migration

To prepare for instance migration:

1. Login as **sysadm** for the instance you are migrating.
2. Ensure that all local databases that you wish to migrate are cataloged before running the **db2imigr** command.
3. End all applications that are currently using the instance.
4. Stop the database manager and the command line processor.
5. Issue the **db2stop** and **db2 terminate** commands.

Login as root to perform the remaining steps.

Create a group and user for Fenced User Defined Functions (optional). This is described in “Step 2: Create a DataJoiner Instance” on page 57.

The **db2imigr** script is provided to migrate an existing instance to Version 2.1.1. The instance migration routine is invoked from the `/usr/lpp/djx_02_01_01/instance` directory as follows:

```
►►──db2imigr──instanceName──┬──────────────────────┬──┬────────────┬──────────────────►◄
                            └─-a──┬─server─┬───────┘  └─-uUdfName──┘
                                  ├─client─┤
                                  └─dcs────┘
```

where:

instanceName
Represents the name of the existing instance to be migrated.

-a Indicates the authentication type for the new instance. Valid authentication types are SERVER, CLIENT, and DCS. If the *-a* parameter is not specified, then the authentication type will default to SERVER.

Note: The authentication type of the instance applies to all databases under the instance.

-u Indicates the AIX user and group under which Fenced UDFs will execute.

Restriction:
UdfName may not be **root** or **bin**.

If *UdfName* is unspecified, the user and group default to nobody:nobody; this means that there are no file access permissions defined for UDFs or DARIs.

You may set the *UdfName* later with the **db2iupdt** command, if you do not do it now.

While **db2imigr** does not automatically initiate the migration of instances, it checks to detect conditions that would prevent the successful migration of any of the local databases that are cataloged in the instance. If it detects any of these conditions, then **db2imigr** aborts the instance migration and generates a report that lists the conditions that were detected. The report is stored in a file called `INSTHOME/migration.log`.

An error is logged for each database that is in one of the following states:

- Backup pending
- Roll-forward pending
- Database inconsistent

An error is also logged for each database object that uses either SYSCAT or SYSSTAT as the qualified object name.

Restriction:
SYSCAT and SYSSTAT are reserved schema names in Version 2.1.1.

A warning is also logged for each local database entry with an authentication type that is different from the authentication type specified on the **db2imigr** command. While this will not stop the migration, you may still want to take some action. See “Correcting Migration Inhibitors” on page 206 for suggested corrective actions, correct all the errors that are reported, and run **db2imigr** again.

When the **db2imigr** verification finds no errors, the instance migration will be initiated.

Selective Migration

To migrate only certain databases under a given instance, first execute the **db2ckmig** command to verify that the selected databases can be migrated.

1. Login as **sysadm**
2. Ensure that all local databases that you wish to migrate are cataloged before running the **db2ckmig** command.
3. End all applications that are currently using database manager.
4. Stop the database manager and the command line processor.
5. Issue the **db2stop** and **db2 terminate** commands.
6. Execute the **db2ckmig** command in `/usr/lpp/djx_02_01_01/bin` as follows:

```
►►──db2ckmig──database──/l filename──────────────────────────────────────────►◄
```

database
Specifies an alias name of a database to be scanned

/l Specifies a file to be used to keep a log of conditions that would prevent the successful migration of the database.

An error is logged if the database is in one of the following states:

- Backup pending
- Roll-forward pending
- Database inconsistent

An error is also logged for each database object that uses either SYSCAT or SYSSTAT as the qualified object name.

Restriction:
SYSCAT and SYSSTAT are reserved schema names in Version 2.1.1.

The **db2ckmig** parameters may be preceded by either a ″**-**″ or ″/″.

Correct all the errors that are reported (see “Correcting Migration Inhibitors” on page 206 for suggested corrective actions).

Back up the databases and restore them under a newly created Version 2.1.1 instance.

Correcting Migration Inhibitors

The following table shows common inhibitors to successful database migration and prescribes the appropriate solutions.

Table 9. Common Migration Inhibitors and Recommended Solutions

State of Database	Corrective Action
Backup pending	Backup the database.
Roll-forward pending	Recover the database as required. Perform or resume a **ROLLFORWARD DATABASE**.
Database inconsistent	Restart the database to return it to a consistent state.

See the (Version 1) *DB2 Administration Guide* for more information about the actions that are required to correct these conditions.

Requirement:
If databases contain one or more objects that use SYSCAT or SYSSTAT as schema names, then these objects must be dropped and recreated using a different schema name. Therefore, we recommend that you avoid using schema names that begin with SYS altogether.

All local databases now have the same authentication type as the instance where they reside. The authentication type in the database directory is ignored by DataJoiner Version 2.1.1 servers. If a warning is logged due to a *conflicting* authentication type, you have the following options to retain the same authentication type in your database:

- Change the authentication type of the instance.
- Move the database to another instance that has the desired authentication type.

Before changing the authentication type of the instance, you should make sure that the new authentication type will be appropriate for all databases that reside there. Be sure to consider the security implications of the different authentication types.

Suggestion:
If there are databases that you do not want to migrate, you can uncatalog them (along with all aliases). **db2imigr** does not perform any verification of uncataloged databases.

Migration Processing

The instance migration process performs the following actions:

- Backs up the down-level instance.

  A Version 1.2 instance is moved from INSTHOME/sqllib to INSTHOME/sqllib_v1, and a Version 2.1 instance is moved from INSTHOME/sqllib to INSTHOME/sqllib_v2. Once you are certain that you no longer require the down-level instance you can delete the INSTHOME/sqllib_v1 or _v2 directory.
- Creates a Version 2.1.1 instance.

  Relevant information from down-level DataJoiner directories and configuration files is copied and merged with appropriate Version 2.1.1 defaults.
- Updates the list of instances.
- Copies the down-level version's db2profile and db2cshrc files to db2profile.v2 and db2cshrc.v2 under the Version 2.1.1 instance.
- Copies the contents of the down-level version's function directory, INSTHOME/sqllib/function, to the Version 2.1.1 instance.

Reversing Instance Migration

If for some unexpected reason you must reverse the migration, login as root and issue the **db2imigrev** command as follows:

```
db2imigrev instanceName
```

Note: Databases that have been migrated to Version 2.1.1 are unusable to Version 1.2 or 2.1.

This will restore the INSTHOME/sqllib_v1 or _v2 directory to INSTHOME/sqllib and create a copy of the Version 2.1.1 INSTHOME/sqllib to INSTHOME/sqllib_v211.

You can now work with Version 1.2 or 2.1 databases that remain. To return to Version 2.1.1, execute **db2imigr** again; this restores Version 2.1.1 of DB2 from the copy (INSTHOME/sqllib_v211).

Database Migration

Use the following information to migrate your databases. The database manager must be started before migration can begin.

Restrictions:

You cannot migrate a database that is in one of the following states:

- Backup pending
- Roll-forward pending
- Database inconsistent

In addition, you cannot migrate a database that contains any database objects that have a qualifier (schema name) of SYSCAT or SYSSTAT. These two schema names are reserved for use by the database manager.

1. Backup your database.

   CAUTION!

   Database migration is not a recoverable process. If you backup your database before the schema names are changed, you will not be able to restore the database from backup by using DataJoiner Version 2.1.1. You will have to use the version of the database manager from which you are migrating your databases.

   If you do not have a backup of your database from before you attempted migration, and the migration failed, you have no way of restoring your database using DataJoiner Version 2.1.1 or your previous version of the database manager.

2. Migrate the database by using one of the following:
   - The **MIGRATE DATABASE** command-line processor command
   - The **RESTORE DATABASE** command, when restoring a full backup of the database.

   "Selective Migration" on page 205 describes what to do if you do not want to migrate all databases in a given instance.

3. Rebind all packages. All existing packages are marked as invalid during catalog migration. You can use the **DB2RBIND** utility to revalidate all packages, or allow package revalidation to occur implicitly when a package is first used. The **REBIND PACKAGE** or **BIND** commands selectively bind a particular package.

4. Tune your database and database manager configuration parameters to take advantage of Version 2.1.1 enhancements.

Part 6. Replicating Data

This section describes how to plan, configure, administer, and operate DataJoiner's replication administration tool. Because the replication administration tool has many of the same capabilities as the IBM DB2 Universal Database Replication Version 5 or Version 6[2] Control Center, this section merely expands the scope of the *IBM DB2 Replication Guide and Reference (V5)* and *IBM DB2 Replication Guide and Reference (V6)*, focusing on replication that involves non-IBM databases. All information about non-IBM replication can be found in this book. Information about replication between IBM's DB2 databases is documented in *IBM DB2 Replication Guide and Reference (V5)*. We suggest you read through *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* to understand the basics of IBM replication.

This section uses industry standard terminology for database, copying, and LAN concepts, except where noted otherwise. A comprehensive glossary is located in the *DB2 Universal Database Glossary*. The following terms represent concepts specific to the replication administration tool:

Control server
The database where the subscription definitions are located. The *control server* can be co-located at either the source or target server or at a DB2 database that is neither the source nor the target. The control server is usually located where the Apply program runs because the Apply program frequently reads the tables in the control server.

Differential-refresh copying
In differential-refresh copying, only the data that has changed in the source table since the last replication cycle is copied to the target table.

Full-refresh copying
During full-refresh copying, the Apply program:

1. Deletes all the rows from the target or target subset view
2. Reads all the rows from the source table or view
3. Copies (*refreshes*) the source rows to the target table.

Non-IBM server
A relational database that is not an IBM produced product. The non-IBM database (for example, Oracle, Sybase, or MS SQL Server) is

[2] DB2 UDB Version 5 ships with DataJoiner Version 2, but you can also use DB2 UDB Version 6 with DataJoiner Version 2.

identified (or defined) to DataJoiner to be accessed for replication. When a DataJoiner database is identified as a source or target server, the heterogeneous server must be identified to DataJoiner, specifying the specific Oracle, Sybase, MS SQL Server, or other data source.

Replication
Refers to the propagation of data from source tables to target tables. *Heterogeneous replication* is replication that involves a non-IBM database that is either a source or target for replication.

Sources
Source tables or DB2 source views in a replication scenario.

Source server
When the source data is in DB2, the source server is the database where the source data is stored and where the Capture program runs. When the source data is in a non-IBM database, the source server is a DataJoiner database and Capture triggers are located on the non-IBM database that holds the source data. The non-IBM database is accessed through DataJoiner.

Subscription members
A specification that identifies (to the Apply program) the exact source table that contains the data to be copied and the exact target table that the data will be copied to. This specification is one item in a subscription set.

Subscription set
A set of subscription members and their attributes that are all processed together in a single Apply program unit of work (UOW). Common attributes include: source server, target server and Apply qualifier.

Target server
The database where the target copy of the data is stored. When the target data is in a non-IBM database, the target server is a DataJoiner database.

Update-anywhere replication
Replication in which the replication source or the target table can be updated and the changes replicated to all related tables. Transaction-conflict detection is available only for replication between DB2 family sources and targets. Row-level conflict detection is supported between Microsoft Jet and any of the following: Oracle, Sybase, Microsoft SQL Server, and Informix.

Chapter 22. Data Replication–What is it?

Basically, data replication consists of two processes:

- The capturing of changed data from a table or view in a *source* database
- The copying (or "applying") of changed data from a *source* table to one or more *target* tables in the same or different databases

Although some aspects can vary, the basic processes are change capture and data copy. The IBM Replication Solution provides flexibility in determining how the changed data is to be captured and how (and under what conditions) this data is to be copied to the targets.

You must adminsiter the two basic processes. Replication administration is the process of creating and managing your replication environment. The administrative tasks consist of:

- Defining a table or view as a source table for change capture and replication
- Creating a target table
- Defining the conditions under which the data from the defined source table is copied to the target table

Additional administrative tasks include preparing a database server to participate in replication or changing an existing source table.

The IBM Data Replication Solution

IBM's data replication solution consists of data replication tools and products that function together in a single solution. The tools and products are:

- DataJoiner's replication administration tool: The tool for administering replication between IBM and non-IBM databases. The replication administration tool works with DataJoiner, Capture triggers, and the Capture and Apply programs.
- DB2 Universal Database (DB2 UDB): The Capture and Apply programs packaged[3] with DB2 UDB are compatible with DataJoiner's replication administration tool and DataJoiner V2.
- DataPropagator Relational (DPROPR) Capture and Apply programs
- DB2 DPROP for Microsoft Jet
- IMS DataPropagator (formerly called DataPropagator NonRelational)
- DataRefresher

- DataJoiner
- Lotus NotesPump

This is an overview of how these products support the complete data replication solution. The discussion here is about the replication features between IBM and non-IBM databases only. DB2 UDB V5, DPROPR Capture and Apply programs, DPROPNR, DataRefresher, DataJoiner, and Lotus NotesPump are discussed in other IBM publications.

Support of Replication Environments

By supporting sources and targets that include the DB2 family, IMS, VSAM, Oracle**, Sybase**, Microsoft**, Informix**, Lotus Notes, and others, IBM's replication solution ensures that you have timely, reliable, and consistent data across your enterprise. IBM supports replication in heterogeneous environments as follows:

- IBM's architecture is built on standard SQL to leverage the database capabilities for data enhancement, network connectivity, and data security.
- An architected data staging area supports interoperability among multivendor databases, between heterogeneous data models, and among products from independent software vendors. IBM's DPROPNR, working with DataRefresher,[4] brings IMS, VSAM, and flat file data into the staging area, making it available for replication to client/server targets.[5] Additionally, independent vendors offer products that read and write the data staging tables relying on IBM Capture programs, DataJoiner, and the Apply program.
- IBM replication products directly support multivendor sources and targets through IBM's multidatabase server product, DataJoiner V2. DataJoiner provides transparent access to Oracle, Sybase, Microsoft SQL Server**, and Informix, thus enabling replication to run across multivendor environments.
- Lotus NotesPump extends replication to Lotus Notes and Open Database Connectivity (ODBC)-accessible data (for example, data that can be accessed using Lotus Approach or Microsoft Access).
- DB2 DPROP for Microsoft Jet extends IBM's enterprise data replication solution to support Microsoft Access and Microsoft Jet databases in LAN, occasionally-connected, and mobile environments.

[3] The Capture and Apply programs are separately-priced features of DB2 UDB for OS/390. They are packaged with DB2 UDB on Windows 9x and NT, OS/2, and on AIX and other UNIX platforms.

[4] DataRefresher and ETI•Extract, by themselves, provide full-refresh capabilities. When used with the Data Difference utility (DDU), they can provide differential-refresh capabilities. The DDU is packaged with the latest versions of IBM's DPROPNR and Apply for MVS products.

[5] See http://www.software.ibm.com/data/dpropnr/ on the Internet for more information about DPROPNR and DataRefresher.

Replication Solution and Product Environment

Figure 24 conceptually shows the relationships between the components of IBM's replication solution.

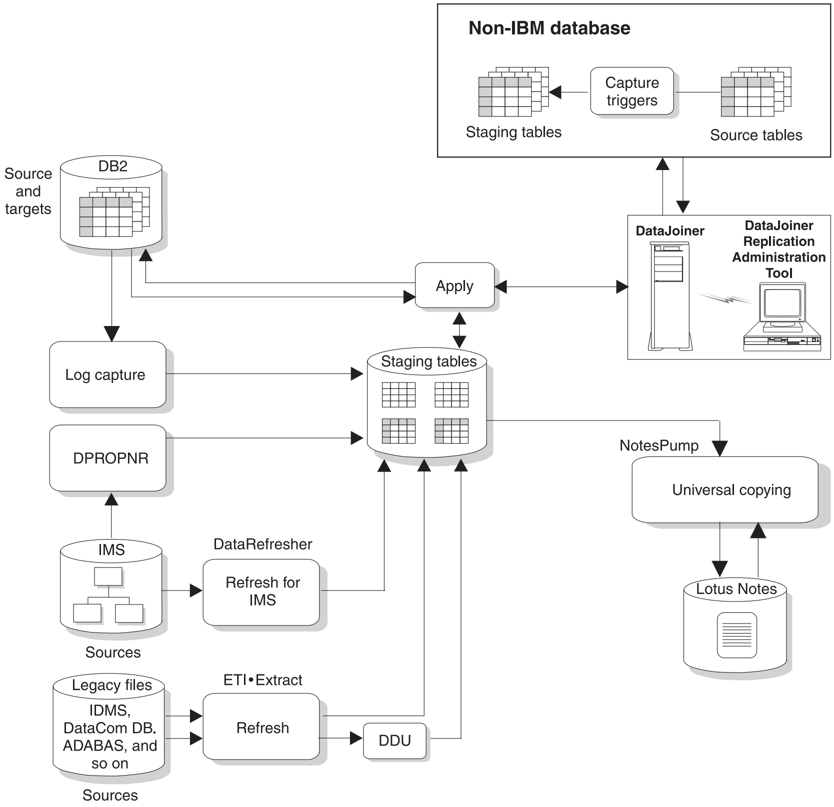


Figure 24. IBM's Data Replication Solution and Product Environment

For non-IBM sources, the replication administration tool works through DataJoiner to create Capture triggers on the source tables. These triggers capture the changed data for copying and place it into staging tables on the non-IBM source server. The Apply program accesses the source server through DataJoiner nicknames. The updates can be subsequently staged at any number of sites, allowing for flexible data distribution.

For non-IBM targets, the replication administration tool works through DataJoiner to create the target tables on the non-IBM server and nicknames for those target tables in a DataJoiner database. The Apply program then applies (copies) the changed data destined for the targets to the nicknames and DataJoiner applies those changes to the real target tables on the non-IBM server.

The Apply program can filter, enhance, distribute, and replicate data from the source tables to the target tables. DataJoiner enables the Apply program to *transparently* read from or write to the non-IBM sources and targets.

Multivendor Data Sources and Targets

IBM’s DataJoiner enables replication between IBM and non-IBM sources and targets, including DB2, Oracle, Sybase, SQL Anywhere, Microsoft SQL Server, and Informix. DataJoiner can help your business meet the challenge of efficiently accessing distributed data by enabling users to develop a virtual, enterprise-wide relational database.

DataJoiner provides a single-site, local image of all your data, relational and nonrelational, local and remote, from IBM and non-IBM platforms. DataJoiner masks the differences in SQL dialects, data access methods, networking protocols, and operating systems. In the context of replication, DataJoiner, together with the replication administration tool, provides a configuration that can access multivendor sources or targets as if they were all a DB2 database.

Chapter 23. The Replication Tools

The tools that provide support for the IBM replication solution are:

DataJoiner’s replication administration tool	The replication administration tool serves as your replication administration interface. This tool runs on Windows NT or 9x.
DataJoiner	DataJoiner provides access to non-IBM databases for replication purposes.
Capture program	The Capture program captures the data changes from DB2 databases for replication purposes.
Capture triggers	The Capture triggers capture the data changes from non-IBM databases for replication purposes.
Apply program	The Apply program performs the replication of the data from the source database to the target database.

The following sections describe how the replication administration tool, DataJoiner, the Capture program, the Capture triggers, and the Apply program work and interact with each other.

The Replication Administration Tool

The replication administration tool serves as your replication administration interface, automating many replication activities. From the replication administration tool, you can select your source tables and prepare them for change capture, create your target tables, and create the control information for the Apply program. The control information causes the Apply program to copy data from the selected source tables to the selected target tables. You can also use the replication administration tool to browse or change replication activities.

The databases that the replication administration tool supports as sources or targets are:

- DB2 UDB V5 and V6
- DB2 for MVS V3R1 or V4R1, and DB2 for OS/390 V5R1 or V6
- DB2 for common servers V2 and DataJoiner V2
- Oracle V7.0.13
- Informix V7.1 or later
- Sybase V4R6 or later (UNIX) and Sybase V11 (Windows NT)
- Sybase SQL Anywhere Version 5.0 (Windows NT)
- Microsoft SQL Server V4.21 or later (UNIX) and Microsoft SQL Server V6.0 or later (Windows NT)
- DB2 DPROP for Microsoft Jet (as target only)

You can use DataJoiner with the replication administration tool to define, synchronize, automate, and manage copy operations from a single control point for data across your enterprise, as shown in Figure 25. You can also tailor or enhance data as it is copied, thus delivering detailed, divided, summarized, or derived data when and where it is needed.

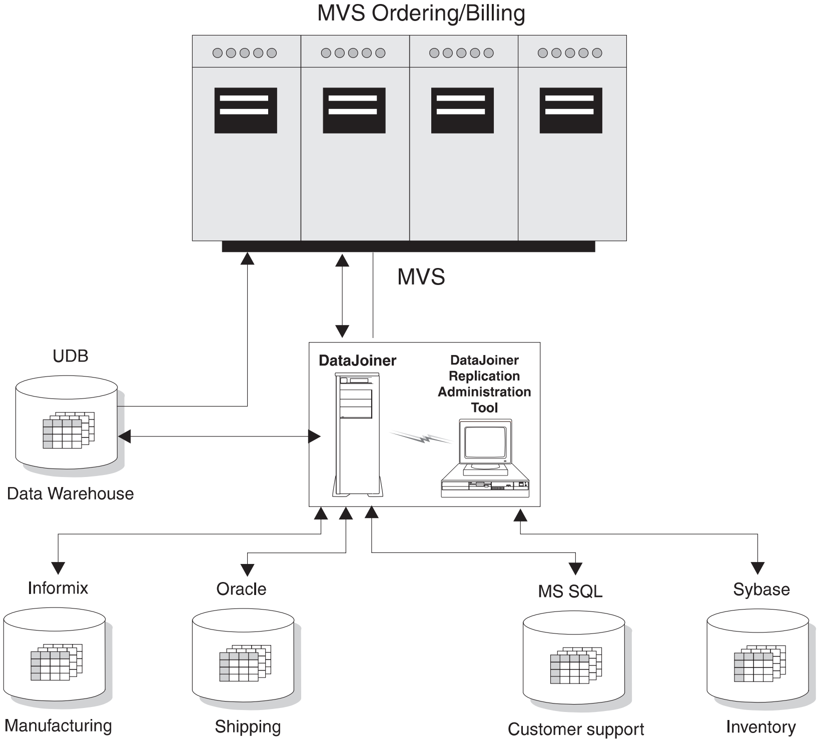


Figure 25. DataJoiner with the replication administration tool Scenario. DataJoiner with the replication administration tool enables you to copy data across your enterprise.

The replication administration tool provides objects and actions that define and manage source and target table definitions. Working through DataJoiner, the replication administration tool create:

- Capture triggers on the non-IBM source servers
- Nicknames in the DataJoiner database for the remote tables where the changed data is to be captured
- Target tables (and their associated nicknames) in the non-IBM database for the remote target tables

The Apply program then reads from and writes to DataJoiner nicknames, eliminating the need to connect explicitly to non-IBM databases.

If the source database is a DB2 database, the Capture program for that database captures the changes, therefore, the Capture triggers and DataJoiner are not involved. If the target database is a DB2 database, the Apply program writes the changed data to the DB2 target database directly and DataJoiner is not involved.

Administering Replication: Overview

When you use the replication administration tool to perform replication administration tasks, the replication administration tool connects to the source, target, or control server in order to create and update the control information and target tables on the server (depending on the operation performed). The client workstation where the replication administration tool is located, must be authorized and able to connect to all the source, target, and control servers that are managed by the replication administration tool.

For DB2 source, target, or control servers, DataJoiner's distributed database connection services (DDCS)or the IBM DDCS product provides connectivity.

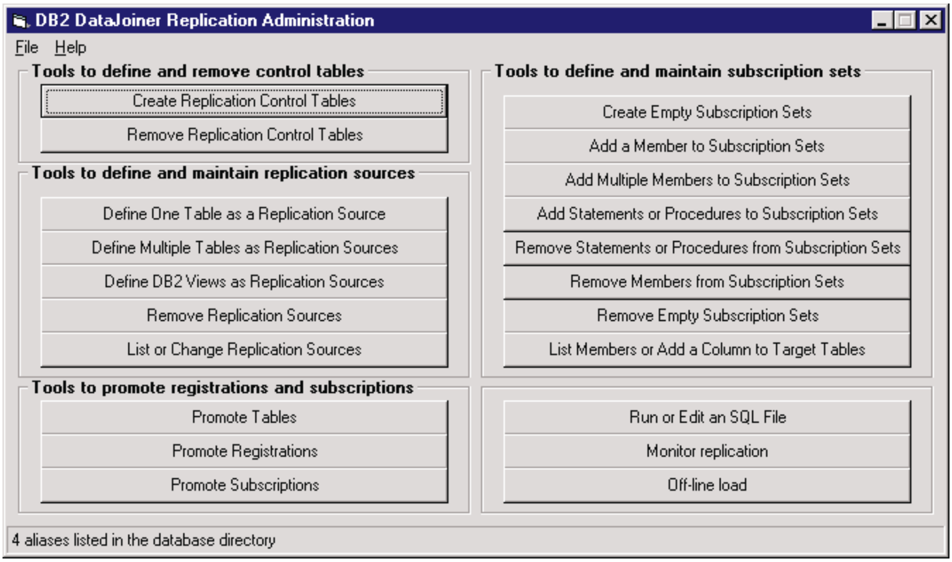
For non-IBM sources and targets, the replication administration tool uses DataJoiner to connect to the non-IBM servers. Non-IBM databases are not supported as control servers.

The replication administration tool graphical user interface (GUI) is divided into five areas that deal with control tables, promoting tables and subscription sets, sources, subscription sets, and the running or editing of SQL (see Figure 26 on page 218).

Using the GUI, you can perform the following administration tasks:

- Define DB2 tables, non-IBM tables, and DB2 views as sources. Sources are the tables or views that data is copied from.
- Create the control tables and put them on your source, target, and control servers. The control information for the replication process is kept in these control tables.
- Change the definitions for existing DB2 source tables in order to add new columns
- Remove or drop source tables and servers
- Define target table definitions, which consist of two parts:
  - Subscription members, which are requests for copying data from one source table to one target table
  - Subscription sets, which contain and group your subscription members into a single unit of work
- Change existing subscription members for DB2 target tables in order to add new columns

- Remove or drop subscription sets or subscription members that are no longer needed
- Add SQL statements or delete SQL statements or CALLed procedures that should be run before or after the target tables are replicated
- Run or edit SQL that is generated by the replication administration tool.


DB2 DataJoiner Replication Administration
File
Help
Tools to define and remove control tables
Create Replication Control Tables
Remove Replication Control Tables
Tools to define and maintain replication sources
Define One Table as a Replication Source
Define Multiple Tables as Replication Sources
Define DB2 Views as Replication Sources
Remove Replication Sources
List or Change Replication Sources
Tools to promote registrations and subscriptions
Promote Tables
Promote Registrations
Promote Subscriptions
Tools to define and maintain subscription sets
Create Empty Subscription Sets
Add a Member to Subscription Sets
Add Multiple Members to Subscription Sets
Add Statements or Procedures to Subscription Sets
Remove Statements or Procedures from Subscription Sets
Remove Members from Subscription Sets
Remove Empty Subscription Sets
List Members or Add a Column to Target Tables
Run or Edit an SQL File
Monitor replication
Off-line load

4 aliases listed in the database directory

Figure 26. The Replication Administration Tool's Primary Window. The replication administration tool is divided into five areas: control tables, sources, promote, subscription sets, and SQL.

How the Replication Administration Tool Manages the Replication Environment

The replication administration tool helps you perform three major tasks:

- Creating or removing control tables
- Defining, maintaining, or removing source definitions
- Creating or removing subscription sets or subscription members

Source tables are tables that data is copied from. The copy of the data can then be updated or inserted into, or deleted from, one or more target tables.

The Control Tables

This section contains an overview of the control tables. The details of each are discussed in “Tables Used by the Capture Triggers” on page 225, “Control Tables Used by the Apply Program” on page 228, the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)*.

Various components of the IBM replication solution use a set of control tables to hold information about:

- The source tables that are defined on a given source server
- The subscription sets that have been created between a specific set of source tables and target tables
- The information that is needed for communication between the Capture program (or Capture triggers) and the Apply program
- The information that is needed to manage (prune) the size of staging tables to prevent them from growing indefinitely

Not all of these control tables are used at every server. However, control tables that are used on source servers *must* exist in any DB2 or DataJoiner database to which the Apply program is bound (the Apply program contains static SQL that accesses those tables). For this reason, IBM recommends (and the replication administration tool generates) SQL to create all of the control tables on any specified source, target, or control server.

In order for the replication administration tool to put the control tables on the specified servers, the replication administration tool must use user IDs and passwords that have the proper authorized access to those servers. You define the user IDs and passwords for particular servers by using the replication administration tool. See “Setting Administrative Preferences” on page 272 for more information about specifying these user IDs and passwords.

Clicking **Create Replication Control Tables** from the replication administration tool primary window creates the control tables. This action generates SQL that can run immediately, can be edited (for customization purposes), or can run at a later time.

Note: You *must* create the control tables (that is, run the generated SQL) on a server *before* performing any replication requests (such as defining sources or creating subscription sets) that affect that server.

Sources and Targets

You can create your source and subscription member definitions through the replication administration tool. You use three objects to set up and maintain your replication environment:

Source definitions
Identify data in the source table as data to be replicated. This information is stored in the register control table.

Subscription member
Contains information that is used for copying data from one source table to one target table.

Subscription sets

Contains subscription members. The subscription sets group subscription members that are associated with a particular Apply program instance into a single unit of work. Subscription sets also describe the relationships between the source tables and one or more targets and contain the specifications for the target tables, their location, structure, a timing schedule, and, if the data being replicated is to be manipulated, the SQL necessary to do so.

SQL Generated for Replication Requests

Most of the replication administration tool functions generate SQL which, when run, creates, updates, or drops the specified objects on your servers.

When you choose the replication administration tool function, you can edit, save, and run the generated SQL immediately, or you can save the generated SQL and run it later. The file is saved as an ASCII file in your workstation file system.

If you choose to save the generated SQL as an ASCII file, you can edit it and run it at a later time through the replication administration tool. See "General Steps for Setting up Replication" on page 280 and "Editing the Replication Administration Tool-Generated SQL" on page 281 for more information about generating and editing SQL by using the replication administration tool. See "Running the Replication Administration Tool-Generated SQL" on page 282 for more information about running the SQL.

If you defer running the generated SQL, you can customize the replication tasks for your shop or application and determine when and how you run the SQL. You defer the processing of the replication action until a specified time, and create libraries of files containing SQL for backup, or site-specific customizing. You can also rerun the definitions as necessary. See "General Steps for Setting up Replication" on page 280 for more information on deferring and customizing generated SQL.

Whether you run the generated SQL immediately or at a later time, you must perform the activities for generating and running the SQL in a particular order. The order ensures that you create control tables at the source server before defining replication sources and that you create control tables at the source server *and* the control server before creating replication subscriptions. A recommended order is:

1. Generate SQL to create control tables at the source server and the control server.
2. Tailor the generated SQL if necessary.
3. Run the SQL to create control tables at the source server.

4. Generate SQL to define replication sources at the source server. The replication administration tool assumes that the source tables, nicknames, or DB2 views that are to be defined as a replication source already exist.
5. Run the generated SQL to define the replication sources at the source server.
6. Run the SQL to create control tables at the control server.
7. Generate SQL to create subscription sets on the control server.
8. Run the SQL to create subscription sets on the control server.
9. Generate the SQL to create subscription members at the control server.
10. Run the SQL to create subscription members at the control server.

DataJoiner

The replication administration tool accesses non-IBM databases through DataJoiner. DataJoiner presents a single view for various DB2 family and non-IBM databases, masking differences in data types, SQL dialects, and communications. You create nicknames for the non-IBM source or target tables in a DataJoiner database and then use the replication administration tool to create a source definition for the nicknamed object. Within a single DataJoiner local database, you can define the nicknames for one or more source tables that reside on *one* remote, non-IBM source server.

After you create subscription sets (using the replication administration tool) for a non-IBM source server, the Apply program connects to the DataJoiner database that is associated with the non-IBM server and accesses (through nicknames) the information in the register control table and the staging table on the non-IBM source server (see Figure 27).

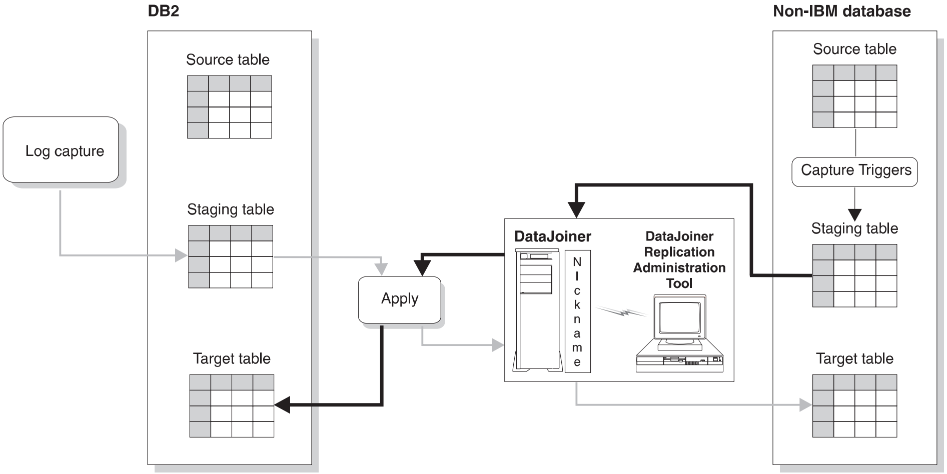


Figure 27. DataJoiner in Action. In a scenario where the source table is a non-IBM table (the dark arrows), DataJoiner nicknames give the replication administration tool and the Apply program access to the non-IBM source server and to changes made to the non-IBM source table (through the staging table). In a scenario where the source table is a DB2 table (the light arrows), DataJoiner nicknames give the Apply program access to the non-IBM target tables.

The Capture Program for DB2 Sources

The Capture program is the replication tool that captures the changed data from a DB2 source table (through DB2 logs) and makes the changed data available for replication. The Capture program runs at the DB2 source server database. Refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for more information on the Capture program.

Capture Triggers for Non-IBM Sources

Capture triggers are used for replication from non-IBM databases. They capture changed data from a source table and make the changed data available for replication. Capture triggers perform the same task as the Capture program does for DB2, but in a different manner. The replication administration tool generates the Capture triggers; they run in the source server database.

Capture Triggers Overview

The replication administration tool, working through DataJoiner, creates Capture triggers at the non-IBM source database when you define that database as a source server. Capture triggers capture committed changes made to source data and places the captured changes into a staging table, called the *consistent change data* (CCD) table. The CCD table has a nickname in DataJoiner that programs that want to replicate the changes (for example, the Apply program) can access. See “Changed Data Tables” on page 235 for more information about CCD tables.

There are three triggers for each source table: DELETE, UPDATE, and INSERT.

How the Capture Triggers Capture the Data Changes

The Capture triggers work with three objects: the CCD table, the register control table, and the pruning control table. The pruning control table is discussed in “Pruning the CD and CCD Tables” on page 237.

The replication administration tool generates SQL that, when run:

- Creates Capture triggers on the source table
- Creates the CCD table on the source server
- Inserts a row into the register control table (to represent the new source table)
- Creates a nickname for the CCD table in the DataJoiner database

Whenever a delete, update, or insert operation occurs at the defined source, a Capture trigger records the change into the CCD table and puts a marker in the register control table. When the Capture triggers retrieve changed information, they can also obtain before and after columns data to put into the CCD table.

The Apply program looks at a marker in the register control table to identify changes that have not yet been replicated. The Apply program then reads the CCD table (through DataJoiner nicknames), copies the changes to the target server, and applies the changes to the target table. Figure 28 on page 224 shows the relationship between the Capture triggers, the source table, the register control table, and the CCD table.

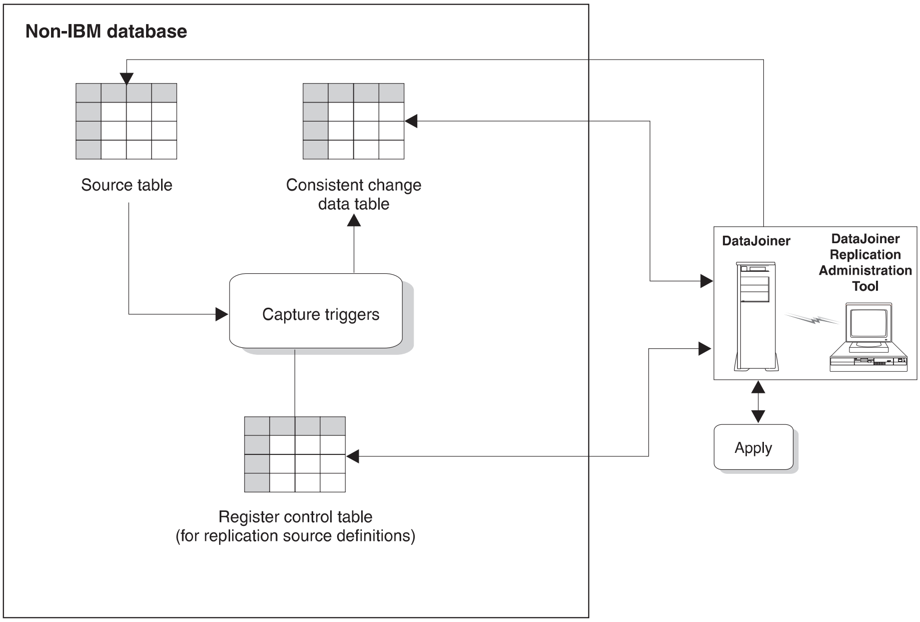

Non-IBM database
Source table
Consistent change data table
Capture triggers
Register control table (for replication source definitions)
DataJoiner
DataJoiner Replication Administration Tool
Apply

Figure 28. Capture Triggers at the Source Server. The Capture triggers monitor source changes, capture the changed data, and write the changed data to the CCD table.

Capture Triggers and Pre-existing Triggers

When the replication administration tool creates and places Capture triggers on a non-IBM database, you may experience the following occurrences:

- On Oracle: If an attempt is made to create a Capture trigger on a table where there is a pre-existing trigger with the same name, or where the pre-existing trigger performs an identical event (insert-before, insert-after, delete-before, delete-after, update-before, update-after), Oracle issues the following message: ORA-04081 (trigger name already exists). If this error is generated, the Capture trigger is not created.
- On Informix: If an attempt is made to create a Capture trigger on a table where there is a pre-existing trigger with the same name, or where the pre-existing trigger performs an identical event (insert, delete, update), Informix issues an *-741* error and will not create the Capture trigger.
- On Microsoft SQL Server or Sybase: If an attempt is made to create a Capture trigger on a table where there is a pre-existing trigger with the same name, or where the pre-existing trigger performs an identical event (insert, delete, update), Microsoft SQL Server and Sybase do not generate error or warning messages indicating a conflict. Microsoft SQL Server and Sybase replace the pre-existing trigger with the new Capture trigger. However, the replication administration tool does check to see if a trigger already exists. If a trigger with the same events exists, the replication

administration tool creates the new triggers but all lines within the trigger body are commented out. You must determine how you want to merge pre-existing triggers with the new triggers. Then you can uncomment out the lines in the new triggers.

If you anticipate conflict between the replication administration tool's Capture triggers and pre-existing triggers, we recommend that you put the content of both triggers into one trigger. For each table event, append the pre-existing business trigger to the end of the Capture trigger script that is generated by the replication administration tool.

Tables Used by the Capture Triggers

Two control tables and one staging table are used on each non-IBM source server to manage the task of capturing changes to the data in the source table. These tables (and, in the case of the Apply program, their nicknames in the DataJoiner database) are accessed and updated by the Capture triggers and the Apply program to keep track of:

- Capturing changed data for the source tables that are defined on that source server
- Applying the changed data to the various target tables

You must create the control tables on the source server (using the replication administration tool's **Create Replication Control Tables** function, *before* defining any source tables on the source server.

The control tables that the Capture triggers use on the source server are:

Register control table
Contains the name of the source table and the name of the CCD table that is associated with the source table. Also contains markers that track the sequence in which changes to the source table occur. There is one register table for each source server.

Pruning control table
Contains information about how far through the CCD table the Apply program has progressed in replicating changes to the target tables. One of the Capture triggers, the pruning trigger, uses this information to prune the CCD table. For more information on pruning, see "Pruning the CD and CCD Tables" on page 237.

The staging table that the Capture triggers use is called the CCD table. The CCD table contains the changed data thatthat the Capture triggers write, and is a source table itself for update copies. There is one CCD table, located on the source server, for each source table.

The control and staging tables are described in more detail in the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)*.

The Apply Program

The Apply program actually copies the changed data from the source table to the target table. You can run the Apply program at any server that has connectivity to the source and target servers, but the Apply program generally runs at the target server. See “Pull versus Push Apply Design” on page 242 for more information on locating the Apply program. The way the Apply program works with the replication administration tool and DataJoiner is almost identical to the way the Apply program works with DB2 systems.

The Apply program:

- Reads the changed data that was previously captured and stored in a change data (CD) table or CCD table
- Creates spill files (local to the Apply program) to hold the changed data that is to be replicated
- Copies the changes to target tables

When the source table is a DB2 table, the Apply program reads data directly from source table when copying the entire source table data for a full refresh to the target table. When the source table is in a non-IBM database, the Apply program accesses the source table through the DataJoiner nickname for that non-IBM source table. The Apply program must do an initial full refresh before any insert, update, or delete transaction can applied to a defined replication source.

The Apply program connects to the various servers (the control server, the source server, and the target server) to perform its tasks. It can be located anywhere in the network, as long as it can make the connections required. It retrieves the changed data with a block fetch and propagates the changed data with row inserts, updates, and deletes. As a result of this design, to achieve the best possible performance, the Apply program should be local to the target tables. When replicating to a non-IBM target table, the Apply program’s performance is best when it is located with the DataJoiner database that contains the nicknames for that non-IBM target table.

The Apply program runs independently of the replication administration tool, but uses control information that the replication administration tool creates. The primary control information that the Apply program uses is stored in control tables at the control server. The control tables that are used by the Apply program are described in “Control Tables Used by the Apply Program” on page 228.

The Apply program applies the changed data to target tables that are either on local servers (where the Apply program runs) or remote servers, working through DataJoiner nicknames if either the source or target tables are in a non-IBM database.

Figure 29 shows the Apply program's relationship with the source tables, the source server control tables, the subscription definition control tables, and the target table.

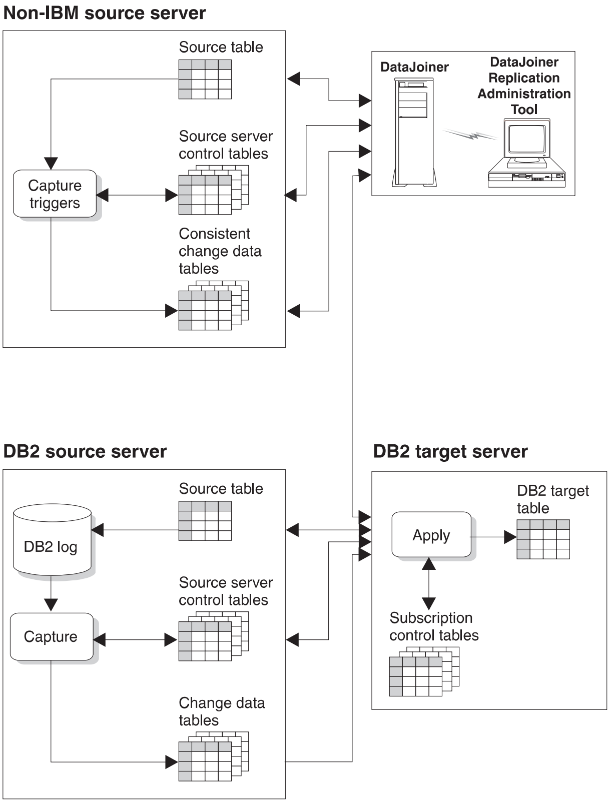


Figure 29. The Apply Program. The Apply program reads the data from the source server tables, manages the source server control tables, manages the subscription control tables, and updates the target table.

The Apply Qualifier

Several instances of the Apply program can share the same control server database. When started, the Apply program is supplied with an Apply qualifier, which is an arbitrary qualifier that is independent from a logon user ID. The Apply qualifier allows a logon user ID to run more than one instance of the Apply program. The Apply qualifier is one of the items that is specified in a subscription set.

Each subscription set has an associated Apply qualifier and one Apply qualifier can be associated with many subscription sets. As a result of this association, an Apply instance is responsible for processing only those subscription sets that have the matching Apply qualifier. You can have several instances of the Apply program running at the same time, each processing different subscription sets that are represented in the same control server. Using the Apply program password file feature[6], you can have the Apply program use a different user ID and password when connecting to each of the servers, independent of the user ID in effect when starting the Apply process or job.

Control Tables Used by the Apply Program

The control tables that are used by the Apply program are located at the source server and the control server. When the source table is a non-IBM database, the Apply program accesses the control tables through DataJoiner nicknames.

The Apply program relies on the following control tables to manage the replication of source table changes to target tables:

Source server control tables
The control tables that are used by Capture triggers and the Apply program. These control tables (which include the register and pruning control tables[7]) must exist on any DB2 or DataJoiner database that is bound to the Apply program.

Control server control tables
The control tables that are located on the control server. They are:

Subscription set control table
Defines the characteristics of each subscription set, such as the subscription set name, the Apply qualifier that is associated with the subscription set, and the timing of the subscription set.

Subscription members control table
Contains information for an individual subscription member that maps a specific source table to a specific target table. Subscription sets of DB2 views usually have many subscription members for the same target table.

[6] The Apply program password file is different from replication administration tool’s password file.

[7] The *register* table contains the name of the source table and its associated CD table. The *pruning control* table contains information about how much of the CD table the Apply program has replicated to the target tables; the Capture program uses this information to prune the CD table.

Subscription columns control table
Contains supplemental information about each copied column, such as renamed columns or computed columns.

Subscription events control table
Contains the timing information for processing the subscription set based on event triggering. A user application (or other subscription sets) maintains this table. This table itself can be replicated if you want to distribute event notifications. The replication administration tool supports continuous-timing processing of subscription sets.

Subscription statements control table
Contains the SQL statements or names of stored procedures to be run before or after the subscription set is processed.

Apply trail control table
Records statistics about refreshes and updates that are performed during each subscription set cycle.

Row-replica target list table
Maintains the names of the row-replica tables. This allows DB2 DPROP for Microsoft Jet to maintain a list of known row-replica tables in a DB2 or DB2 DataJoiner database. DB2 DPROP for Microsoft Jet uses this information during schema analysis to determine which, if any, row-replica tables should be deleted because the corresponding subscription member has been dropped since the last synchronization.

Subscription schema changes table
Used to signal modifications to a subscription.

Target Server Tables
The control tables that are located on the Microsoft Jet target server. They are:

Synchronization generations table
Used to prevent cyclic updates from propagating back to the RDBMS from a Microsoft Jet database.

Key string table
Maps Microsoft Jet table identifiers and row identifiers to primary key values.

Error messages table
Contains error codes and error messages.

Error information table
Contains additional information to identify the row-replica table and row that caused an error.

Error side information table
Contains the names of the conflict tables.

Conflict table
This table (one per target table, as needed at the target server) contains row data for DB2 DPROP for Microsoft Jet-detected conflict losers. If there is a conflict between the same row in the Microsoft Jet database (target server) and the source server, the row in the Microsoft Jet database ″loses,″ is added to the conflict table, and is replaced by the row in the source.

These control tables are described in more detail in the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)*.

The Apply Program and Control Servers

The control server is the logical server that contains the subscription control tables. Each Apply program is associated with a control server, which is specified during each call of the Apply program. Multiple Apply programs can share a control server.

Each control server can contain information about subscription sets that are associated with one or more Apply qualifiers. The qualifier associates a control server with an Apply program and one or more subscription sets.

The Apply Processing Cycle

Table 10 describes how the Apply program completes a replication cycle. In this example, the Apply program runs on the target server and connects to the control and source servers to access the control tables and to pick up the changed data.

Table 10. The Apply Program Cycle: A High-Level Overview of How a Subscription Set is Processed.

Step	Server
1. Check each subscription set associated with the Apply program’s Apply qualifier to see if it is “eligible for processing”, based on either time or an event.	Control server
2. For each “eligible” subscription set, check the sources for each subscription member to see if there is new, changed data to be copied to the target.	Source server
3. Pick up recent change data to be applied to the target tables for a subscription set.	Source server
4. Write the answer sets into a local ″spill″ file (possibly an in-memory file).	Target server

Table 10. The Apply Program Cycle: A High-Level Overview of How a Subscription Set is Processed. (continued)

Step	Server
5. Apply the change data in the spill file to the target tables.	Target server
6. Update subscription set status.	Control server
7. Report subscription member progress in the pruning control table.	Source server

Improving Apply performance for Sybase or Microsoft SQL Server

If you run the Apply program using a DBLIB connection for either Sybase or Microsoft SQL Server, and you have a slow network, you can significantly improve your overall replication performance. DB2 DataPropagator can use buffers to hold replicated data and send each buffer over the network rather than sending individual changes. You set the size of the buffers using the **create server option** statement. To take advantage of this improvement:

1. Retrieve the names of the packages for the Apply program. To find your package names, issue the following SQL statement:

   ```
   SELECT PKGNAME
   FROM SYSCAT.PACKAGES
   WHERE PKGNAME LIKE 'ASN%'
   ```

   The package names change with each release and with each service update, but this query retrieves names that are specific to your service level.
2. If you have an `apply_names.ini` file (in the `sqllib` directory), replace the package names with the ones that you retrieved in step 1. If you do not have an `apply_names.ini` file, create one and list the package names. The following lines show an example of an `apply_names.ini` file:

   ```
   ASN6A001+
   ASN6B001+
   ASN6C001+
   ASN6F001+
   ASN6I001+
   ASN6M001+
   ASN6P001
   ```
3. Create server options for the Apply packet and buffer sizes. Sample sever options for Sybase are:

   ```
   create server option apply_packet_size for server type sybase setting 16384;
   create server option apply_buffer_size for server type sybase setting 16384;
   ```

   Sample sever options for Microsoft SQL Server are:

   ```
   create server option apply_packet_size for server type mssqlserver setting 16384;
   create server option apply_buffer_size for server type mssqlserver setting 16384;
   ```

You can set the packet and buffer size to any appropriate value less than or equal to the maximum setting for Sybase or Microsoft SQL Server, and adjust as necessary.

4. Set the following environment variable:

   ```
   DJX_ASYNC_APPLY=TRUE
   ```

5. If you created or changed the `apply_names.ini` file, or if you changed the DJX_ASYNC_APPLY variable, you must stop and restart DataJoiner before these changes take effect. To stop and restart DataJoiner, issue the **db2stop** and **db2start** commands.

Chapter 24. Data Replication Concepts

This chapter describes concepts specific to the replication administration tool. The concepts are similar to the advanced replication concepts that are described in the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)*.

Subscription Members and Sets

A subscription member defines a source-to-target relationship between a source table and a target table. The member also defines the structure of the target table and specifies what is to be replicated to that target table. A subscription set contains the set's attributes and zero or more subscription members.

Subscription members that are associated with a particular source server *and* a particular target server can be grouped together in subscription sets. As a result of this grouping, the changed data for all the subscription members in a subscription set are replicated from their source tables to the specified target tables in one database unit-of-work.

One row in the subscription set control table identifies each subscription set. However, two rows identify a replica set. These rows differ from each other in one column that indicates the direction of replication.

See the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for more information about subscription rules and recommended uses.

Types of Copies

You can define subscription members with the replication administration tool to create the following types of copy tables, known as target table types:

User copy
A complete, condensed copy of the source table that must have a primary key.

Point-in-time copy
A complete, condensed copy of the source table at a certain point-in-time that must have a primary key. This table contains a time stamp column to indicate when a transaction occurred.

If the target is a non-IBM database, these two types (user copy and point-in-time) are the only two types supported.

When the target database is DB2, you can define two other types of copy tables: CCD (when a DB2 database is either source or target) and replica (when DB2 databases are both source and target). The replication administration tool does not support base aggregate or change aggregate types.

When the target database is Microsoft Jet, you define a row-replica table. Changes to this table are replicated back to the source table in an update-anywhere scenario. It differs from other replica tables in that conflicts are detected row by row, rather than by transaction.

See the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for descriptions of these other copy table types.

Full-Refresh and Differential Refresh

The Apply program copies data from the source to the target either by full-refresh copying or differential refresh copying.

During full-refresh copying, the Apply program:

1. Empties (deletes) all the rows from the target table
2. Reads all of the rows from the source table
3. Copies the rows to the target table

If the source table is in a non-IBM database, the Apply program accesses the source table through the DataJoiner nickname for the source table. You can specify full refresh only while defining a source table. If you have large tables, you might want to use a fastload program to simulate an initial full refresh copy. See the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for more information about fastload programs that can be used with the ASNLOAD Apply exit.

During differential-refresh copying, the Apply program copies only the changed data from the CD or CCD table to the target table. The first time the Apply program copies data to the target table, it does a full refresh to populate the target table. After the target table is populated, differential refresh is used.

Note: A non-complete CCD table is never fully refreshed because it contains only data that was changed in the source table.

Changed Data Tables

With the IBM Replication-Solution architecture, you can stage changed data; that is, place data in a table (called a staging table) to be accessed by another process (typically the Apply program). This section discusses two types of staging tables: CD tables and CCD tables.

When the source table is a DB2 table, the Capture program captures all the changes that are made to a source table and inserts change data rows into a CD table. The Apply program then pulls the committed changes from the CD tables, after which the Capture program automatically prunes change data rows from CD tables when they are no longer needed.

When the source table is in a non-IBM database, the Capture triggers capture committed changes that are made to a source table and insert change data rows into a CCD table. The Apply program then pulls the changes from the CCD tables. A pruning trigger (associated with the pruning control table) automatically deletes the change data rows from the CCD tables when the rows are no longer needed.

The CCD table that is created on a non-IBM source server, is built and filled differently than the CCD table discussed in *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)*, but the Apply program uses both kinds of CCD tables in the same manner.

In this book, we refer to the CCD table that the DataJoiner's replication administration tool creates in non-IBM databases and maintains by the Capture triggers. The CCD table discussed in the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)* is maintained by an Apply process and is an optional, secondary staging table that is a copy of previously captured changes.

CD and CCD Tables

A CD table receives changed data rows from the Capture program when changes are made to a DB2 source table. Rows in a CD staging table reflect changes that are equivalent, if not identical, to the original operational updates. Both uncommitted and incomplete changes can appear in rows in a CD table. The CD table has no knowledge of transaction boundaries. Nor does the CD table know whether or not the transactions that issue the updates are committed, are incomplete, or are in flight.

The Apply program joins the CD table with the unit-of-work (UOW) control table to determine which changes in the CD table have been committed and are to be applied to the target table. Uncommitted changes that are in the CD table are pruned as the changes to the source table are aborted.

Rows in a CCD staging table reflect the committed results from the Capture triggers when an insert, update, or delete operation takes place against a non-IBM source table. CCD staging tables contain only committed change data. The Apply program queries the register control table to find out what committed changes have not been copied yet to the target table. After the Apply program has this information, it retrieves the appropriate committed changes from the CCD tables and applies them to the target table. Figure 30 shows the columns in a CCD table.

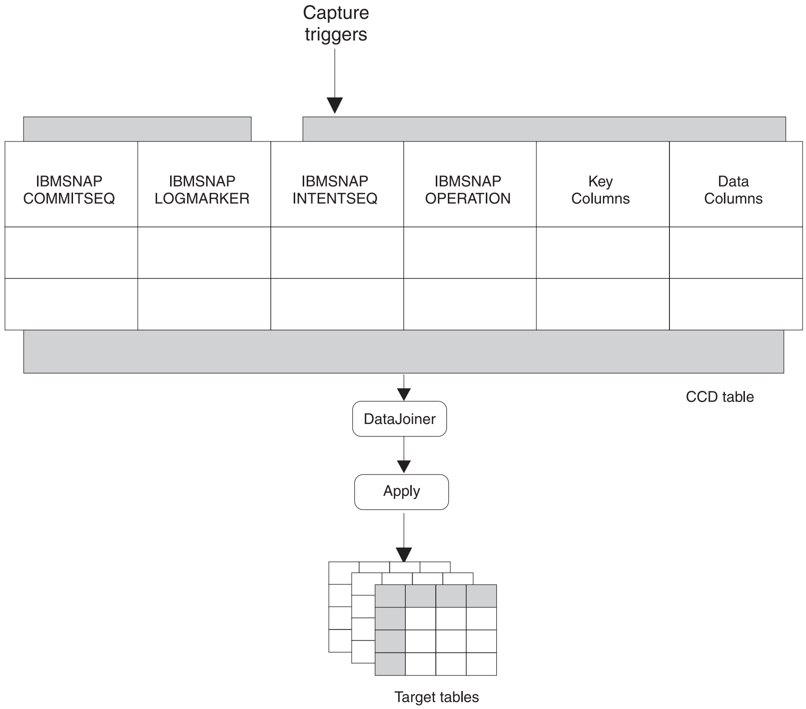


Figure 30. The CCD Table. The Apply program uses information in the CCD table to determine what to replicate.

If the source table is in a non-IBM database:

- The CCD staging table is local to the source table. If the source is a DB2 database, the CCD staging table can be local to, or remote from, the original source table. See the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for information on staging tables that are local or remote to the original source table.
- Two types of CCD tables are used. Generally, non-complete CCD staging tables are used when replicating changes to a target table. However, if the CCD staging table needs to be equivalent to a CD table that is maintained by the Capture program (in a DB2 environment), non-condensed CCD staging tables are used. Non-condensed CCD tables contain a row for each

insert, update, and delete to the source table with which the CD table is associated. See the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for a complete description of these types of staging tables.

- The non-complete CCD staging table (maintained by non-IBM triggers) is used only when replicating changes, not when the Apply program performs the full, initial refresh to a target table. When the Apply program performs a full, initial refresh to the target table, the Apply program uses the source table itself.

Pruning the CD and CCD Tables

When the source table is a DB2 table, the Capture Program prunes the CD table based on information inserted into the pruning control table by the Apply program. In this scenario, the Apply program maintains the CCD tables and *does not* automatically prune them. Refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for information about how these CCD tables are maintained and pruned.

When the source table is a non-IBM table, the Capture triggers prune the CCD table based on a synchpoint that the Apply program inserts into the SYNCHPOINT column of the pruning control table. Initially, when the Apply program performs a full refresh, it sets the synchpoint value to zero. A zero value signals the Capture triggers to start capturing changes to the source table. When the Apply program copies changes from the CCD table to the target table, the Apply program updates the SYNCHPOINT column. The pruning control trigger can then prune entries in the CCD table up through the row with the highest synchpoint value for changes that have already been replicated.

For the Capture program, you control whether or not pruning occurs by using the PRUNE or NOPRUNE parameter. You also control when the pruning takes place and how the prune interval is set in the tuning parameters table. See the Capture and Apply chapter for your platform in the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for more information about these parameters.

Transaction-Based versus Transaction-Consistent Replication: Using Internal CCD Tables to Reduce Network Load

The replication administration tool supports both *transaction-based replication* (replication of every update used by every transaction) and *transaction-consistent replication* (replication of only the committed results of the recent activity).

Figure 31 illustrates the difference between the two types.

```
Transaction 1: Update table1 set col1 = 'X' where key1 = 425
               Update table2 set col2 = 'B' where key2 = 425
Transaction 2: Update table1 set col1 = 'Y' where key1 = 425
Transaction 3: Update table1 set col1 = 'Z' where key1 = 425
```

Figure 31. Transaction-Based versus Transaction-Consistent Replication

In transaction-based replication, all four transactions are captured and replicated. In transaction-consistent replication, only the second update in Transaction 1 and Transaction 3 are replicated.

Transaction-consistent replication is superior to transaction-based replication because it produces the same change data results with fewer updates replicated. This type of replication reduces network load and can increase the availability of the target table.

You implement transaction-consistent replication by using the CCD tables created by the replication administration tool (the CCD tables maintained by the Apply program that are *local* to a CD table, not the CCD tables maintained by the Capture triggers). The CCD tables that are created by the replication administration tool are:

- Internal (local to the source table)
- Non-complete (containing only the rows involved in the replication process)
- Condensed (keeping only a copy of the latest captured value for each row)

The Apply program uses this CCD table as the source for replicating changes to the target table. The subscription sets are transaction consistent.

Before-Image Columns and After-Image Columns

A before-image column contains the value of a data column in a source table *before* that data column is updated. Similarly, an after-image column contains the value of a data column in a source table *after* the value in that data column is updated. Most database managers log both the before-image and after-image columns of the source table for each change to that table. The replication administration tool's Capture triggers can capture both before-image and after-image columns. Before-image columns can be useful to applications that require auditing or rollback capability.

See "IBM Replication Solution Restrictions" on page 247 for a list of the restrictions associated with before-image columns.

Before-Image Column Values

When you define a replication source and specify that you want to capture the before-image value of the source table, the before-image and after-image columns in the CD or CCD table have the following values when the source column changes:

Action	Column Value
Insert	The before-image column has a NULL value.
Update	Column values before the change are captured in the before-image columns; values after the change are in the after-image columns.
Delete	Both the before-image and after-image columns contain the before-image value.

NULL Values in Before-Image Columns

When the target table that will have before-image columns in it is initialized, before-image columns have NULL values. Full refresh copies are from the source table; therefore no before-image values are available. For example, the before-image columns of a point-in-time target table have NULL values if the source table has no updates because no before-image columns were copied.

Subsetting Target Tables

The replication administration tool supports both vertical and horizontal subsetting of the source table. Therefore, you can specify that only a subset of the source table columns and rows are to be replicated to a target table, rather than all of the columns and rows.

Vertical Subsetting

In some replication scenarios, you might not want to replicate all of the columns in the source table or view. This is appropriate, for example, if some of the columns in the source table are very large (like LOBs) or if the column data types are not supported by the intended target table. When the columns that actually get replicated are a subset of all the columns in the source table, this is called *vertical subsetting*.

See “IBM Replication Solution Restrictions” on page 247 for the restrictions associated with vertical subsetting.

You can implement vertical subsetting in either of the following two ways.

- While defining a source table, you can select only the columns that you want to be *available* for replication to a target table. Data for the unselected columns will not be copied into the CD or CCD table for this source table.

  Note: CD or CCD tables must contain sufficient key data for point-in-time copies, which are maintained with key-qualified predicates.
- While defining a subscription member for a source table (which results in the creation of that subscription member), use the **Add a Member to Subscription Sets** option to select, from the available source table columns, which columns you want to actually replicate to the target table for that subscription member.

Horizontal Subsetting

You can also subset the rows from the source table that are to be copied to the target table by specifying a row predicate when defining the subscription member for the target table. This predicate is in the form of a “WHERE clause” and is used to select the rows from the source table that will be copied to the target table.

The replication administration tool assumes that target table primary keys will not change. When the primary key of a non-IBM source table is updated, the Capture triggers place an UPDATE row in the CCD table for the non-IBM source table. When the Apply program reads this UPDATE row, the Apply program first attempts to perform a searched UPDATE, which will likely fail with a `row not found` condition. The Apply program then automatically converts the failed UPDATE into an INSERT. The target table then can have both the original row with the original key *and* the updated row with the new key. The original key no longer exists in the source table.

When the primary key of a DB2 source table is updated, the Capture program places an UPDATE row in the CD table for the source table. If one or more of the columns that are defined in the predicate of a DB2 source table are updated, you must use the Capture program function that defines updates as DELETE and INSERT statements. To learn how to use this option, see “Replication Logical Partitioning Key Support” in Chapter 2 of *IBM DB2 Replication Guide and Reference (V5)* or “Enabling replication logical-partitioing-key support” in Chapter 6 of *IBM DB2 Replication Guide and Reference (V6)*. The replication administration tool’s Capture triggers do not support the logical partitioning key feature. Updates will not be treated as DELETE and INSERT.

Combining Data

One of the important features of the IBM replication solution is that you can create and maintain target tables with contents that are joins or unions of existing source tables.

The IBM replication solution enables you to create:

- Joins of tables from a single DB2 source server (using views)
- Unions of tables from one source server
- Unions of tables from multiple source servers, sometimes referred to as multisite unions

You can perform a join of rows from source tables in a single source server by defining a view as a source and within this view definition, performing multi-table joins. With views you can also combine data by:

- Altering source definitions or subscription members
- Performing computations on one or more existing columns in the source table as the column is replicated to the target table

You can create a logical union of two or more source tables all residing on the same source server. This is done by having multiple subscription members within the same subscription set. Each subscription member is a source of the union.

You can create a logical union of two or more source tables that reside on different source servers. This is a more complicated scenario.

The technique for creating this logical union requires you to:

- Create target tables and views

  The target table must be partitioned through views such that each view isolates the rows that come from one of the source tables. Each target view must isolate its rows from the other target views.

  The target view defines the target rows that will be copied from a particular source table.
- Define subscriptions

  Each subscription member causes the changed source data to be replicated to a *target* view. Each subscription member lists the appropriate target view as the target table, not the target table itself.

  Multiple subscription members are in multiple subscription sets because there are multiple source servers.

SQL Statements and Stored Procedures

You can specify SQL statements or stored procedure calls that are to be run before or after the Apply program processes a subscription set. This feature is useful for manipulating change data and controlling the sequence in which subscription sets are processed (signalling subscription events). There are three types of run-time processing statements:

- At the source server, before the changed data is fetched (opening of the answer set cursors)
- At the target server, before the changes are applied
- At the target server, after the changes are applied

If the source table is in a non-IBM database, DataJoiner processes the SQL statements. The stored procedures use the SQL CALL statement, supported by the replication administration tool, without parameters. The procedure name must be eight characters or less in length. The run-time procedures of each type are processed together in a single unit-of-work. Acceptable SQLSTATEs can be defined for each processing statement.

Pull versus Push Apply Design

A pull versus push configuration is defined as where (physically) the Apply program is running, either at the source server or the target server. In the push method, the Apply program runs at a server other than the target server and pushes changed data to the targets. In the pull method, the Apply program runs at the target server and pulls changed data from the source. One Apply program could be pushing for some subscription sets and pulling for others.

When the Apply program processes a subscription member, it first connects to the source server in order to fetch the current changed data. If the source table is in a non-IBM database, the Apply program connects to a DataJoiner database (with DataJoiner connected to the non-IBM database) as the source server and accesses the data through DataJoiner nicknames. The changed data is fetched into a spill file that is local to the Apply program. When the data is retrieved, the Apply program connects to the target server and applies the changes, one row at a time, as an insert, update, or delete operation to each target table.

If the target table is in a non-IBM database, the Apply program connects to a DataJoiner database (with DataJoiner connected to the non-IBM database) and applies the data to the target table through DataJoiner nicknames.

Figure 32 on page 243 and Figure 33 on page 244 show push and pull scenarios. Notice that in both figures, the box that represents DataJoiner is labeled as the source server. This is from the Apply program’s point of view. The Apply program accesses the target tables through DataJoiner; that is, to the Apply program, DataJoiner *is* the source server.

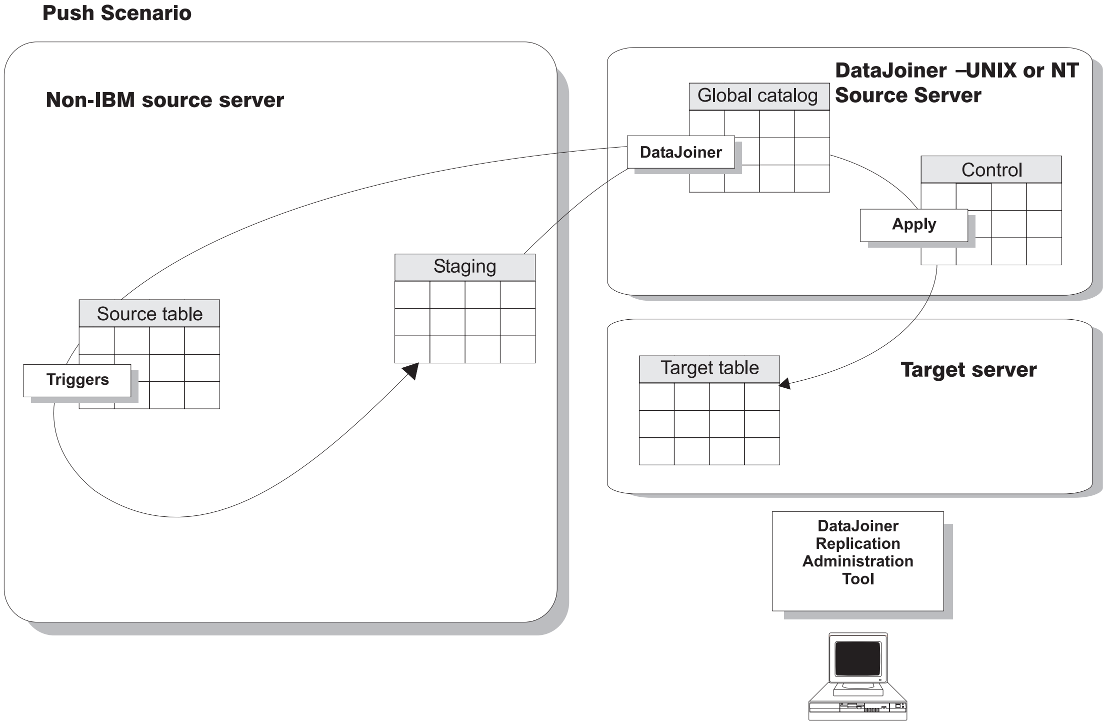


Figure 32. The Apply Program Push Scenario

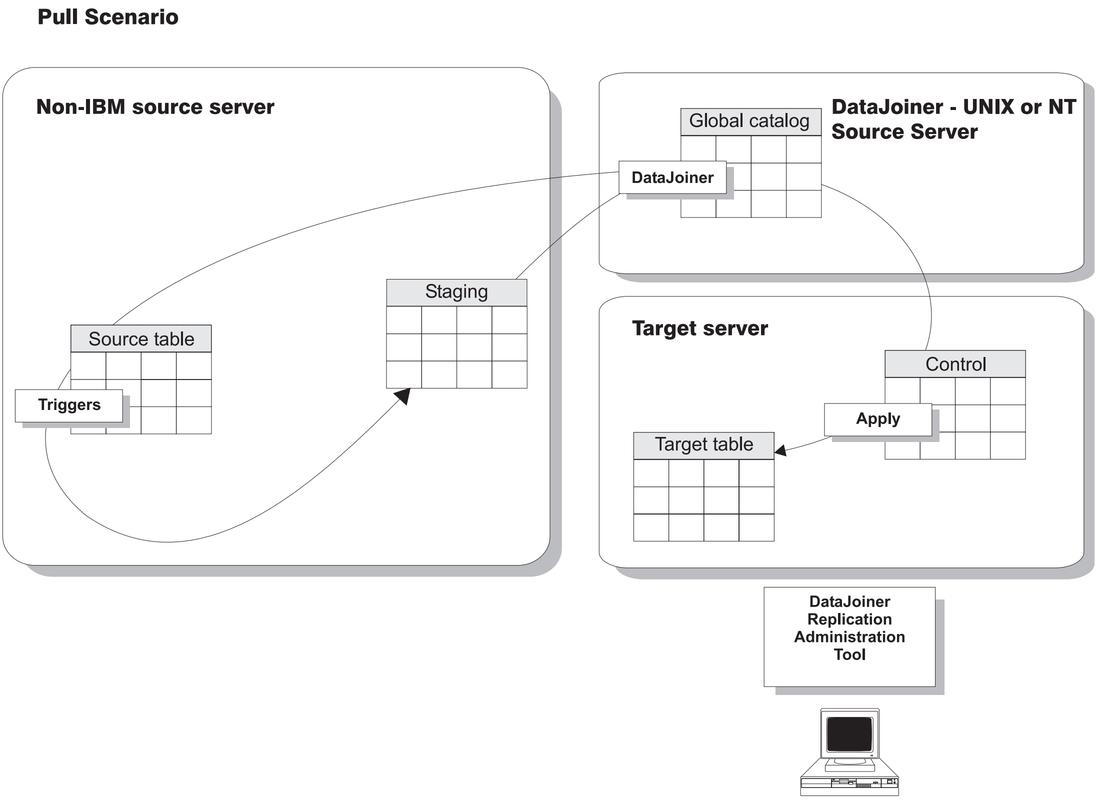


Figure 33. The Apply Program Pull Scenario

In pull mode, the Apply program connects to the remote source server (or to a DataJoiner database if the source table is in a non-IBM database) to retrieve the data. When all data is retrieved, the Apply program connects locally to the target server and applies the changes to the target table. The row-by-row process occurs as a local operation.

In push mode, the Apply program connects to the local source server, which is a DataJoiner database for non-IBM source servers. Then the Apply program connects to the remote target server (through DataJoiner if the source table is in a non-IBM database) and pushes the updates to the remote target table. The row-by-row process occurs as a remote operation.

The only thing you need to do to configure a push or pull configuration, is to decide where to run the Apply program. DataJoiner with the replication administration tool recognizes both configurations. DataJoiner with the replication administration tool automatically sets up the control tables so the Apply program can push or pull data.

Generally, a pull configuration performs better than a push configuration because the insert, update, and delete operations are performed to a local target rather than over a network. This is a more efficient use of the network. However, under the following circumstances a push configuration is a better choice:

- When there is no Apply program installed on the target server. In this instance, you might want to use the Apply program at the source server and *push* the data to the target server.
- The source table changes very infrequently, but when it changes it should be replicated as soon as possible.

Update-Anywhere Replication

The replication administration tool supports update-anywhere replication only for DB2 sources and targets. Non-IBM sources or targets are not supported. Refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for information on update-anywhere replication within the DB2 database family.

Views for Sources and Targets

You can define source views only for DB2 sources. You can define target subset views only for DB2 targets.

Source Views

Join views fill many requirements: denormalizing (restructuring) copies in data warehouse scenarios (thus enabling easier querying of copied data), and addressing the routing problem, sometimes called the database partitioning problem in distributed computing scenarios. For example, knowing where to send a bank account update may require a join of the account table with the customer table in order to know which branch of the bank the customer deals with. Typically, production databases are normalized so that the geographic details, such as branch-number, are not stored redundantly throughout the production database.

Using DataJoiner, you can create a join. Using the replication administration tool, you can define a join as a source table. The joins can only include tables that are defined as source tables. If the source tables defined in the join have CCD tables, a CCD view is created from the source server control tables.

DataJoiner with the replication administration tool supports the following types of view definitions:

- Simple inner-joins over one or more defined source tables

- Simple inner-joins over CCD staging tables that are defined as source tables and maintained by an Apply program or an application other than the replication administration tool and an external data source, such as DataPropagator NonRelational and IMS source data
- Only DB2 views or views of tables that reside within DataJoiner databases. Views of tables that are stored on Oracle, Microsoft SQL Server, Sybase, or Informix are not supported.

Target Views

You use the target views only when consolidating data from multiple sources. Multiple source tables are updated and consolidated into a union at the target server. The replication administration tool supports these types of views, even though it does not create them.

Preventing Gaps in the Target Table

The Capture triggers operate synchronously; therefore you do not experience gaps between the capturing of the changed data and the replication of that data to the target table. However, under certain circumstances during DB2 replication, gaps can occur if you stop running the Capture program. Refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for information about these circumstances.

Coordination of Source Security and Target Security

In a heterogeneous environment, each system has its own security system. These security systems need to be coordinated so the replication data can flow from the source database to the target database. DataJoiner coordinates the security according to how the security information from each system is defined to DataJoiner. See “Mapping of User IDs” on page 260 for more information about the security necessary for data replication and how the replication administration tool and DataJoiner enable this security.

Promote Functions

You can use the promote functions from the replication administration tool to reverse engineer your tables, registrations, or subscriptions, in order to create a script file with appropriate data definition language (DDL) and data manipulation language (DML). These functions are useful when you define tables, registrations, or subscriptions on one system (a test system, for example), and you need to copy the replication environment to another system (a production system, for example). Table 11 shows the three promote functions.

For example, use the promote functions to define subscriptions for remote DB2 Personal Edition targets. After you define a model target system in your test environment, you can create subscription scripts (and modify the Apply Qualifier and so on) for your DB2 Personal Edition systems, which are not otherwise supported from a central control point.

Table 11. Promote Functions

Promote table	This function promotes tables, table spaces, and indexes. It does not promote constraints defined for a table. This function is fully supported for UDB V5, but for the IBM Common Server you can only promote tables, not table spaces.
Promote registration	This function promotes registrations and view registrations from a source server.
Promote subscription	This function promotes subscriptions, subscription sets, subscription members, subscription columns, subscription prune control, and subscription statements. From the **Promote Subscriptions** window, you can change your subscriptions (before promoting them) by setting new values for any of the following fields: Apply Qualifier Set Name Source server Source alias Target server Target alias Control server Control alias

IBM Replication Solution Restrictions

Keep the following restrictions in mind when you are planning to use the replication administration tool in your replication scenarios:

- Check the restrictions of all the products in your replication environment to make sure that your use of the replication administration tool is consistent with those restrictions.
- You must have one DataJoiner database for *each* non-IBM source server.

Although one DataJoiner database can be used as a source table for replicating data to more than one non-IBM target server, a unique DataJoiner database is needed for *each* non-IBM source server. The reason for this is as follows.

For every replication scenario, there is a set of control tables, each with names that cannot be changed. When replicating to non-IBM target servers, none of these control tables needs to be located in the non-IBM database because a DataJoiner database is the target for the Apply program. The nicknames used here refer to the target table and not to any of the control tables.

For non-IBM source servers, however, some of the control tables *must* be located in the non-IBM database so the Capture triggers can update them. Because of this location requirement, the DataJoiner nicknames associated with those control tables must be the actual control table names, and their schema must be ASN. Because a DataJoiner database cannot contain more than two identical nicknames with identical schemas, one DataJoiner database must be used for each non-IBM source server. You can, however, support multiple non-IBM source servers within one DataJoiner instance by creating multiple DataJoiner databases within that one DataJoiner instance.

- The control tables that are used by Capture triggers and the Apply program must exist on any DB2 or DataJoiner database to which the Apply program is bound.
- You can define source views only for DB2 sources. You can define target subset views only for DB2 targets.
- The replication administration tool does not support:
  - Double-byte character character set (DBCS)
  - Synchronous replication (replication of data within the same unit of work as the original change to the source database). If synchronous data replication is essential to your application, use DRDA two-phase commit within the application.

    However, the replication administration tool works asynchronously and therefore does not affect the application that is making the original change to the source database. By working asynchronously, the replication administration tool provides you with the benefits of staging the changed data (for example, better network use, less database contention, and the opportunity to enhance the data as it is replicated).
  - Delimited names.
  - Update-anywhere replication for non-IBM sources and targets.
  - Microsoft SQL Server’s ability to access a database by defining an alias for the user.
- The Capture triggers do not support optional processing updates as delete and insert statements.
- The restrictions associated with before-image columns are:

- Before-image columns in a target table cannot be designated as primary key columns of the target tables.
- Before-image columns cannot be referred to in subscription member predicates for user copies and point-in-time copies.
- Before-image columns cannot be used in replicas or row replicas

Chapter 25. Replication Requirements

The following sections describe the replication administration tool hardware and software requirements.

Supported Sources and Targets

You can use the replication administration tool to set up replication between the databases that are listed in Table 12:

Table 12. Supported Sources and Targets

Database	**Capture and Apply Programs needed**
DB2 UDB V5 or V6	Includes Capture and Apply programs. The Capture program is required when DB2 UDB is used as source. The Apply program is required when the Apply program is located at the DB2 UDB target.
DB2 for MVS V3R1, DB2 for MVS V4R1, and DB2 for OS/390 V5R1 or V6	The Capture program for MVS is required when DB2 for MVS or DB2 for OS/390 is the source. The Apply program for MVS is required when the Apply program is located at the DB2 for MVS or DB2 for OS/390 target.
DB2 for common servers V2 and DataJoiner V2	The Capture program is required when DB2 for common servers or DataJoiner is the source. The Apply program is required when the Apply program is located at the DB2 for common servers or DataJoiner target.
Oracle V7.0.13 or later (UNIX) and Oracle V7.3 or later (Windows NT)	The replication administration tool creates Capture triggers on an Oracle source. The Apply program is required when Oracle is the target (with DataJoiner as the target server).
Informix V7.1 or later (UNIX) and Informix V7.2 or later (Windows NT)	The replication administration tool creates Capture triggers on an Informix source. The Apply program is required when Informix is the target (with DataJoiner as the target server).
Sybase V4.6 or later (UNIX) and Sybase V11 or later (Windows NT)	The replication administration tool creates Capture triggers on an Sybase source. The Apply program is required when Sybase is the target (with DataJoiner as the target server).
Sybase SQL Anywhere 5.0 (Windows NT)	The replication administration tool creates Capture triggers on an SQL Anywhere source. The Apply program is required when SQL Anywhere is the target (with DataJoiner as the target server).

Table 12. Supported Sources and Targets (continued)

Database	Capture and Apply Programs needed
Microsoft SQL Server V4.21 or later (UNIX) and Microsoft SQL Server V6.0 or later (Windows NT)	The replication administration tool creates Capture triggers on an Microsoft SQL Server source. The Apply program is required when Microsoft SQL Server is the target (with DataJoiner as the target server).
Microsoft Jet	DB2 DPROP for Microsoft Jet includes Capture, Apply, and Administration functions in a single program.

DataJoiner support for data sources varies by platform. DataJoiner also has client software requirements for each data source. See “Requirements for DataJoiner Data Sources” on page 28 for a list of clients supported by DataJoiner.

The Replication Administration Tool Hardware Requirements

The hardware requirements are:

- A personal computer that supports Windows NT or Windows 95 or Windows 98
- 24 MB RAM (32 or more is recommended)
- 2 MB hard drive storage

DB2 DPROP for Microsoft Jet Hardware Requirements

DB2 DPROP for Microsoft Jet runs on any IBM PC-compatible hardware with a minimum of 16 MB of storage. The program requires 1 MB of disk space. In addition, DB2 DPROP for Microsoft Jet requires additional storage space for:

- Microsoft Jet tables
- Spill files for fetching answer sets
- Log and trace files

You install DB2 DPROP for Microsoft Jet on the client as part of the DataJoiner Replication Administration tool installation process.

The Replication Administration Tool Software Requirements

You can set up your the replication administration tool environment in a variety of ways. This section shows the most common ways to set up your environment and the software requirements that are associated with each environment setup. Figure 34 on page 253 shows software requirements if DataJoiner for Windows NT and the replication administration tool are located on the same workstation. Figure 35 on page 254 and Figure 36 on page 255 show software requirements if DataJoiner and the replication administration tool are on separate systems.

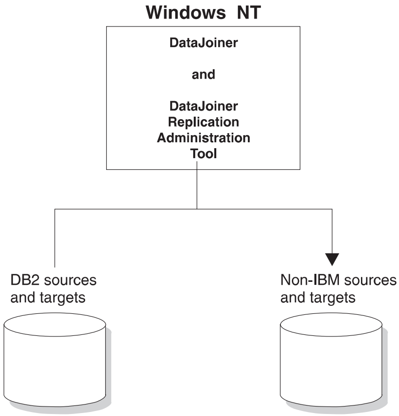


Figure 34. DataJoiner and the Replication Administration Tool on a Single Windows NT Workstation.

In Figure 34, DataJoiner provides access to DB2 databases. DataJoiner contains the same function as CAE V2 and DDCS V2. With this configuration, the replication administration tool software requirements are:

- Microsoft Windows NT V3.5.1 or later with Microsoft service pack 3
- DataJoiner for NT V2

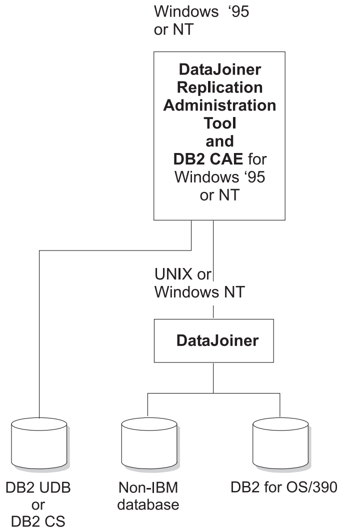


Figure 35. DataJoiner and the Replication Administration Tool on Separate Systems: DB2 for OS/390 Uses DataJoiner to Connect.

In Figure 35, the replication administration tool installed on Windows NT, 95, or 98 works with DataJoiner for UNIX or Windows NT to access DB2 and non-IBM sources and targets. With this configuration, the replication administration tool software requirements are:

- Microsoft Windows 95 with Microsoft service pack 2, Windows 98, or Windows NT V3.5.1 or later with Microsoft service pack 3
- Client Application Enablers (CAE) V2.11 or later for Windows NT, Windows 95, or Windows 98

DataJoiner V2 for UNIX or Windows NT provides access to DB2 for OS/390 and non-IBM sources and targets. If you have DB2 UDB or DB2 for common servers on the same workstation that you have the replication administration tool, you do not need to install CAE. It is provided with DB2.

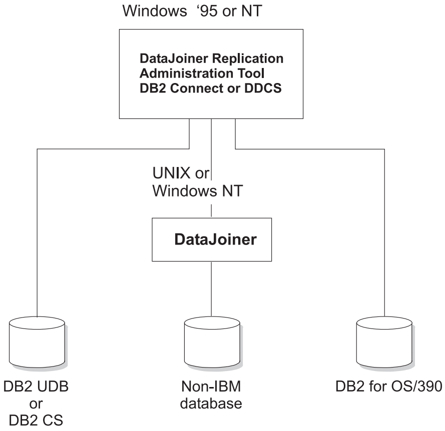


Figure 36. DataJoiner and the Replication Administration Tool on Separate Systems: DB2 for OS/390 Uses the Replication Administration Tool to Connect.

In Figure 36, the replication administration tool provides access to DB2 for OS/390. With this configuration, the replication administration tool software requirements are:

- Microsoft Windows 95, 98, or Windows NT V3.5.1 or later
- DB2 Connect or DDCS for common servers V2R3 or later, to access DB2 for OS/390 sources and targets

DataJoiner V2 for UNIX or Windows NT provides access to non-IBM sources and targets. CAE is provided with DB2 Connect. DDCS prerequisites DB2, and DB2 provides CAE.

DB2 DPROP for Microsoft Jet Software Requirements

To run DB2 DPROP for Microsoft Jet, the following software must be installed on the client machine:

- Microsoft Windows NT, Windows 95, or Windows 98
- Microsoft Jet or Microsoft Access for Windows NT (or for Windows 95 or 98)
- DB2 Client Application Enabler (DB2 CAE)
- DB2 DPROP for Microsoft Jet

For information about installing Microsoft products, refer to the documentation that is shipped with that product. For information about installing DB2 Client Application Enabler, see the documentation available with the product.

To run DB2 DPROP for Microsoft Jet, the DataJoiner Replication Administration tool must be installed on the control server.

Capture and Apply Program Requirements

When you replicate from a non-IBM source, the replication administration tool provides Capture triggers. When you replicate from a DB2 source, the Capture program is provided as part of the DB2 package or is separately orderable. The Apply program for UNIX is provided with DataJoiner for UNIX. The Apply program for NT is provided with DataJoiner for NT. When you replicate to a DB2 target, the Apply program is provided as part of the DB2 package or is separately orderable.

Requirements for the Capture and Apply programs are described in the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V5)* and the *IBM DB2 Replication Guide and Reference (V6)*. Table 13 lists the Capture and Apply programs supported by DataJoiner with the replication administration tool.

Table 13. Summary of Capture and Apply Programs Supported by DataJoiner with the Replication Administration Tool

The Capture Program	The Apply Program
Capture for UNIX	Apply for UNIX (you can select Apply during DataJoiner custom installation)
Capture for Windows NT	Apply for Windows NT (Apply for Windows NT is included with DataJoiner)
The V5 Capture program	The V5 Apply program
Capture for OS/2	Apply for OS/2

Authorization Requirements

The replication administration tool uses mechanisms already in place for database security. This section describes the authorizations that the replication administration tool and DataJoiner use for the:

- Administration of replication
- Capture program
- Apply program

A sample set of privilege levels for replication user IDs is provided in “System-Specific Privilege Levels” on page 259.

Authorization Requirements for Administration

For each database involved in replication, the administrator who is working with replication must have sufficient privileges to define replication sources and targets. As administrator, you must be able to:

- Select catalog tables of source or target databases.
- Create control tables at the source, target, or control servers.

For non-IBM databases, you must be able to:

- Create tables.
- Create Capture triggers on:
  - Source tables
  - Control tables
- Create procedures.
- Create sequences.[8]
- Select system tables.
- Select catalog tables.

For DB2 databases, you must be able to:

- Create tables.
- Create table spaces.
- Bind plans on each DB2 database involved in replication, including the source server, target server, and control server.

Most users set the replication administration tool to use the ID and password of a database administrator to connect to source, target, and control servers. The section "Setting Administrative Preferences" on page 272 describes how to set up one or more IDs and passwords in the replication administration tool. To complete the set up of your replication environment you need to map your authorizations through DataJoiner. See "Mapping of User IDs" on page 260.

Authorization Requirements for the Capture Program

For heterogeneous replication, the replication administration tool creates Capture triggers to capture changes from the source. Capture triggers do not require additional authorization. If your source is on a DB2 database, you use the Capture program to capture changes; some authorization is required. Refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for Capture program authorization and security issues.

[8] In Oracle only.

To complete the set up of your replication environment you need to map your authorizations through DataJoiner. See “Mapping of User IDs” on page 260.

Authorization Requirements for the Apply Program

The user ID that will run the Apply program (Apply user ID) must be a valid logon ID on the source, control, and target servers. This user ID must be able to:

- Access the replication source tables.

  If the source table is at a non-IBM database, the Apply user ID:

  – Must be able to access the DataJoiner nickname associated with the source table

  – Must be mapped to a user ID (through DataJoiner) that has access privileges for the source table at the non-IBM database
- Access the replication target tables.

  The Apply user ID must have update privileges for the target table. For a non-IBM database, the Apply user ID:

  – Must be able to update the DataJoiner nickname associated with the target

  – Must be mapped to a user ID (through DataJoiner) that has update privileges for the target table at the non-IBM database
- Access and update all control tables that are generated by the replication administration tool and built at the source and target database.

  For non-IBM databases, the register and pruning control tables have nickname equivalents in DataJoiner. The Apply user ID that is pushing or pulling changed data to a target:

  – Must have access and update privileges to the nicknamed source control tables in DataJoiner

  – Must be mapped to a user ID (through DataJoiner) that has both access and update privileges for the control table at the non-IBM database
- Have run privileges on the Apply program plan at the source, target, and control server.

  With the proper authorization, any user ID can run any Apply program instance. The restriction of one user ID per Apply program instance has been removed.

Additional notes about the Apply program authorization:

- You can use different user IDs at your source, target, and control servers.
- On DB2 for MVS, the Apply for MVS load library must be APF-authorized.

- An Apply program running on DB2 Universal Database might require a password file to connect to the source or target server. See “Providing End-User Authentication in UNIX” on page 276 for more information.

To complete the set up of your replication environment you need to map your authorizations through DataJoiner. See “Mapping of User IDs” on page 260.

System-Specific Privilege Levels

This sections provides a sample set of privilege levels you can use for IDs used for replication.

DB2 UDB
You must have DBADM, CONTROL, or SELECT privileges at the source, target, and control server.

DB2 for common servers
We recommend SYSADM or DBADM privileges at the source, target, and control server.

DB2 for MVS or DB2 for OS/390
At the source, target, and control server, you must have CREATETS privilege and any of the following privileges: SYSADM, DBADM, or CREATETAB.

Oracle IDs are set up with a default table space (with unlimited quota), default temp table space of TEMP, and default profile of DEFAULT. They are granted a system privileges of CREATE SEQUENCE/SESSION/TABLE/TRIGGER/VIEW. No password is required to enable the role, and the operating system requires no authentication.

Sybase on UNIX
IDs are set up with a default database. They are part of a group that has access grants of CREATE DEFAULT/PROCEDURE/RULE/TABLE/VIEW.

Sybase on NT
IDs are set up with a default database. They are part of a group that has object permission filters of PROCEDURE/TABLE/VIEW and command permissions of CREATE DEFAULT/PROCEDURE/RULE/TABLE/VIEW.

Sybase SQL Anywhere
IDs are set up with a default database. They are part of a group that has object permissions filters of TABLES/VIEWS/STORED PROCEDURES on ALL OBJECTS.

Microsoft SQL Server

IDs are set up with a default database. They are part of a group that has object permissions filters of TABLES/VIEWS/STORED PROCEDURES on ALL OBJECTS.

Informix

IDs are set up with default database and database privileges of RESOURCE which gives them CREATE INDEX/PROCEDURE/SYNONYM/TABLE/VIEW privileges.

Mapping of User IDs

This section explains how user IDs used for replication activities are mapped, through DataJoiner, to source, target and control servers. For a high-level discussion of DataJoiner's mapping coordination, see "Coordination of Source Security and Target Security" on page 246. This section discusses DataJoiner's mapping of IDs on the level of replication administration tool preference connections, DataJoiner catalog tables, and Apply program password files.

When accessing a DB2 database, the replication administration tool or the Apply program must map their user ID and password to a user ID and password in the DB2 database. When accessing a non-IBM database, the replication administration tool and the Apply program must map their user ID and password to DataJoiner. And DataJoiner must then map the user ID and password, through user mapping, to non-IBM database. [9]

The replication administration tool user ID and Apply user ID must have sufficient authority:

- In DataJoiner, to work with replication control tables and nicknames.
- In non-IBM databases, to work with source, target, and control tables.

The ID that is used in DataJoiner does not have to be the same ID that is used in the non-IBM database because DataJoiner can translate IDs and passwords.

Figure 37 on page 261 shows what happens when the replication administration tool sets up replication to allow data to replicate from a Sybase source to a DB2 for OS/390 target.

[9] If you are using Microsoft SQL Server in your replication environment, do not use alias user IDs. Alias user IDs will not map correctly.

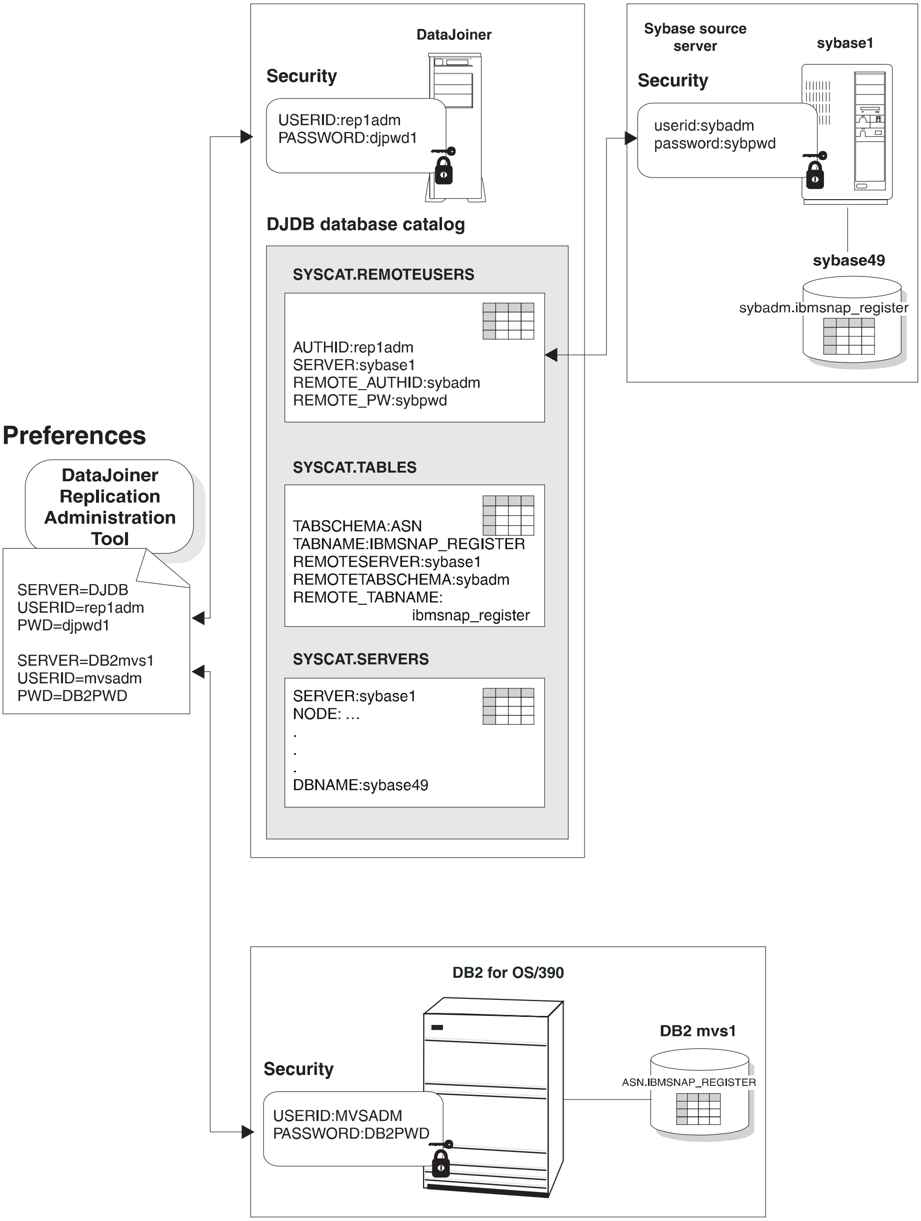
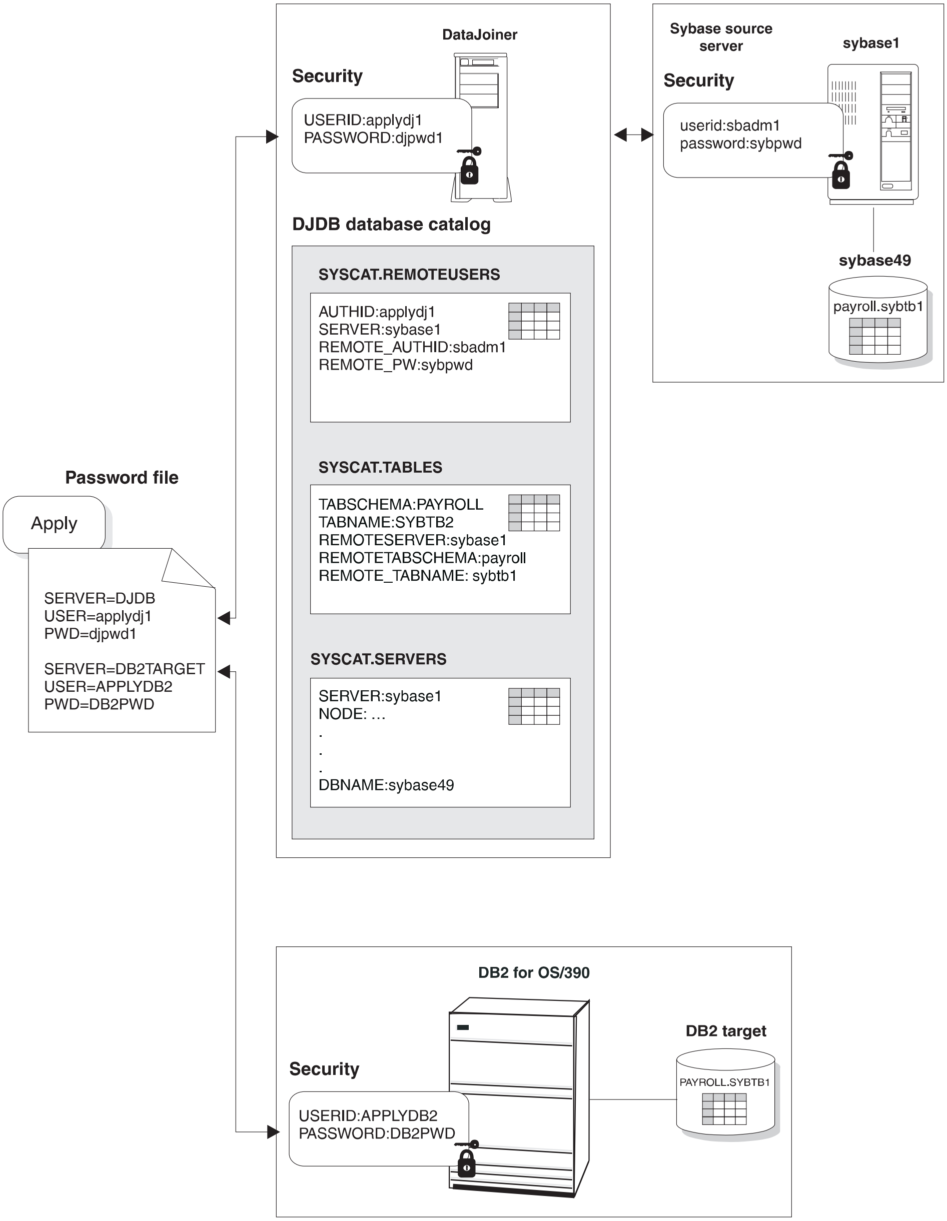


Figure 37. DataJoiner Maps the Replication Administration Tool User IDs to Source and Target. The user ID specified in the replication administration tool is mapped in DataJoiner to a non-IBM source and a DB2 for OS/390 target

In Figure 37, the administrator creates control tables and defines replication sources and targets using the replication administration tool. Mapping occurs between the user ID and password set in the replication administration tool's

Preferences folder and DataJoiner’s SYSCAT.REMOTEUSERS catalog table. DataJoiner then connects to the Sybase database. If the user ID and password are valid, the control tables are placed at the Sybase database. To place control tables at DB2 for MVS, the replication administration tool connects directly to DB2, using the user ID and password set in the Preferences folder.

Figure 38 on page 263 shows what happens when the Apply program performs replication.



USERID:applydj1
PASSWORD:djpwd1

userid:sbadm1
password:sybpwd

AUTHID:applydj1
SERVER:sybase1
REMOTE_AUTHID:sbadm1
REMOTE_PW:sybpwd

TABSCHEMA:PAYROLL
TABNAME:SYBTB2
REMOTESERVER:sybase1
REMOTETABSCHEMA:payroll
REMOTE_TABNAME: sybtb1

SERVER:sybase1
NODE: …
.
.
.
DBNAME:sybase49

SERVER=DJDB
USER=applydj1
PWD=djpwd1

SERVER=DB2TARGET
USER=APPLYDB2
PWD=DB2PWD

USERID:APPLYDB2
PASSWORD:DB2PWD

Figure 38. DataJoiner Maps User ID and Apply Password File to Connect to Source and Target. The Apply program passes user ID and password to DataJoiner, which uses them to access a non-IBM source and a DB2 for OS/390 target.

In Figure 38, the Apply password file specifies the database server, user ID and password. The Apply program connects to the DataJoiner database using the user ID and password from the Apply password file. DataJoiner then searches its SYSCAT.REMOTEUSERS catalog table to find the Sybase user ID and password that is mapped to the connected user. DataJoiner then connects to the Sybase source database that uses the Sybase user ID and password. When replicating changes to DB2 for OS/390, the Apply program connects directly to DB2 for OS/390. The Apply program does not need to connect to a DataJoiner database to access DB2 for OS/390. The Apply program connects through DataJoiner DDCS or DB2 Connect. The entry in the Apply password file must be valid for DB2 for OS/390.

Chapter 26. Setting up the DataJoiner Environment for Replication

This chapter explains how to begin to set up your replication environment while installing and configuring DataJoiner. In this chapter, you:

1. Install DataJoiner.
2. Create a DataJoiner instance and, within the instance, a database for each non-IBM database that is a source for replication. Your target databases can use an existing database within the same DataJoiner instance, or you can create a new database for your targets.
3. Configure access to your data sources (your replication source, target, and control servers).
4. Connect clients to DataJoiner. If the Apply program that you are using for replication is remote from DataJoiner, you must connect the Apply program as a client to DataJoiner.

This chapter explains these steps for DataJoiner for UNIX. After this chapter, you must proceed to "Chapter 27. Installing the Replication Administration Tool and Connecting All Databases" on page 269 to install the replication administration tool and connect all databases that are involved in replication.

Replication in DataJoiner for UNIX

This section describes how to prepare DataJoiner for UNIX for replication.

Installing DataJoiner

Restore product files using installp or smitty by following the steps in "Chapter 5. Installing DataJoiner" on page 41. Installation of some components is optional. Make sure that you install:

- TCP/IP support, if the replication administration tool will access DataJoiner through TCP/IP
- APPC support, if the replication administration tool will access DataJoiner through advanced program-to-program communications (APPC)
- DRDA support, if accessing DB2 for MVS as a source or target server
- The Apply component of DataJoiner

Later, as described in "Chapter 27. Installing the Replication Administration Tool and Connecting All Databases" on page 269, you will install the replication administration tool on your NT workstation.

Setting up an Instance

In “Chapter 6. Setting up an Instance” on page 53, you:

- Create an instance
- Set up environment variables
- Create DataJoiner databases
- Set up the Apply program as a local client

This section focuses on setting up the Apply program and creating DataJoiner databases as pertaining to replication.

Setting up Apply for UNIX as a Local Client

In “Step 3: Set Environment Variables” on page 58, set the environment variables for the user ID that runs the Apply program to be a local client on DataJoiner.

In addition to using the Apply program that is provided with DataJoiner, you can also use other Apply programs on other platforms as discussed in “Connecting Clients to DataJoiner” on page 267.

Creating DataJoiner Databases

In “Step 6: Create a DataJoiner Database” on page 62, you create databases that will be configured to access non-IBM databases as part of your replication system. You must create one DataJoiner database for *each* non-IBM replication *source* server. You can support many non-IBM replication *target* servers with one DataJoiner database. The DataJoiner databases that you set up reside on one DataJoiner instance.

For Oracle as a source:
Use the COLLATE USING parameter within the CREATE DATABASE command. Use IDENTITY.

Configuring Database Connections

Within “Chapter 7. Configuring Access to Data Sources” on page 71, several sections are applicable to replication. The Apply program in DataJoiner uses the connections that you configure in this section to access source, target, and control servers.

Table 14. Configuring Database Connections in DataJoiner: UNIX

If Accessing	You need to	Refer to this section
DB2 for MVS	Catalog DB2 in DataJoiner	“Accessing DRDA Data Sources using DataJoiner DDCS” on page 102.

Table 14. Configuring Database Connections in DataJoiner: UNIX (continued)

If Accessing	You need to	Refer to this section
DB2 UDB V5 or V6, or DB2 CS V2	Catalog DB2 in DataJoiner	"Chapter 9. Accessing DB2RA Data Sources" on page 107.
Informix	Create a server mapping, user mapping, and nickname for Informix, within DataJoiner.	"Chapter 10. Accessing Informix Data Sources" on page 123.
MS SQL Server or Sybase	Create a server mapping, user mapping, and nickname for Microsoft SQL Server or Sybase, within DataJoiner.	"Chapter 12. Accessing Microsoft SQL Server and Sybase Data Sources Using Sybase Open Client" on page 141, or, if applicable, refer to "Chapter 13. Accessing Microsoft SQL Server Sources through MERANT DataDirect Drivers" on page 147.
Oracle	Create a server mapping, user mapping, and nickname for Oracle, within DataJoiner.	"Chapter 11. Accessing Oracle Data Sources Using SQL*Net or Net8" on page 135.

You must define server and user mappings for each non-IBM database (in each DataJoiner database) that requires access to the non-IBM source or target.

Connecting Clients to DataJoiner

"Remote Clients" on page 31 applies to Apply programs installed on DB2 for MVS, DB2 UDB, or DB2 CS that must access non–IBM sources or targets through DataJoiner.

For Apply for MVS, refer to "Chapter 18. Configuring the DataJoiner DRDA Application Server and Clients" on page 177. The chapter describes how to enable applications on MVS, such as Apply for MVS, to access DataJoiner.

For Apply programs on DB2 UDB or DB2 CS, refer to the *Installing and Using Clients* for your platform.

Client software must be installed and configured for each DataJoiner database that is created to replicate with a remote Apply program. Later, in "Configuring Access from the Replication Administration Tool to DataJoiner and DB2" on page 271, you will connect the replication administration tool as a client to DataJoiner.

What to Do Next

Before proceeding to “Chapter 27. Installing the Replication Administration Tool and Connecting All Databases” on page 269, make sure that you install all Capture and Apply programs on the systems of your choice. Refer to “Capture and Apply Program Requirements” on page 256 for a list that is supported by DataJoiner with the replication administration tool. And refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)* for instructions on installing the Capture and Apply programs.

Chapter 27. Installing the Replication Administration Tool and Connecting All Databases

This chapter describes how to install the replication administration tool and complete the setup of your replication environment. In this chapter you:

- Install the replication administration tool on your Windows 95, 98, or NT workstation.
- Configure access from the workstation on which the replication administration tool is installed to each source, target, and control server.
- Set the DB2CODEPAGE environment variable for DataJoiner for UNIX access.
- In the replication administration tool, set replication-administrative preferences.
- From the workstation on which the replication administration tool is installed, bind DB2 utilities and DB2 CLI to all source, target, and control servers.
- Using the replication administration tool, create replication control tables at the source, target, and control servers.
- For the DB2 source server, bind the Capture program to the DB2 source server. Bind the Apply program to each source, target, and control server. Capture triggers, placed at non-IBM databases, need not be bound.
- From where the Apply program is located, set end-user authentication.

Installing the Replication Administration Tool

The replication administration tool includes ObjectREXX as part of the installation. Check to see if you already have ObjectREXX installed and verify the release level. The replication administration tool will replace your downlevel copy of ObjectREXX, or you can choose not to install ObjectREXX and use your existing copy.

If you are installing the replication administration tool on Windows NT and have not installed the replication administration toolas part of the DataJoiner installation process, log into Windows NT with a valid user name that has administrator authority. If you are installing the replication administration tool on Windows 95 or 98, no administrator log on is required. To install the replication administration tool:

1. Insert the DataJoiner CD-ROM into the appropriate drive. Or download DJRA from the Web.[10]
2. Use the setup program:
   a. Open the run window by using one of the following methods:
      - From the Program Manager, select **File** —>**Run**.
      - Select the **Start** button, then select the **Run**.

   b. In the **Command Line** field, type the following command:

      ```
      <cdrom_drive>:\SETUP.exe
      ```

      where *<cdrom_drive>* is the letter that designates your CD-ROM drive.

   c. Select **OK** on the Run window.

   The first Setup window opens.
3. Respond to the setup program's prompts. Online help is available to help you with the remaining steps.
4. When you have completed setup, the replication administration tool appears in the Windows Program directory. To start the replication administration tool:
   a. Click the **Start** icon.

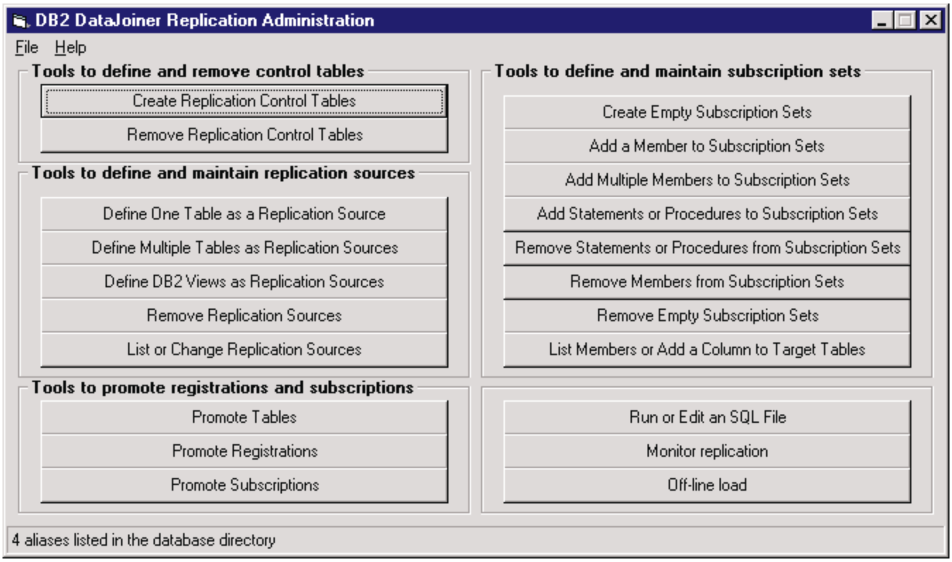
   b. Select the **Programs** menu.

   c. Select the **DataJoiner for Windows** menu.

   d. Select the **Replication** menu.

   e. Select **The Replication Administration Tool**. The replication administration tool primary window opens, as shown in Figure 39 on page 271.

[10] http://www.software.ibm.com/data/dpropr


DB2 DataJoiner Replication Administration
File
Help
Tools to define and remove control tables
Create Replication Control Tables
Remove Replication Control Tables
Tools to define and maintain replication sources
Define One Table as a Replication Source
Define Multiple Tables as Replication Sources
Define DB2 Views as Replication Sources
Remove Replication Sources
List or Change Replication Sources
Tools to promote registrations and subscriptions
Promote Tables
Promote Registrations
Promote Subscriptions
Tools to define and maintain subscription sets
Create Empty Subscription Sets
Add a Member to Subscription Sets
Add Multiple Members to Subscription Sets
Add Statements or Procedures to Subscription Sets
Remove Statements or Procedures from Subscription Sets
Remove Members from Subscription Sets
Remove Empty Subscription Sets
List Members or Add a Column to Target Tables
Run or Edit an SQL File
Monitor replication
Off-line load

4 aliases listed in the database directory

Figure 39. The Replication Administration Tool Primary Window

Configuring Access from the Replication Administration Tool to DataJoiner and DB2

Connect the replication administration tool as a client in DataJoiner to enable the replication administration tool to access source, target, and control servers through DataJoiner. You must connect the replication administration tool to each DataJoiner database that is accessed for replication. Refer to *Installing and Using Clients* for information about configuring clients.

You must also configure database connections from your NT workstation to all DB2 databases involved in replication. Refer to *Installing and Using DB2 Clients for Windows 95 and Windows NT: V2* or the *Installing and Configuring DB2 Clients* for instructions on configuring for DB2 UDB or DB2 client servers. For DB2 for MVS or DB2 for OS/390, refer to “Chapter 18. Configuring the DataJoiner DRDA Application Server and Clients” on page 177, or your *DDCS User’s Guide*, V2.3.

Setting DB2CODEPAGE for DataJoiner for UNIX Access

From your replication administration tool workstation, set the DB2CODEPAGE environment variable when the replication administration tool access DataJoiner for UNIX. The value you set is based on your country code. For example, if your country code is US, you would:

1. Select the **My Computer** icon.
2. Select **Select System** icon
3. From the System Properties folder, select **Environment**.
4. Type in DB2CODEPAGE in **Variable**.
5. For US, type in the value 437 in **User Variables**.

Setting Administrative Preferences

You can specify your preferences for:

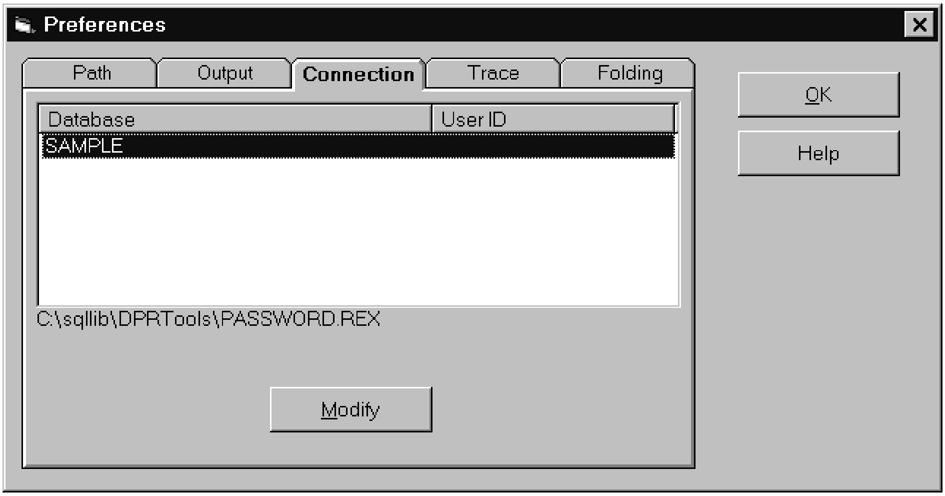
- Location of console or file output (optional)
- User IDs and passwords (required)
- Tracing SQL execution activity (optional)

To set preferences, select **Preferences** from the **File** menu on the replication administration tool primary window. You can change the preferences you set whenever you want to.

On the Connection page of the Preferences notebook, you see a list of databases that are currently catalogued on your system. Select a database, then select the **Modify** pushbutton to set the user ID and password.

If you are using Microsoft SQL Server in your replication environment, do not use an alias user ID. Microsoft SQL Server will reject the alias user ID.

Figure 40 on page 273 shows the Connection page on the Preferences notebook.



Database	User ID
SAMPLE	

C:\sqllib\DPRTools\PASSWORD.REX

Figure 40. The Preferences Notebook: Connection

Binding to Source, Target, and Control Servers

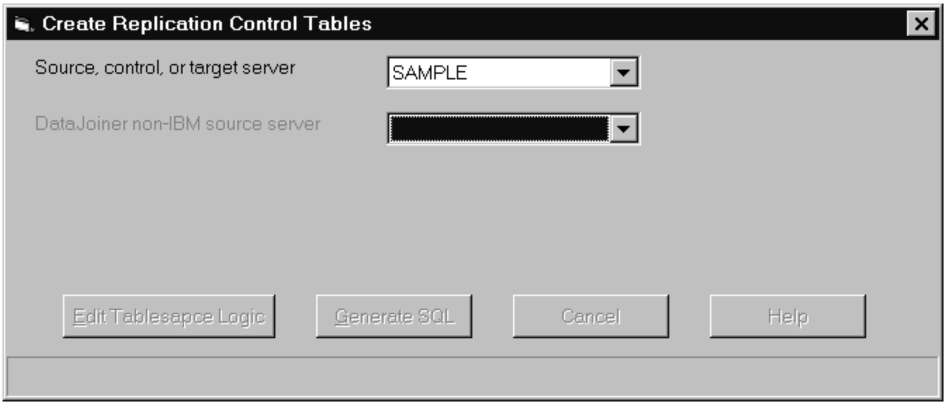
Bind DB2 utilities to each source, target, and control server, including DataJoiner. You must run the **bind** command separately for each DB2 or DataJoiner source, target, or control server. From the replication administration tool workstation, you must bind DB2 utilities and DB2 CLI; if you have a mixture of Windows NT, Windows 95, and Windows 98 workstations you must bind DB2 utilities to each DataJoiner and each DB2 database from at least one Window 95 or 98 workstation and at least one Window NT workstation. An example of what you enter is:

```
cd c:\sqllib\bnd
db2 connect to data-server-db
db2 bind @db2ubind.lst blocking all grant public
```

Creating Replication Control Tables

Create control tables at each DataJoiner and DB2 system involved in replication. When you complete this step, the replication administration tool places a register table and a pruning control table at the non-IBM database source and creates nicknames for these tables in DataJoiner.

From the replication administration tool primary window, click the **Create Replication Control Tables**. The created window opens Figure 41 on page 274.



Create Replication Control Tables

Source, control, or target server: SAMPLE

DataJoiner non-IBM source server:

Figure 41. Creating Replication Control Tables.

The fields you complete in order to create a control table are:

Source, control, or target server

When you click the down arrow, the replication administration tool checks to see what type of server it is and then lists all servers available to your user ID and workstation. The list includes DataJoiner servers and DB2 servers. If you select a DataJoiner server from the list, the DataJoiner non-IBM source server pulldown list becomes active. If you do not choose a DataJoiner server, you will link directly to a DB2 target database.

DataJoiner non-IBM source server

If DataJoiner is the replication source server, identify the name of the non-IBM data source that is used in the replication. The pull down list contains DataJoiner server names. The replication administration tool generates the pull down list according to the database name you provided in the **Source, control, or target server** field. Specify **(None)** if you are using the DataJoiner server as a target or control server, but not a source server.

Edit Tablespace Logic

Click on this button to customize table space names for control tables or for CREATE TABLESPACE options. The default table space names for DB2 for OS/390 are:

- TS_UOW for the UOW table
- TS_CNTL for all other control tables

For other platforms, the default table space name is TS*nnnnnn*, where *nnnnnn* is a unique identifier.

Generate SQL

Click this push button after you supply all the information on this panel you need to generate SQL. While the SQL is being generated, a window is displayed showing processing messages and error messages, if any.

When the procedure completes successfully, save the file by selecting Save from the File pull down menu. You can now edit the generated SQL, according to the guidelines that are listed in "Editing the Replication Administration Tool-Generated SQL" on page 281. When you are ready, run the SQL by selecting Run from the File pull down menu. You must save the generated SQL before you can run the SQL. You must run the SQL for generating control tables before you generate and run SQL to create replication sources or subscriptions.

Binding the Capture and Apply Programs in DB2 Systems

Before you bind your Capture and Apply programs, make sure that you have created control tables as described in "Creating Replication Control Tables" on page 273.

From the DB2 source server, you must bind the Capture program to the DB2 source server. From each DataJoiner and DB2 source, target, and control server, you must bind the Apply program.

For more information on binding the Apply program in DataJoiner, refer to "Finalizing the Setup for the Apply Program in UNIX". For more information on binding other Capture and Apply programs, refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)*.

Finalizing the Setup for the Apply Program in UNIX

This section describes how to do the following tasks in the UNIX environment:
- Bind the Apply program.
- Provide end-user authentication.

Binding the Apply Program in UNIX

If you create several DataJoiner databases on a DataJoiner instance, you must bind Apply to each of the DataJoiner databases. To bind the Apply program to source, target, and control servers:

1. Ensure that the user ID you are using to bind the Apply program has the required privileges. See "Authorization Requirements" on page 256.

2. Logon to the UNIX server on which DataJoiner is running using the Apply user ID from the target server system.
3. Change to the directory where the Apply program bind files are located, which is in the DataJoiner instances directory: home/sqllib/bnd.
4. Connect to the source server database by entering:

   ```
   DB2 CONNECT TO database
   ```

   where *database* is the source server database.

   If the source is a non-IBM database, the Apply program considers DataJoiner as the server database.
5. Create and bind the Apply program packages to the source server database by entering both of the following commands:

   ```
   db2 bind @applycs.lst isolation cs blocking all
   db2 bind @applyur.lst isolation ur blocking all
   ```

   Where:

   CS The list in cursor stability format.

   UR
   The list in uncommitted read format.

   These commands create a list of packages. The names of the packages can be found in the applycs.lst and applyur.lst files.
6. Repeat the connect and bind steps for each server that the Apply program connects to. You must bind the Apply program to the source, target, and control servers.

   Because the Apply program control tables use static SQL calls, the Apply bind process searches for nearly all of the control tables at each server that it is bound to, regardless of whether or not the control tables are used at each server.

For more information on binding Apply programs, refer to the *IBM DB2 Replication Guide and Reference (V5)* or the *IBM DB2 Replication Guide and Reference (V6)*.

Providing End-User Authentication in UNIX

Remote DB2 systems involved in replication might require that the Apply program running locally on DataJoiner for UNIX provide a password or use a user ID different from the ID used on the local DataJoiner server. The Apply password file allows you to connect to the remote DB2 and remote DataJoiner systems. Create the password file on the local DataJoiner system if your remote DB2 and DataJoiner system are cataloged on the local DataJoiner system with AUTHENTICATION=SERVER. If your remote DB2 and

DataJoiner system are cataloged with AUTHENTICATION=CLIENT, you do not need to provide a password file. You can also use the password file to use different user IDs on your remote DB2 system than what you use on your local DataJoiner system.

For more information about authentication and security, refer to the *IBM DataJoiner Administration Supplement* or the *IBM DB2 Universal Database Administration Guide*.

If you create a password file:

The password file must:

- Be named:

  ```
  APPLYQUALinstnameCNTLSRVR.PWD
  ```

  Where:

  APPLYQUAL
  : The Apply qualifier, in uppercase.

  instname
  : The DataJoiner instance name, in lowercase, that Apply program runs under. The value of DB2INSTANCE.

  CNTLSRVR
  : The name of the control server, in uppercase

  For example: `DATADIRapply1REPRTDB.PWD`.

  The naming convention that is used here is the same as the log file name (.LOG) and the spill file name (.SPL), but with a file extension of PWD.
- Reside in the directory from which the Apply program starts.
- Contain all server-name / user-name / password sets for the file. The user name is optional. The password file enables you to use a different (or the same) password at each server.
- Contain one or more records in the following format:

  ```
  SERVER=server_name USER=user_name  PWD=password
  ```

  or

  ```
  SERVER=server_name  PWD=password
  ```

  Do not include blank lines or comment lines.

We recommend that you limit read access of this file to the user ID that runs the Apply program.

If you do not create a password file:

The Apply program in DataJoiner must be able to issue an SQL CONNECT statement without specifying the user ID and password. Ensure that:

- The DB2 for MVS database is cataloged as AUTHENTICATION=CLIENT.
- The logon ID belongs to PRIMARY GROUP=SYSTEM.

When copying from DB2 for MVS sources, ensure that:

- SECURITY=SAME for MVS CPI-C node.
- You specify the following values when you define the LU name that uses the VTAM APPL:
  - VERIFY=NONE to indicate that any logical unit (LU) can request an LU-LU session.
  - SECACPT=ALREADYV to indicate user ID and password checking at the requester.

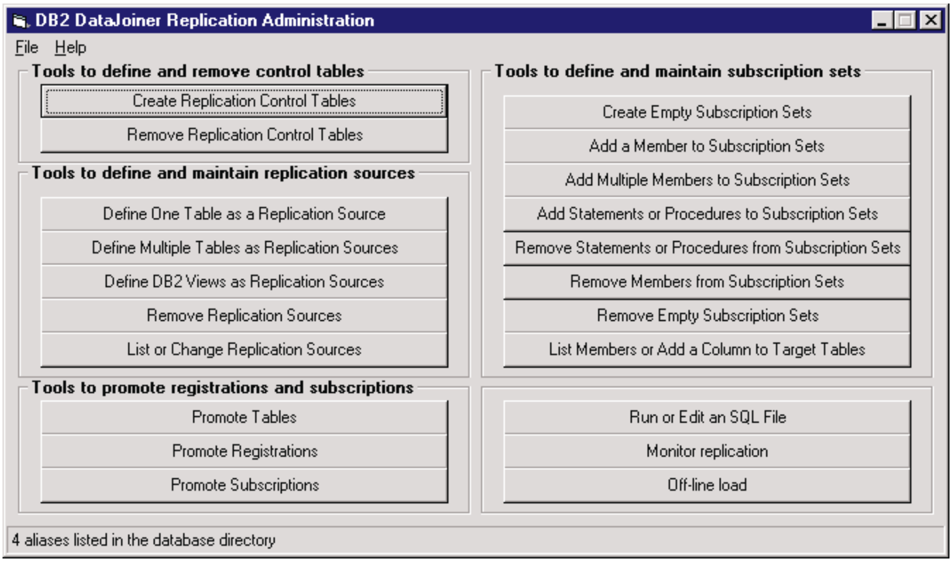
Chapter 28. Starting and Using the Replication Administration Tool

This section describes how to begin running the replication administration tool and describes the basic process of setting up replication.

Starting the Replication Administration Tool

To start the replication administration tool:

1. To start the replication administration tool first start all databases involved in replication. Then:
   a. Click the **Start** icon on the Windows 95, 98, or NT desktop.
   b. Select the **Programs** menu.
   c. Select the **DataJoiner for Windows** menu.
   d. Select the **Replication** menu.
   e. Select **The Replication Administration Tool**. The replication administration tool primary window opens, as shown in Figure 42.


DB2 DataJoiner Replication Administration
File
Help
Tools to define and remove control tables
Create Replication Control Tables
Remove Replication Control Tables
Tools to define and maintain replication sources
Define One Table as a Replication Source
Define Multiple Tables as Replication Sources
Define DB2 Views as Replication Sources
Remove Replication Sources
List or Change Replication Sources
Tools to promote registrations and subscriptions
Promote Tables
Promote Registrations
Promote Subscriptions
Tools to define and maintain subscription sets
Create Empty Subscription Sets
Add a Member to Subscription Sets
Add Multiple Members to Subscription Sets
Add Statements or Procedures to Subscription Sets
Remove Statements or Procedures from Subscription Sets
Remove Members from Subscription Sets
Remove Empty Subscription Sets
List Members or Add a Column to Target Tables
Run or Edit an SQL File
Monitor replication
Off-line load

4 aliases listed in the database directory

Figure 42. The Replication Administration Tool Primary Window

General Steps for Setting up Replication

Generally, each function in the replication administration tool consists of five steps:

1. Select the function you want (for example, **Define One Table as a Replication Source** or **Add a Member to Subscription Sets**). A window opens.
2. If applicable, edit the replication administration tool logic to affect the statements that are created when you generate SQL by selecting the **Edit Logic**, **Edit Predicate Logic**, or **Edit Create Table Logic** button from the replication administration tool function window. See "Editing the Replication Administration Tool Logic" on page 281 for more information.
3. Generate the SQL.
   1. Provide the required information, as prompted in the displayed window.

      Where wildcards are allowed, use either a string with no (%) wildcards, or a pattern string with one or more (%) wildcards, with one trailing (%) wildcard. The (%) symbol matches a substring of none or many arbitrary characters. When a pattern string contains a (%) wildcard, any (_) characters also in the string are interpreted as single-character wildcards.
   2. Create the SQL by selecting the **Generate SQL** button. SQL is generated into a file but not run yet.
   3. Save the file.
4. Review and edit, if needed, the generated SQL file from the console window or by selecting the **Run or Edit an SQL file** button from the primary window.

   DataJoiner's pass-through and user mapping functions *require* that you use only the replication administration tool to run the SQL. See "Editing the Replication Administration Tool-Generated SQL" on page 281 for more information.
5. Run the SQL file by selecting the **Run or Edit an SQL file** button and then selecting the **Run** button from the primary window.

   Make sure that you run your generated SQL files before generating another set of SQL files. If you do not run your SQL after it is generated, the replication administration tool might generate duplicate names for objects. See "Running the Replication Administration Tool-Generated SQL" on page 282 for more information.

Editing the Replication Administration Tool Logic

Three files are available for edit: SRCESVR.REX, TARGSVR.REX, and CNTLSVR.REX.

SRCESVR.REX
Specifies the owner and name of the CD or CCD table and the table space in which the CD or CCD table is placed. Select the function to define one or multiple tables as replication sources and click **Edit Logic** to edit SRCESVR.REX before you generate SQL.

CNTLSVR.REX
Specifies the criteria by which rows from the source table can be replicated to the target. You can specify which columns in a source table are eligible for replication. You also can specify values to search individual source tables and replicate only the data that matches the value criteria. Select the function to add one or multiple members to subscription sets and click **Edit Predicate Logic** to edit the CNTLSVR.REX file.

TARGSVR.REX
Specifies the table space or segment in which to create target tables. Check this file to make sure that the table spaces are being defined in the location of the target database where you want them. Select the function to add one or multiple members to subscription sets and click **Edit Target Table Logic** to edit the TARGSVR.REX file.

TBLSPACE.REX
Specifies the table spaces for the control tables and the UOW table. Check this file to make sure that the table spaces are being defined in the location where you want them. Select the function to create replication control tables and and click **Edit Table Logic** to edit the TBLSPACE.REX file.

Note: You can specify where table spaces are created in the SRCESVR.REX and TARGSVR.REX by editing the default directory `C:\` to the directory you prefer. When you type in your directory, make sure you add a backslash (`\`) after the directory. For example, if you are changing the directory from `C:\` to `F:\Test\`, make sure that you enter a slash before the word `Test` and after the word `Test`. If you just enter `F:\Test`, an error will occur when you attempt to generate SQL.

Editing the Replication Administration Tool-Generated SQL

You can edit the replication administration tool’s SQL from the console window or by selecting the **Run or Edit an SQL file** button from the primary window. You can edit the SQL for several reasons. For example, you might want to:

- Review and edit the parameters for the CCD tables that are used in the definition of a replication source.

  For example, if the two-part names of the CD or CCD tables are altered, the values of columns CD_OWNER, CD_TABLE and CCD_OWNER, CCD_TABLE specified in the ASN.IBMSNAP_REGISTER table row must be changed to match the changes made to the actual table names.
- Edit create table and index statements to represent clusters and other database objects.
- For Oracle, ensure that tables are created in the existing table spaces that you want.
- For Microsoft SQL, create control tables on an existing segment.
- Review and edit subscription member predicates as a way of defining multiple subscriptions at one time. You can use substitute variables in your predicates and resolve the variables with programming logic.

When editing generated SQL, be careful not to change special markers that the replication administration tool places within the SQL. For example, :ENDOFTRIGGER: or :ENDOFPROCEDURE: is part of a comment that is necessary for the replication administration tool to run successfully. Altering create trigger blocks can result in incorrect SQL that ends in error when run.

Running the Replication Administration Tool-Generated SQL

The **Run SQL** button on the replication administration tool GUI is intended to be used for SQL generated by the replication administration tool. SQL you generate outside the replication administration tool might not run successfully if you use the replication administration tool to start it. Likewise, you might not be able to run SQL generated by the replication administration tool at a DB2 command line. We recommend running the replication administration tool-generated SQL through the replication administration tool's GUI.

Running Capture and Apply

After you have defined replication sources and created subscription sets and members, you are ready to run the Capture and Apply programs. For more information on running Capture and Apply programs, refer to the *IBM DB2 Replication Guide and Reference (V5)* or the the *IBM DB2 Replication Guide and Reference (V6)*. Capture triggers begin running automatically.

Operating Apply for UNIX platforms

An administrator can use the commands in the following sections to perform the following Apply program tasks:

- Starting

- Scheduling
- Stopping

Before you start the Apply program

Before you start the Apply program, ensure that:

- You have the proper authorization. See *IBM DB2 Replication Guide and Reference (V6)* for information about authorization for the Apply program.
- The control tables are defined.
- At least one subscription is created and activated.
- The Apply package is created.
- A password file has been created, if necessary, for end-user authentication at the source server. See *IBM DB2 Replication Guide and Reference (V6)* for more information.
- The Capture program is started, and the ASN0100I initialization message was issued (if you are running a Capture program).

Starting Apply for UNIX platforms

After you start the Apply program, it runs continuously until:

- You stop it in an orderly way.
- You cancel it.
- An unexpected error or failure occurs.

To start the Apply program on UNIX:

1. Log on with the user ID that will run the Apply program.
2. Ensure that you set the DB2 instance name as shown:
   ```
   export DB2INSTANCE=db2_instance_name
   ```
3. Set the LIBPATH and language environment variables or edit the .profile file in the same environment in which the Apply program starts.

   AIX example:
   ```
   export LIBPATH=db2instance_home_directory/sqllib/lib:/usr/lib:/lib
   export LANG=en_US
   ```

   HP-UX example:
   ```
   export SHLIB_PATH=db2instance_home_directory/sqllib/lib:/usr/lib:/lib
   export LANG=en_US
   ```

   Solaris and UnixWare 7 example:
   ```
   export LD_LIBRARY_PATH=db2instance_home_directory/sqllib/lib:/usr/lib:/lib
   export NLS_PATH=˜/usr/lib/locale/%L/%N:db2instance_home_directory/sqllib/msg/en_US/%
   export LANG=en_US
   ```

where *db2instance_home_directory* is the name of the DB2 instance's home directory.

4. Enter the **asnapply** command and options:

```
►►──asnapply──apl_qual──┬───────────┬──┬─────────────┬──┬────────────┬──┬───────────┬──────────►
                        └─ctrl_serv─┘  ├─loadxit─────┤  ├─inamsg─────┤  ├─notrc─────┤
                                       └─noloadxit───┘  └─noinamsg───┘  ├─trcerr────┤
                                                                        └─trcflow───┘

►──┬─────────┬──┬────────────┬──┬───────────┬──┬──────────┬──┬──────────┬──┬──────────┬──────►
   └─trcfile─┘  ├─notify─────┤  ├─sleep─────┤  └─delay(n)─┘  └─copyonce─┘  └─logreuse─┘
                └─nonotify───┘  └─nosleep───┘

►──┬───────────┬──┬──────────┬──────────────────────────────────────────────────────────────►◄
   └─logstdout─┘  └─trlreuse─┘
```

Table 15 defines the invocation parameters.

Table 15. ASNAPPLY invocation parameter definitions for UNIX platforms

Parameter	Definition
apl_qual	Specifies the Apply qualifier that the Apply program instance uses to identify the subscriptions to be served. The Apply qualifier is case sensitive and must match the value of APPLY_QUAL in the subscription set table. This must be the first parameter.
ctrl_serv	Specifies the name of the server where the replication control tables will reside. If you do not specify this parameter, the default is the default database or the value of DB2DBDFT.
loadxit	Specifies that the Apply program is to invoke ASNLOAD, an IBM-supplied exit routine that uses the export and load utilities to refresh target tables.
noloadxit (default)	Specifies that the Apply program will not invoke ASNLOAD.
inamsg (default)	Specifies that the Apply program is to issue a message when the Apply program is inactive.
noinamsg	Specifies that the Apply program will not issue this message.
notrc (default)	Specifies that the Apply program does not generate a trace.
trcerr	Specifies that the Apply program generates a trace that contains only error information.

Table 15. ASNAPPLY invocation parameter definitions for UNIX platforms (continued)

Parameter	Definition
trcflow	Specifies that the Apply program generates a trace that contains both error and execution flow information.
trcfile	If both trcfile and trace are specified, the Apply program writes trace output to the trace file (*.trc). If you do not specify this option, the Apply program sends trace output to the standard output, stdout.
notify	Specifies that the Apply program is to invoke ASNDONE, an exit routine that returns control to the user when the Apply program processing ends.
nonotify (default)	Specifies that the Apply program will not invoke ASNDONE.
sleep (default)	Specifies that the Apply program is to go to sleep if no new subscriptions are eligible for processing.
nosleep	Specifies that the Apply program is to stop if no new subscriptions are eligible for processing.
delay(*n*)	Where *n*=0, 1, 2, 3, 4, 5, or 6. Specifies the delay time (in seconds) at the end of each Apply program cycle when continuous replication is used. The default delay time is 6 seconds.
copyonce	The Apply program executes one copy cycle for each eligible subscription set and then terminates. An eligible subscription set is such that: • ACTIVATE > 0 • REFRESH_TIMING = R or B or REFRESH_TIMING = E and the specified event has occurred. MAX_SYNCH_MINUTES and END_OF_PERIOD are honored if specified.
logreuse	The Apply program reuses the log file (*.app) by first deleting it and then re-creating it when the Apply program is restarted. If you do not specify this option, the Apply program appends messages to the log file, even after the Apply program is restarted.
logstdout	The Apply program sends all messages to both the standard output (stdout) and the log file.
trlreuse	The Apply program empties the Apply trail table when the Apply program is started.

Scheduling Apply for UNIX platforms

Use the **at** command to start the Apply program at a specific time. For example, the following command starts the Apply program at 3:00 p.m. on Friday:

```
at 3pm Friday asnapply myqual
```

Stopping Apply for UNIX platforms

Use the **asnastop** command or a key combination to stop the Apply program in an orderly way.

```
►►──asnastop──apply_qualifier────────────────────────────────────────────►◄
```

To use the command, do the following from a window where the Apply program is not running:

1. Set environment variable DB2INSTANCE to the value set when the Apply program was started.
2. Set environment variable DB2DBDFT to the source server specified when the Apply program was started (or the DB2DBDFT value used when the Apply program was started).
3. Enter the command.

Chapter 29. DataJoiner with the Replication Administration Tool: Data Typing

When you are defining replication sources for non-IBM database tables, data typing often requires extra steps. When the source table is a DataJoiner nickname, DataJoiner handles any data-typing transformations when the nickname of the source table is created outside of the replication administration tool. For more information see the *DataJoiner Application Programming and SQL Reference Supplement*.

During the subscription process, the final data mappings occur. When the target is a table that is accessed through a DataJoiner nickname, the DataJoiner nickname process does not always create the correct type, because:

- DataJoiner requires that the target table be created in the non-IBM database *before* a nickname is created on DataJoiner.
- When the nickname is created, DataJoiner chooses a data type known to DB2 common server that maps to the largest possible value in the target table.
- Sometimes the largest value from the source table (typically in DB2 for MVS), is not the largest value that you can store in the target column.
- The replication administration tool generates ALTER NICKNAME statements for the COLTYPE, SCALE, and LENGTH in the DataJoiner database for the target table nickname. For Oracle, Sybase, and MS SQL Server, the replication administration tool selects the data types in the DataJoiner database. DataJoiner can then perform a conversion when accessing the target table. For Informix, no column modifications are required.

Using the replication administration tool, you can create a subscription in which your non-IBM database is both the source server and target server. When the replication administration tool creates the nickname for the target table, DataJoiner creates the same data types that it generated for the source so that data types match.

In general, you should let the replication administration tool create the target table and ALTER NICKNAME statements. Otherwise, you must take care to choose the same data-type mappings that DataJoiner would generate if you want to create your own target tables on non-IBM databases.

This section discusses how the replication administration tool with DataJoiner handles data typing in three specific scenarios.

DB2 to Oracle Replication

The source is DB2 and the target is Oracle, the following restrictions and conversions are performed:

Table 16. Data Type Conversion: DB2 to Oracle

DB2 for MVS Source data type	Oracle Target data type	Changes after Create Nickname		
		COLTYPE	LENGTH	SCALE
FOR BIT DATA(n)	RAW(n)			
CHAR (<256)	CHAR(n)			
CHAR (>=256) VARCHAR2(n)	VARCHAR2(n)			
VARCHAR(n)	VARCHAR2(n)			
GRAPHIC(n)	CHAR(n)			
VARGRAPHIC(n)	VARCHAR(n)			
LONG VARCHAR	LONG			
DATE	DATE	DATE	4	
TIMESTAMP	DATE			
TIME	DATE	TIME	3	
SMALLINT	NUMBER(5)	DECIMAL	5	0
INTEGER	NUMBER(10)	DECIMAL	10	0
DECIMAL(n,m)	DECIMAL(n,m)	DECIMAL	n	m
FLOAT	FLOAT			

DB2 to Informix

When the source is DB2 and the target is Informix, the following restrictions and conversions are performed:

Table 17. Data Type Conversion: DB2 to Informix

DB2 for MVS Source data type	Informix Target data type	Changes after Create Nickname		
		COLTYPE	LENGTH	SCALE
FOR BIT DATA(n)	CHAR(n) [1]			
CHAR(n)	CHAR(n)			
VARCHAR(<256)	VARCHAR(n)			
VARCHAR(>=256)	TEXT			
GRAPHIC(n)	BYTE			
VARGRAPHIC(n)	BYTE		No updates occur after create nickname	

Table 17. Data Type Conversion: DB2 to Informix (continued)

DB2 for MVS Source data type	Informix Target data type	Changes after Create Nickname		
		COLTYPE	LENGTH	SCALE
LONG VARCHAR	TEXT			
DATE	DATE			
TIMESTAMP	DATETIME YEAR TO FRACTION(5)			
TIME	DATETIME HOUR TO SECOND			
SMALLINT	SMALLINT			
INTEGER	INT			
DECIMAL(n,m)	DECIMAL(n,m)			
FLOAT	FLOAT			

Note: [1]For CHAR data in Informix, a string is terminated when the first non-printable character is encountered. In this example, the CHAR FOR BIT data from DB2/MVS could be truncated when stored in Informix, making it appears as though an x'00' is part of the string.

DB2 to Microsoft SQL Server, Sybase, or Sybase SQL Anywhere

When the source is DB2 and the target is MS SQL Server, Sybase, or Sybase SQL Anywhere, the following restrictions and conversions are performed:

Table 18. Data Type Conversion: DB2 to MS SQL Server, Sybase, or SQL Anywhere

DB2 for MVS SOURCE	MS SQL SERVER, SYBASE, OR SQL ANYWHERE TARGET	Changes after Create Nickname		
		COLTYPE	LENGTH	SCALE
CHAR (n) FOR BIT DATA	BINARY(n)	CHAR [1]		
VARCHAR(n) FOR BIT DATA(N)	VARBINARY(n) or BINARY(n)			
CHAR(n)	CHAR(n)			
VARCHAR(<256)	VARCHAR2(n) or VARCHAR(n)			
VARCHAR(<256)	TEXT			
GRAPHIC(n) [2]	?			

Table 18. Data Type Conversion: DB2 to MS SQL Server, Sybase, or SQL Anywhere (continued)

DB2 for MVS SOURCE	**MS SQL SERVER, SYBASE, OR SQL ANYWHERE TARGET**	**Changes after Create Nickname**		
		COLTYPE	**LENGTH**	**SCALE**
VARGRAPHIC(n) [2]	?			
LONG VARCHAR [2]	?			
DATE	DATETIME or DATE	DATE	4	
TIMESTAMP	DATETIME or DATESTAMP			
TIME	DATETIME or TIME	TIME	3	
SMALLINT	SMALLINT			
INTEGER	INT			
DECIMAL(n,m)	DECIMAL(n,m)			
FLOAT	FLOAT			

Note: [1]The replication administration tool creates the target table in MS SQL Server with data type ″binary.″ The DataJoiner nickname is created with COLTYPE of VARCHAR. The replication administration tool updates the COLTYPE to CHAR.
Note: [2]These DB2/MVS data types have not been tested.

DB2 to Microsoft Jet

When the source is DB2 and the target is Microsoft Jet, the following restrictions and conversions are performed:

Table 19. Data Type Conversion: DB2 to Microsoft Jet

DB2 for MVS SOURCE	**MS JET TARGET**	**Changes after Create Nickname**		
		COLTYPE	**LENGTH**	**SCALE**
CHAR (n) FOR BIT DATA	Memo or OLE Object[1]			
VARCHAR(n) FOR BIT DATA(N)	Memo or OLE Object[1]			
CHAR(n)	Text or Memo[1]			
VARCHAR(<256)	Text or Memo[1]			
GRAPHIC(n) [2]	?			
VARGRAPHIC(n) [2]	?			

Table 19. Data Type Conversion: DB2 to Microsoft Jet (continued)

DB2 for MVS SOURCE	MS JET TARGET	Changes after Create Nickname		
		COLTYPE	**LENGTH**	**SCALE**
LONG VARCHAR [2]	Memo			
DATE	Datetime			
TIMESTAMP	Datetime			
TIME	Text			
SMALLINT	Number (Integer)			
INTEGER	Number (Long Integer)			
DECIMAL(n,m), NUMERIC	Number (Integer), Number (Long Integer), Number (Double), or Text[1]			
PRECISION, FLOAT, DOUBLE	Number (Double)			

Note: [1]The value used depends on the precision set for the field.
Note: [2]These DB2/MVS data types have not been tested.

Chapter 30. Non-IBM Databases: Table Names, Control Tables, and the CCD

This section describes how the replication control, source, and CCD table names in a non-IBM database and their associated nicknames are determined.

Table Names and Qualifiers

Both a table and a DataJoiner nickname must exist for any non-IBM table that you attempt to define as a replication source. The unqualified table and DataJoiner nicknames must match, though the table and nickname qualifiers can differ.

Control Tables in a Non-IBM Database

The Apply program refers to the register control table and pruning control table by using ASN as the qualifier. Creating tables with the ASN qualifier in the non-IBM database assumes that an ASN schema exists and that the person creating the control tables has authority to create tables in the ASN schema. To compensate for the assumptions, when you select **Create Replication Control Tables**, the replication administration tool creates the control tables in the non-IBM database using the remote ID stored in DataJoiner of the user as the table qualifier, but always using ASN as the DataJoiner nickname qualifier for the control tables.

CCD Table and Capture Triggers

There are two parts to the CCD table name: the unqualified name and the qualifier.

The default name is CCD followed by a series of character-digits. The default digit-string is derived from a timestamp. You can set the CCD table name by assigning a string to OUT.CD_TABLE (this program variable is used for either CD or CCD table names) in the SRCESVR.REX file.

The default CCD qualifier is the table qualifier (not nickname qualifier) of the non-IBM source table. You can set the qualifier by assigning a string to OUT.CD_OWNER in the SRCESVR.REX file. Use caution when changing the CCD qualifier because triggers perform inserts to the CCD tables, and they are created using the register and pruning control table qualifiers; you have to ensure the triggers have the proper authorizations. IBM recommends not changing the CCD qualifier for non-IBM databases.

The Capture and pruning triggers refer to and use the qualifiers for the register control table and pruning control table, not the ASN nickname qualifiers. Capture triggers, referring to source tables and the CCD tables, use the table qualifiers, not the DataJoiner nickname qualifiers.

CCD Nicknames in DataJoiner

The default CCD nickname qualifier is the qualifier of the nickname for the non-IBM source table. If OUT.CD_OWNER is set in the SRCESVR.REX file, the setting overrides the default and is used as the CCD nickname qualifier. The unqualified name of the CCD table (which you can set in SRCESVR.REX) uses the unqualified portion of the DataJoiner nickname for the CCD table.

The Register and Pruning Control Tables

The ASN.IBMSNAP_REGISTER and ASN.IBMSNAP_PRUNCNTL nicknames refer to tables in the non-IBM database with the same unqualified names with probably a different qualifier, for example, MYUSERID.IBMSNAP_REGISTER. Even though the register and pruning control tables reside in a non-IBM database, references to the source and CCD tables in the register control table rows use the nicknames, not the table names, so that the Apply process operates entirely in the DataJoiner name space. The Apply process is not aware that a particular source table is a table in a non-IBM database.

Part 7. Appendixes

Appendix A. DRDA Application Server

Using DataJoiner as a DRDA AS enables DataJoiner to function as a database server for Application Requesters that use the DRDA protocol, as well as for other clients that use DB2 private protocols; the AS option provides support for DRDA level 1. With the AS option installed, applications that use DRDA Application Requesters (AR) can create packages at the DataJoiner server and subsequently execute them subject to the SQL that is supported at the server.

For more information about DRDA concepts and a detailed description of DRDA commands and bind options, refer to the following publications:

- *Distributed Relational Database Architecture Reference* (SC26-4651-01)
- *DB2 for MVS Connections with AIX or OS/2* (SG24-4558-00)

Because DRDA Application Servers and Application Requesters communicate using the APPC communication protocol, the Systems Network Architecture (SNA) support option (`djx_02_01.sna_clients`) must also be installed.

Supported DRDA Functions

In DRDA, a function is categorized as either required or optional. Table 20 identifies which functions (required and optional) are implemented in the AS on DataJoiner, and Table 21 on page 298 lists the bind option values that are supported.

Table 20. DataJoiner DRDA Application Server Function

Description	Required (R) Optional (O)	Supported	Not Supported
DRDA level 1 required function	R	X (Note 1)	
Rebind	O		X
Describe user privileges	O		X
Describe RDB table	O		X
Interrupt RDB request	O		X

Note:

(1) Certain required functions are not supported.

Bind Options Supported by the DataJoiner DRDA Application Server

Table 21 shows the BIND options that are supported by the DataJoiner DRDA Application Server.

Table 21. Bind Options Supported by the DataJoiner DRDA Application Server

Bind Option	Value	Supported?	DB2 for MVS Precompile Option (Note 1)	DB2/VM Preprocessing Option	OS/400 Precompile Option	DDCS Prep or Bind Option
Package Version Name	**Null**	No	VERSION			VERSION
	Any other value	No				
Bind Existence Checking	**Object existence optional**	No	VALIDATE (**RUN**)[b]	**NOEXIST**	GENLVL(**10**, 11-40)	VALIDATE **RUN**
	Object existence required	Yes	VALIDATE (BIND)[b]	EXIST	GENLVL(00-09)	VALIDATE BIND
Package Replacement Option	**Replacement allowed**	No	ACTION (**REPLACE**)	**REPLACE**	REPLACE (***YES**)	ACTION **REPLACE**
	Replacement not allowed	No	ACTION (ADD)	NEW	REPLACE (*NO)	ACTION ADD
Package Authorization Option	**Keep authorizations**	Yes		**KEEP**		RETAIN **YES**
	Revoke authorizations	No		REVOKE		RETAIN NO
Statement String Delimiter (Note 2)	Apostrophe	Yes	**APOSTSQL**	**SQLAPOST**	OPTION([...] *APOSTSQL) (Note 3)	STRDEL APOSTROPHE
	Double quote	No	QUOTESQL	SQLQUOTE	OPTION([...] *QUOTESQL) (Note 4)	STRDEL QUOTE
Statement Decimal Delimiter (Note 5)	Period	Yes	**PERIOD**	**PERIOD**	OPTION([...] *PERIOD) or OPTION([...] ***SYSVAL**) (Note 6)	DECDEL PERIOD

Table 21. Bind Options Supported by the DataJoiner DRDA Application Server (continued)

Bind Option	Value	Supported?	DB2 for MVS Precompile Option (Note 1)	DB2/VM Preprocessing Option	OS/400 Precompile Option	DDCS Prep or Bind Option
	Comma	No	COMMA	COMMA	OPTION([...] *COMMA) or OPTION([...] ***SYSVAL**) (Note 6)	DECDEL COMMA
Date Format (Note 7)	**ISO**	Yes	DATE (ISO) (Note 8)	DATE (**ISO**)	DATFMT (*ISO) (Note 8)	DATETIME **ISO** (Note 9)
	USA	Yes	DATE (USA)	DATE(USA)	DATFMT (*USA)	DATETIME USA
	EUR	Yes	DATE (EUR)	DATE (EUR)	DATFMT (*EUR)	DATETIME EUR
	JIS	Yes	DATE (JIS)	DATE (JIS)	DATFMT (*JIS)	DATETIME JIS
Time Format (Note 7)	**ISO**	Yes	TIME (ISO) (Note 8)	TIME (**ISO**)	TIMFMT (*ISO) (Note 8)	DATETIME **ISO** (Note 9)
	USA	Yes	TIME (USA)	TIME (USA)	TIMFMT (*USA)	DATETIME USA
	EUR	Yes	TIME (EUR)	TIME (EUR)	TIMFMT (*EUR)	DATETIME EUR
	JIS	Yes	TIME (JIS)	TIME (JIS)	TIMFMT (*JIS)	DATETIME JIS
Package Isolation Level (Note 10)	Repeatable read	Yes	ISOLATION (**RR**)[b]	ISOLATION (**RR**)		ISOLATION RR
	Read Stability (All)	Yes		ISOLATION (RS)	COMMIT (*ALL)	ISOLATIONRS
	Cursor stability	Yes	ISOLATION (CS)[b]	ISOLATION (CS)	COMMIT (*CS)	ISOLATION **CS**
	Uncommitted Read (Change)	Yes		ISOLATION (UR)	COMMIT (***CHG**)	ISOLATION UR
	No commit	No (Note 11)			COMMIT (*NONE)	ISOLATION NC

Table 21. Bind Options Supported by the DataJoiner DRDA Application Server (continued)

Bind Option	Value	Supported?	DB2 for MVS Precompile Option (Note 1)	DB2/VM Preprocessing Option	OS/400 Precompile Option	DDCS Prep or Bind Option
Bind Creation Control	**No errors allowed**	Yes	SQLERROR (**NOPACKAGE**)[b]	**NOCHECK**	OPTION([...] *GEN) GENLVL(00-09, **10**, 11-20)	SQLERROR **NOPACKAGE**
	Check only	Yes		CHECK	OPTION([...] *NOGEN)	SQLERROR CHECK
	Errors allowed	No	SQLERROR (CONTINUE)[b]	ERROR	OPTION([...] *GEN) GENLVL(21-40)	SQLERROR CONTINUE
Bind Explain Option	**No SQL statements**	Yes	EXPLAIN (**NO**)[b]	EXPLAIN (**NO**)		EXPLAIN **NO**
	All explainable SQL statements	No	EXPLAIN (YES)[b]	EXPLAIN (YES)		EXPLAIN YES
Package Owner Identifier	<***Authorization ID***>	No	OWNER[b]	OWNER		OWNER
	Any other value	No				
RDB Release Option	**Release at commit**	No	RELEASE (**COMMIT**)[b]	RELEASE (**COMMIT**)		RELEASE **COMMIT**
	Release at conversation deallocation	No	RELEASE (DEALLOCATE)[b]	RELEASE (DEALLOCATE)		RELEASE DEALLOCATE
Default RDB Collection ID	<***Authorization ID***>	No	QUALIFIER[b]	QUALIFIER	DFTRDBCOL	QUALIFIER
	Any other value	No				
Title (Package Description)	Any value (ignored by DB2)	No		LABEL	TEXT	TEXT
Query Block Protocol Control	**Fixed row**	Yes	CURRENTDATA (**YES**)[b]	SBLOCK	ALWBLK (***READ**)	BLOCKING **UNAMBIG**
	Limited block	Yes	CURRENTDATA (NO)[b]	BLOCK	ALWBLK (*ALLREAD)	BLOCKING ALL
	Forced fixed row	Yes		**NOBLOCK**	ALWBLK (*NONE)	BLOCKING NO

Table 21. Bind Options Supported by the DataJoiner DRDA Application Server (continued)

Bind Option	Value	Supported?	DB2 for MVS Precompile Option (Note 1)	DB2/VM Preprocessing Option	OS/400 Precompile Option	DDCS Prep or Bind Option
Package Default Char. Subtype						
	Use system default	No				CHARSUB **DEFAULT**
If Default CCSID is SBCS	BIT	No		CHARSUB (BIT)		CHARSUB BIT
If Default CCSID is SBCS	SBCS	No		CHARSUB (SBCS)		CHARSUB SBCS
If Default CCSID is SBCS	MBCS	No		CHARSUB (MBCS)		CHARSUB MBCS
If Default CCSID is MBCS	BIT	No		CHARSUB (BIT)		CHARSUB BIT
If Default CCSID is MBCS	SBCS	No		CHARSUB (SBCS)		CHARSUB SBCS
If Default CCSID is MBCS	MBCS	No		CHARSUB (MBCS)		CHARSUB MBCS
	Any other value	No				

Package Default CCSID	**Value specified when DB2 database was created**	No		CCSIDSBCS()
				CCSIDGRAPHIC()
				CCSIDMIXED()
						CCSIDS
						CCSIDG
						CCSIDM

	Any other value	No				
Decimal Precision (Note 12)	31	No	DEC (31)			DEC 31
	Any other value	No	DEC (**15**)			DEC 15
Replaced Package Version Name	**Null**	Yes	REPLVER[b]			REPLVER
	Any other value	No				

Table 21. Bind Options Supported by the DataJoiner DRDA Application Server (continued)

Bind Option	Value	Supported?	DB2 for MVS Precompile Option (Note 1)	DB2/VM Preprocessing Option	OS/400 Precompile Option	DDCS Prep or Bind Option

Notes:

(*) Default values are in **bold**.

(1) Most are precompile options. Bind options are indicated by ^{b}.

(2) Defaults to what the target database supports. For DataJoiner the default is apostrophe.

(3) Default for non-COBOL applications.

(4) Default for COBOL applications.

(5) Defaults to what the target database supports. For DataJoiner the default is period.

(6) Depending on the installation, *SYSVAL is equivalent to *PERIOD or *COMMA.

(7) Date and time formats must be the same for the DataJoiner DRDA AS.

(8) Default is dependent on the installation.

(9) Format applies to both date and time. If not specified, it defaults based on the country code. This default is mapped to ISO in DRDA flow.

(10) Package isolation level has no default because an explicit value is always present in the DRDA datastream.

(11) The isolation level will be escalated to Uncommitted Read (Change).

(12) Defaults to what the target database supports. For DataJoiner the default is 31.

Restrictions

The database that is being connected to cannot be cataloged on the DataJoiner DRDA AS node as remote (you can obtain this information by using the LIST DATABASE DIRECTORY command). Any attempt to access a remote database results in SQLCODE -1334 being returned on the connect.

Refer to Table 20 on page 297 and Table 21 on page 298 for a list of DRDA options that are not supported by the DataJoiner DRDA Application Server.

The following required DRDA functions are not supported by this DRDA Application Server:

- Object Existence Optional (BNDEXSOPT)

  This is the default for the Bind Existence Checking (BNDCHKEXS) bind option. DataJoiner does not support this particular bind option value.
- Multiple binds in a logical unit of work (LUW)

  This corresponds to DRDA rules PB5 and PB6.

All input character data is converted by the DRDA Application Server to the database default CCSID (except when the data is tagged as BIT) before it gets to DataJoiner. As a result, you should be aware of the following significant restriction when using this DRDA Application Server:

- *Always use input character data that is tagged as BIT if you do not want it converted to the database default CCSID for the operation.*

  This restriction applies even if you are performing an operation (for example, UPDATE) against a character column that is defined as FOR BIT DATA. If you do not tag the input data as BIT, the results are unpredictable.

Configuration Considerations

You configure the DRDA Application Server using the *drda_heap_sz* (DRDA Heap Size) database manager configuration parameter, which specifies the amount of memory, in pages, that is allocated for use by the DRDA Application Server.

CCSID Support

The following list explains each column of Table 22, which follows.

- **AIX 3.2 Locale** and **AIX 4.1 Locale** list the locale values that are supported by the database manager. The locale specifies a `language_territory` combination. The value of `territory` is used to map to the country code.
- **Country Name** specifies the name of the country. Where required, the language for that locale is listed in parentheses ().
- **Country Code** shows the country code that is mapped from the territory portion of the locale, and which is used by the database manager internally for providing country-specific support.
- **Supported Code Set** shows the code set that is associated with the supported locale. The code set is mapped to the DB2 code page.
- **Database Code Page** shows the IBM-defined code page as mapped from the operating system code set.
- **EBCDIC AR CCSID** shows the CCSIDs that are supported for the EBCDIC AR applications.
- **ASCII AR CCSID** shows the CCSIDs that are supported for the ASCII AR applications.

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID
`ar_AA`	`ar_AA`	Arabic countries	785	ISO8859-6	1089[a]	n/a	864 1046 1089 1256
`Ar_AA`	`Ar_AA`	Arabic countries	785	IBM-1046	1046	420	864 1046 1089 1256

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server (continued)

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID
C	C	Australia	001	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
nl_BE	nl_BE	Belgium (Dutch)	032	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Nl_BE	Nl_BE	Belgium (Dutch)	032	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
fr_BE	fr_BE	Belgium (French)	032	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Fr_BE	Fr_BE	Belgium (French)	032	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
bg_BG	bg_BG	Bulgaria	359	ISO-8859-5	915	1025	855 866 915 1251 1283
en_US	en_US	Canada U.S.A.	001	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
En_US	En_US	Canada U.S.A.	001	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
fr_CA	fr_CA	Canada (French)	002	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 863 1051 1252 1275
Fr_CA	Fr_CA	Canada (French)	002	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 863 1051 1252 1275
hr_HR	hr_HR	Croatia	385	ISO8859-2	912	870	852 912 1250 1282

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server (continued)

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID
cs_CS	cs_CZ	Czech Republic	042	ISO8859-2	912	870	852 912 1250 1282
da_DK	da_DK	Denmark	045	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Da_DK	Da_DK	Denmark	045	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
fi_FI	fi_FI	Finland	358	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Fi_FI	Fi_FI	Finland	358	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
mk_MK	mk_MK	FYR Macedonia	389	ISO8859-5	915	1025	855 866 915 1251 1283
fr_FR	fr_FR	France	033	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Fr_FR	Fr_FR	France	033	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
de_DE	de_DE	Germany	049	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
De_DE	De_DE	Germany	049	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
el_GR	el_GR	Greece	030	ISO8859-7	813	423 875	813 869 1253 1280
hu_HU	hu_HU	Hungary	036	ISO8859-2	912	870	852 912 1250 1282

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server (continued)

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID
is_IS	is_IS	Iceland	354	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Is_IS	Is_IS	Iceland	354	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
iw	iw_IL	Israel	972	ISO8859-8	916	424	862 916 1255
it_IT	it_IT	Italy	039	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
It_IT	It_IT	Italy	039	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
ja_JP	ja_JP	Japan	081	IBM-eucJP	954	930 931 939 5026 5035	932 942 943 954
Ja_JP	Ja_JP	Japan	081	IBM-932	932	930 931 939 5026 5035	932 942 943 954
ko_KR	ko_KR	Korea	082	IBM-eucKR	970	933	949 970
C	C	Latin America	001	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
nl_NL	nl_NL	Netherlands (Dutch)	031	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Nl_NL	Nl_NL	Netherlands (Dutch)	031	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
no_NO	no_NO	Norway	047	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server (continued)

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID
No_NO	No_NO	Norway	047	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
zh_CN	zh_CN	PRC (China)	086	IBM-eucCN	1383	935	1381 1383
pl_PL	pl_PL	Poland	048	ISO8859-2	912	870	852 912 1250 1282
pt_PT	pt_PT	Portugal	351	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Pt_PT	Pt_PT	Portugal	351	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
ro_RO	ro_RO	Romania	040	ISO8859-2	912	870	852 912 1250 1282
ru_RU	ru_RU	Russia	007	ISO8859-5	915	1025	855 866 915 1251 1283
sp_YU	sr_SP	Serbia (Cyrillic)	381	ISO8859-5	915	1025	855 866 915 1251 1283
sh_YU	sh_SP	Serbia (Latin)	038	ISO8859-2	912	870	852 912 1250 1282
sk_SK	sk_SK	Slovakia	042	ISO8859-2	912	870	852 912 1250 1282
si_SI	si_SI	Slovenia	386	ISO8859-2	912	870	852 912 1250 1282
es_ES	es_ES	Spain	034	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Es_ES	Es_ES	Spain	034	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
sv_SV	sv_SE	Sweden	046	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server (continued)

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID
Sv_SV	Sv_SE	Sweden	046	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
fr_CH	fr_CH	Switzerland (French)	041	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Fr_CH	Fr_CH	Switzerland (French)	041	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
de_CH	de_CH	Switzerland (German)	041	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
De_CH	De_CH	Switzerland (German)	041	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
Zh_TW	Zh_TW	Taiwan	081	big5	950	937	938 948 950 964
zh_TW	zh_TW	Taiwan	081	IBM-eucTW	964	937	938 948 950 964
th_TH	th_TH	Thailand	066	TIS620.2533	874	838	874
tr_TR	tr_TR	Turkey	090	ISO8859-9	920	1026	857 920 1254 1281
en_GB	en_GB	U.K.	044	ISO-8859-1	819	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
En_GB	En_GB	U.K.	044	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275
C	C	U.S.A. (English)	001	IBM-850	850	037 273 277 278 280 284 285 297 500 871	437 819 850 1051 1252 1275

Table 22. CCSID Support for DataJoiner for AIX DRDA Application Server (continued)

AIX 3.2 Locale	AIX 4.1 Locale	Country Name	Country Code	Supported Code Set	Database Code Page	EBCDIC AR CCSID	ASCII AR CCSID

a: There is no EBCDIC AR CCSID support for code page 1089 because code page 1089 does not encode all the characters of the IBM-defined EBCDIC Arabic code page 420. Converting character data encoded on code page 420 to code page 1089 would result in some characters being lost.

Supported DRDA Application Requesters

The following sections describe the IBM ARs that are supported. For information about other ARs, contact your supplier.

MVS DRDA Application Requesters

The PTFs required for an MVS AR depend on the version of DB2 for MVS that you are using:

- DB2 for MVS Version 3.1 AR requires PTFs UN75959, UN80493, and UN80607 to be applied
- DB2 for MVS Version 4.1 AR requires PTFs UN80494 and UN80608 to be applied

DDCS DRDA Application Requesters

You can use DDCS Version 2.3 ARs with the DataJoiner DRDA AS. You do not have to apply any PTFs.

DB2 for VM DRDA Application Requesters

The PTFs required for a DB2 for VM (SQL/DS) AR depend on the version of DB2 for VM that you are using:

- DB2 for VM Version 3.3 AR requires PTF UN47865 to be applied
- DB2 for VM Version 3.4 AR does not require any PTFs to be applied

Additional steps are required to ensure that the following DB2 for VM utilities work properly against a DataJoiner database through the DRDA Application Server:

- SQLDBSU
  1. Ensure that the correct PTF is installed on your DB2 for VM system:
     - DB2 for VM V3R3 requires PTF UN84183 to be installed.
     - DB2 for VM V3R4 requires PTF UN84184 to be installed.
  2. Create dummy tables in your DataJoiner database by executing the `sqldbsu` utility that is provided with DataJoiner as follows:

     ```
     sqldbsu database_name
     ```

3. Bind SQLDBSU from DB2 for VM. See the description of using a DRDA environment in the *SQL/DS System Administration for IBM VM Systems* book for details. (You can skip the step on creating and populating the SQLDBA.DBSOPTIONS table because this is done by the `sqldbsu` utility in the previous step.)

- ISQL

  1. Complete the steps that are described above for SQLDBSU.
  2. Set up dummy tables in your DataJoiner database by executing the `isql` utility that is provided with DataJoiner as follows:

     ```
     isql database_name
     ```
  3. Bind ISQL from DB2 for VM. See the description of using a DRDA environment in the *SQL/DS System Administration for IBM VM Systems* book for details.

Note: The `sqldbsu` and `isql` utilities are in `$HOME/sqllib/misc` where `$HOME` is the home directory of the instance owner.

No special setup is required for RXSQL; see the *SQL/DS Procedures Language Interface Installation* book for details.

DB2 for AS/400 DRDA Application Requesters

For OS/400 DRDA ARs to work against the DataJoiner DRDA AS, PTFs need to be applied on OS/400 as follows:

Table 23. OS/400 PTFs required for DRDA AS

OS/400 PTR#	OS/400 V2R3 PTF	OS/400 V3R0.5 PTF	OS/400 V3R1 PTF
SA42461	SF23100	SF23950	SF23270
SA43024	SF23205	SF23950	SF23277
3989448	SF23101	SF23994	SF23271
3993315	SF23722	SF23988	SF23721
3994088	SF23987	SF23986	SF23985
3994297	SF23990	SF23989	SF23960

Appendix B. Configuration Parameters

DataJoiner has been designed with an extensive array of tuning parameters and configuration parameters. These parameters fall into two categories:

- Database manager parameters
- Database parameters

Descriptions for all the configuration parameters are available in the *DB2 Administration Guide*.

The default values and the range values for several of the configuration parameters are dependent on the operating system on which the database manager is installed. In addition, some database manager configuration parameters are applicable only in specific operating environments.

The following table lists the database manager configuration parameters and the default values and valid ranges for DataJoiner. Configuration parameters that are not used on this platform are indicated by the string: "Not applicable for DataJoiner for UNIX systems."

Note: All the units in this table are in KBs.

Table 24. Database manager configuration parameters, default values, and valid ranges for DataJoiner

Parameter	Description	Default [Range]
adsm_nodename	ADSM node name	Null [any valid string]
adsm_owner	ADSM owner	Null [any valid string]
adsm_password	ADSM password	Null [any valid string]
agent_stack_sz	Agent stack size	Not applicable for DataJoiner for UNIX systems
agentpri	Priority of Agents	-1 (system) [-1, 41 - 125]
aslheapsz	Application support layer heap size	15 [1 - 524288]
authentication	Authentication type	SERVER [CLIENT, SERVER, DCS]
backbufsz	Default Backup Buffer Size	1024 [16 - 524288]
cpuspeed	CPU Speed	-1 [-1, 1e-10 - 1]
dft_account_str	Default accounting string	Null [any valid string]
dft_client_comm	Default client communications protocol	Null [Null, TCPIP, APPC]

Table 24. Database manager configuration parameters, default values, and valid ranges for DataJoiner (continued)

Parameter	Description	Default [Range]
dft_mon_bufpool	Default value of snapshot monitor buffer pool switch	OFF [ON, OFF]
dft_mon_lock	Default value of snapshot monitor lock switch	OFF [ON, OFF]
dft_mon_sort	Default value of snapshot monitor sort switch	OFF [ON, OFF]
dft_mon_stmt	Default value of snapshot monitor statement switch	OFF [ON, OFF]
dft_mon_table	Default value of snapshot monitor table switch	OFF [ON, OFF]
dft_mon_uow	Default value of snapshot monitor unit-of-work switch	OFF [ON, OFF]
dftdbpath	Default database path	home directory of instance owner [any valid path]
diaglevel	Diagnostic error capture level	3 [0 - 4]
diagpath	Diagnostic data directory path	Null [any valid path]
dir_cache	Directory cache support	OFF [ON,OFF]
dir_obj_name	Object name in DCE namespace	Null [any valid string]
dir_path_name	Directory path name in DCE namespace	/.:/subsys/database [any valid path]
dir_type	Directory services type	NONE [NONE, DCE]
dos_rqrioblk	DOS requester I/O block size	Not applicable for DataJoiner for UNIX systems
drda_heap_sz	DRDA heap size	128 [16 - 60000]
fileserver	IPX/SPX fileserver name	Null [any valid string]
indexrec	Index re-creation time	restart [access, restart]
ipx_socket	IPX/SPX socket number	879E [879E - 87A2]
keepdari	Keep DARI process indicator	Yes [Yes, No]
max_idleagents	Maximum number of idle agents	3 [0 - maxagents]
maxagents	Maximum number of agents	200 [1 - 64000]
maxcagents	Maximum number of concurrent agents	-1 (maxagents) [-1, 1 - maxagents]
maxdari	Maximum number of DARI processes	-1 (maxagents) [-1, 0 - maxagents]

Table 24. Database manager configuration parameters, default values, and valid ranges for DataJoiner (continued)

Parameter	Description	Default [Range]
maxtotfilop	Maximum total files open per application	Not applicable for DataJoiner for UNIX systems
min_priv_mem	Minimum committed private memory	Not applicable for DataJoiner for UNIX systems
mon_heap_sz	Database system monitor heap size	48 [0 - 60000]
nname	NetBIOS node name	Not applicable for DataJoiner for UNIX systems
numbd	Maximum number of concurrently active databases	8 [1 - 60000]
objectname	IPX/SPX database manager object name	Null [any valid string]
priv_mem_thresh	Private memory threshold	Not applicable for DataJoiner for UNIX systems
query_heap_sz	Query heap size	1000 [2 - 524288]
restbufsz	Default restore buffer size	1024 [16 - 524288]
resync_interval	Transaction resynchronization interval	180 [60 - 60000]
route_obj_name	Routing information object name	Null [any valid string]
rqrioblk	Client I/O block size	32767 [4096 - 65535]
sheapthres	Sort heap threshold	4096 [250 - 524288]
sqlstmtsz	SQL statement size	256 [0 - 32767]
svcename	TCP/IP service name	Null [any valid string]
sysadm_group	System administration authority group name	Null [any valid group name]
sysctrl_group	System control authority group name	Primary group of instance owner
sysmaint_group	System maintenance authority group name	Null [any valid group name]
tm_database	Transaction manager database name	Null [any valid string]
tp_mon_name	Transaction processor monitor name (for all UNIX environments)	Null [CICS, ENCINA, blank, another string]

Table 24. Database manager configuration parameters, default values, and valid ranges for DataJoiner (continued)

Parameter	Description	Default [Range]
tpname	APPC transaction program name	Null [any valid string]
udf_mem_sz	UDF shared memory set size	256 [128 - 60000]

The following table lists the database configuration parameters whose defaults or ranges are influenced by the operating system on which the database manager was installed. The default values and valid ranges are given. Configuration parameters that are not used on this platform are indicated by the string: ″Not applicable for DataJoiner for UNIX systems″. This table does not include parameters that cannot be modified. (See the *DB2 Administration Guide* for a complete list of parameters, including those provided for information purposes.

Table 25. Database manager configuration parameters whose defaults or ranges are influenced by the operating system on which the database manager was installed

Parameter	Description	Default [Range]
applheapsz	Application Heap Size	128 [32 - 60000]
autorestart	Auto Restart Enable	ON [ON, OFF]
avg_appls	Average number of active applications	1 [1 - *maxappls*]
buffpage	Buffer pool size	1000 [2**maxappls* - 524288]
catalogcache_sz	Catalog Cache Size	64 [1 - *dbheap*]
chngpgs_thresh	Changes pages threshold	60 [5 - 80]
copyprotect	Copy Protection Enable	Not applicable for DataJoiner for UNIX systems
dbheap	Database Heap	1200 [32 - 60000]
dft_extent_sz	Default extent size of table spaces	32 [2 - 256]
dft_loadrec_ses	Default number of load recovery sessions	1 [1 - 30000]
dft_prefetch_sz	Default prefetch size	16 [0 - 32767]
dir_obj_name	Object name in DCE Namespace	Null [any valid string]
dlchktime	Time interval for checking deadlocks	10000 [1000 - 600000]
indexrec	Index recreation time	Use system setting [system; access; restart]

Table 25. Database manager configuration parameters whose defaults or ranges are influenced by the operating system on which the database manager was installed (continued)

Parameter	Description	Default [Range]
indexsort	Index sort flag	Yes [Yes; No]
locklist	Maximum storage for lock lists	100 [4 - 60000]
locktimeout	Lock timeout	-1 [-1;0 - 30000]
logbufsz	Log Buffer Size	8 [4 - 128]
logfilsiz	Size of log files	1000 [4 -16384]
logprimary	Number of primary log files	3 [2 - 128]
logretain	Log retain enable	OFF [ON,OFF]
logsecond	Number of secondary log files	2 [0 - 126]
maxappls	Maximum number of active applications	40 [1 - 1000]
maxfilop	Maximum number of database files open per application	64 [2 - 1950]
maxlocks	Maximum percent of lock list before escalation	10 [1 - 100]
mincommit	Number of commits to group	1 [1 - 25]
newlogpath	Change the database log path	Null [any valid path]
num_freqvalues	Number of frequent values retained	10 [0 - 32767]
num_iocleaners	Number of asynchronous page cleaners	1 [0 - 255]
num_ioservers	Number of I/O servers	3 [1 - 255]
num_quantiles	Number of quantiles for columns	20 [0 - 32767]
pckcachesz	Package cache size	36 [1 - *applheapsz*]
rec_his_retentn	Recovery history retention period	366 [-1; 0 - 30000]
seqdetect	Sequential detection flag	Yes [Yes; No]
softmax	Log records to write before soft checkpoint	100 [1 - 100]
sortheap	Sort heap size	256 [16 - 524288]
stat_heap_sz	Statistics heap size	4384 [1096 - 524288]
stmtheap	Statement heap size	2048 [128 - 60000]

Table 25. Database manager configuration parameters whose defaults or ranges are influenced by the operating system on which the database manager was installed (continued)

Parameter	Description	Default [Range]
userexit	User exit enable	OFF [ON, OFF]
util_heap_sz	Utility heap size	5000 [16 - 524288]

See the *DB2 Administration Guide* for a description of the database configuration parameters.

Appendix C. National Language Support

This appendix contains information about the National Language Support (NLS) provided by DataJoiner, including information about countries, languages, and code pages (code sets) supported and how to configure and use DataJoiner NLS features in both your applications and databases.

Notes:

1. DataJoiner for Solaris systems is available in English only.
2. The replication administration tool is not double-byte character set (DBCS)-enabled.

Setting Up Your Language Environment

All language specific messages, help panels, and application defaults are stored independently of the programs, and are translated in the following languages and encoded in the following codesets:

German in Germany	IBM-850	(De_DE)
German in Germany	ISO8859-1	(de_DE)
English in United States	IBM-850	(En_US)
Spanish in Spain	IBM-850	(Es_ES)
Spanish in Spain	ISO8859-1	(es_ES)
French in France	IBM-850	(Fr_FR)
French in France	ISO8859-1	(fr_FR)
Italian in Italy	IBM-850	(It_IT)
Italian in Italy	ISO8859-1	(it_IT)
Japanese in Japan	IBM-932	(Ja_JP)
Japanese in Japan	IBM-eucJP	(ja_JP)
Korean in Korea	IBM-eucKR	(ko_KR)
Portuguese in Brazil	ISO8859-1	(pt_BR)
Simplified-Chinese in PRC	IBM-eucCN	(zh_CN)

You specify the language environment you wish to use by installing the message option for the desired language, then setting the **LANG** environment variable to the desired locale. For example if you want to use Fr_FR messages, you must set **LANG** to Fr_FR and have the Fr_FR message option installed.

The selected message catalog options as well as the messages for the installation scripts are placed in the /usr/lpp/djx_02_01_01/msg/%L directories on the target workstation where %L is equal to the locale name of the message catalog. Selected DBA Utility messages and resources are placed in the following directories:

```
/usr/lpp/djx_02_01_01/msg/%L
/usr/lpp/djx_02_01_01/dba/res/%L
```

When the djx_02_01_01.client option is installed, a link is made for the /usr/lpp/djx_02_01_01/msg/En_US directory to /usr/lpp/djx_02_01_01/msg/prime.

Similarly, when the djx_02_01_01.dd option is installed, a link is made for the /usr/lpp/djx_02_01_01/dba/res/En_US directory to /usr/lpp/djx_02_01_01/dba/res/prime.

The prime or default directory is the directory that is searched for messages if the user's locale does not match any of the installed messages. For example, if the user's locale is set to No_NO (which does not have translated messages) the messages in prime (the default is En_US) will be used instead. The prime directory can be changed by *root* simply by changing the links. For example, if users are all working in IBM-850 French locales such as Fr_BE, Fr_CA, Fr_CH and Fr_FR, it might be preferable to install the French (Fr_FR) messages and make a link for the Fr_FR message directory to prime as follows:

```
ln -s -f /usr/lpp/djx_02_01_01/msg/Fr_FR \ /usr/lpp/djx_02_01_01/msg/prime
ln -s -f /usr/lpp/djx_02_01_01/dba/res/Fr_FR \ /usr/lpp/djx_02_01_01/dba/res/prime
```

Note that if prime is changed it will affect every instance for this version of the product.

If more than one ″default″ directory is needed, one of the following methods can be used:

- Create a link for an existing /usr/lpp/djx_02_01_01/msg/%L directory to /usr/lpp/djx_02_01_01/msg/NewLocale where NewLocale is the name of a new locale such as Fr_BE.

or

- Create a new directory /usr/lpp/djx_02_01_01/msg/newLocale and either copy the files from an existing /usr/lpp/djx_02_01_01/msg/%L directory or create links for the files in the existing /usr/lpp/djx_02_01_01/msg/%L directory to the /usr/lpp/djx_02_01_01/msg/newLocale directory.

For example, suppose the installer wants to support the En_US, No_NO, Po_PO, Fr_BE, Fr_CA, Fr_CH, Fr_FR, fr_BE, fr_CA, fr_FR, and fr_CH locales messages with the following default mappings:

- No_NO and Po_PO to En_US
- Fr_BE, Fr_CA, and Fr_CH to Fr_FR
- fr_BE, fr_CA, and fr_CH to fr_FR.

This can be accomplished as follows:

- Install the djx_02_01_01.client, the djx_02_01_01.msg.Fr_FR.client, and the djx_02_01_01.fr_FR.client options

  This will create the directories:

```
/usr/lpp/djx_02_01_01/msg/En_US
/usr/lpp/djx_02_01_01/msg/Fr_FR
/usr/lpp/djx_02_01_01/msg/fr_FR
```

and the link for:

```
/usr/lpp/djx_02_01_01/msg/En_US to /usr/lpp/djx_02_01_01/msg/prime
```

This link establishes the default prime as En_US, and thus No_NO and Po_PO will default to En_US.

- Create links for /usr/lpp/djx_02_01_01/msg/Fr_FR to:
  - /usr/lpp/djx_02_01_01/msg/Fr_BE
  - /usr/lpp/djx_02_01_01/msg/Fr_CA
  - /usr/lpp/djx_02_01_01/msg/Fr_CH
- Create links for /usr/lpp/djx_02_01_01/msg/fr_FR to:
  - /usr/lpp/djx_02_01_01/msg/fr_BE
  - /usr/lpp/djx_02_01_01/msg/fr_CA
  - /usr/lpp/djx_02_01_01/msg/fr_CH

Note: If you are in Australia or Latin America, you may want to set your locale to En_US (IBM-850) or en_US (ISO8859-1) before database manager installation to match code set requirements and the language of the messages to be selected at installation.

If you are in Latin America, you can also choose to set the locale to Es_ES (IBM-850) or es_ES (ISO8859-1) if you will be using Spanish as the language for the messages. It is also possible to use a Spanish locale and have English as the language for the messages.

Setting Up Your Application and Database NLS Environments

DataJoiner NLS support is based on specific code set and territory settings that are established when a database is created or when an application executes. The code set value is mapped to an IBM-defined code page, and the territory is mapped to an IBM-defined country code.

In DataJoiner terminology, the code set is a character string identifier for the code page, and the territory is a character string identifier for the country code. A code page is a mapping of each character from a character set, such as the Latin alphabet, to a numeric representation. Each code page is identified by a numeric identifier. For example, code set ″IBM-850,″ code page 850 represents the character ″A″ as hexadecimal 41. The country code setting establishes any national conventions that are supported by the database manager. Each country is also represented by a numeric identifier. For example, territory US, country code 001 represents the United States.

This information is used by the database managers as follows:

1. Database Creation

   During database creation, if the territory and codeset are not specified on the CREATE DATABASE command (or API), DataJoiner uses the territory and codeset of the application that invokes the command.

   Note: In AIX the territory value is the second half of a locale which is a combination of language and territory (*language_territory*), for example: En_US; specify only the territory value on the create database command.

   This defaulting approach adheres to portability guidelines for designing international applications and products that by their nature, are required to work in different countries. These guidelines call for dynamically providing the NLS functions that are required by the active environment.

   When you create databases, you need to either provide proper territory and codeset specifications or ensure that the operating environment is set to meet your country's application requirements.

   For internal processing, DataJoiner maps the territory value to a country code and the codeset to an IBM-defined code page value. (See Table 27 on page 331 for a mapping of territory and codeset to country code and code page.) In turn, the country code and code page are used to determine the default collating sequence that will be used for the database. This default collating sequence may be explicitly overridden, see "CREATE DATABASE" in the *DB2 Command Reference* for information on how to override the collating sequence.

   All this NLS information, including the collating sequence, is stored in the database configuration file. Once created this information cannot be changed since it is used to determine additional NLS processing. For example, the collating sequence is used to generate and process indexes that are required on character data type columns. It is also used by character string comparison functions, and by the SORT BY, and GROUP BY clauses in SQL statements.

2. Database Access

   When the database is accessed, the derived country code of the application determines that the date and time presentation (display and print) formats. DataJoiner transforms the date and time from the internal storage format to the appropriate country-external format. For Extended Services for

OS/2 and DB2 for OS/2 database clients that do not pass their country code to the database server, the database country code is used.

Also, while accessing the database, the application code page is compared to the code page of the database to determine if character data conversion is required. See “Conversion Between Different Code Pages” on page 326 for more information about character conversion.

Deriving Code Page Values

The **application code page** is derived from the active environment when the database connection is made. The **database code page** is derived from the value that is specified, either explicitly or by default, at the time the database is created. The following defines how the *active environment* is determined in different operating environments, for example:

UNIX In UNIX-based environments, the active environment is determined from the locale environment variables, which include information about language, territory, and code set.

OS/2 In OS/2, primary and secondary code pages are specified in the `CONFIG.SYS` file. You can use the `chcp` command to display and dynamically change code pages within a given session. See your OS/2 documentation for details.

DOS In DOS, the active code page is determined by the value that is specified in the `COUNTRY` command in the `CONFIG.SYS` file. You can use the `chcp` command to display and dynamically change code pages within a given session. See your DOS documentation for details.

Macintosh

For the Macintosh operating system, if the DB2CODEPAGE environment variable is set, its value is taken as the application code page. If DB2CODEPAGE is not set, the Macintosh code page is derived from the Regional version code from the installed script. Supported Macintosh code pages include:

1275 (Apple Latin 1)
1282 (Apple Latin 2)
1280 (Apple Greek)
1283 (Apple Cyrillic)
1281 (Apple Turkish)
932 (Japanese)

Windows

For Windows, if the DB2CODEPAGE environment keyword is set, its value is taken as the application code page. If DB2CODEPAGE is not set, the Windows code page is derived from the country ID, as specified in the `iCountry` value in the [intl] section of the Windows WIN.INI file. Supported Windows code pages include:

943 (Japanese)
    1252 (Windows Latin 1)
949 (Korean)
    1253 (Windows Greek)
950 (Traditional Chinese)
    1254 (Windows Turkish)
1004 (Latin 1)
    1255 (Windows Hebrew)
1250 (Windows Latin 2)
    1256 (Windows Arabic)
1251 (Windows Cyrillic)
    1381 (Simplified Chinese)

Windows 95

For Windows 95, if the DB2CODEPAGE environment keyword is set, its value is taken as the application code page. If DB2CODEPAGE is not set, the Windows 95 code page is derived from the ANSI code page setting in the Registry. Supported Windows 95 code pages include:

943 (Japanese)
    1252 (Windows Latin 1)
949 (Korean)
    1253 (Windows Greek)
950 (Traditional Chinese)
    1254 (Windows Turkish)
1004 (Latin 1)
    1255 (Windows Hebrew)
1250 (Windows Latin 2)
    1256 (Windows Arabic)
1251 (Windows Cyrillic)
    1381 (Simplified Chinese)

Windows NT

For Windows NT, if the DB2CODEPAGE environment keyword is set, its value is taken as the application code page. If DB2CODEPAGE is not set, the Windows NT code page is derived from the ANSI code page setting in the Registry. Supported Windows NT code pages include:

943 (Japanese)
    1252 (Windows Latin 1)
949 (Korean)
    1253 (Windows Greek)
950 (Traditional Chinese)
    1254 (Windows Turkish)

1004 (Latin 1)
1255 (Windows Hebrew)
1250 (Windows Latin 2)
1256 (Windows Arabic)
1251 (Windows Cyrillic)
1381 (Simplified Chinese)

See Table 27 on page 331 for a complete list of supported locales, code page equivalents.

Deriving Locales in Application Programs

Locales are specific to UNIX-based operating systems. There are two locales:

- The environment locale allows you to specify the language, currency symbol, and so on, that you want to use.
- The program locale contains the current language, currency symbol, and so on, of a program that is executing.

When your program is started, it gets a default C locale. It does **not** get a copy of the environment locale. Your program has a few choices:

- Ignore the environment locale. Your program could hardcode some options. For example, your program could set the language to *spanish* with the `setlocale()` function.
- Copy the environment locale to the programe locale.
- Ignore the environment locale. Use whatever defaults you get from the operating system.

How DataJoiner Derives Locales

With UNIX, the active locale that is used by DataJoiner is determined from the LC-CTYPE portion of the locale. For details, see the NLS documentation for your operating system.

- If LC_CTYPE of the program locale has a value other than that of ’C’, DataJoiner will use this value to determine the application code page by mapping it to its corresponding code page.
- If LC_CTYPE has the value of ’C’ (the ’C’ locale), DataJoiner will set the program locale according to the environment locale using the `setlocale()` function.
- If LC_CTYPE still has a value of ’C’, DataJoiner will use its default locale for that platform. See the *DB2 SDK Building Your Applications* book for your platform for information on the default locale for that platform.
- If LC_CTYPE’s value is no longer ’C’, its new value will be used to map to a corresponding code page.

Programming Considerations

It is strongly recommended that applications be precompiled, bound, compiled, and executed using the same code page. This is because data conversions by the server can occur in both the bind and the execution phases. Users should ensure that the same conversion tables are used by binding and executing with the same active code page. For a discussion of how applications determine the active code page, see “Deriving Code Page Values” on page 321.

Any external data that is obtained by the application will be assumed to be in the application code page. This includes data that is obtained from a file or from user input. Make sure that data sources outside the application uses the same code page as the application.

If you use host variables that use graphic data in your C or C++ applications, there are special precompiler, application performance, and application design issues you need to consider. For a detailed discussion of these considerations, or if you deal with EUC code sets in your applications, see the *DB2 Application Programming Guide*.

Coding SQL Statements

The coding of SQL statements is not language dependent. The SQL keywords must be typed as they are shown in this book, although they may be typed in uppercase, lowercase, or mixed case. The names of database objects, host variables, and program labels that occur in an SQL statement cannot contain characters outside the extended character set that is supported by your code page. For more information about extended character sets, see the *DB2 SQL Reference*.

Constant character strings in static SQL statements are converted at bind timed from the application code page to the database code page, and will be used at execution time in this database code page representation. To avoid such conversions if they are not desired, you can use host variables in place of string constants.

The server does not convert file names. To code a file name, either use the ASCII invariant set, or provide the path name in hexadecimal values that are physically stored in the file system.

Coding Remote Stored Procedures

When coding stored procedures that will be running remotely, the following considerations apply:

- Data in a stored procedure must be in the database code page.

- Data passed to or from a stored procedure using an SQLDA that has character data type must really contain character data. Numeric data and data structures must never be passed if the client application code page is different from the database code page. This is because the server will convert all character data in an SQLDA. To avoid character conversion, you can pass data by defining it in binary string format by using a data type of BLOB or by defining the character data as FOR BIT DATA.

Precompiling and Binding

At precompile/bind time, the precompiler is the executing application. The active code page when the database connection was made prior to the precompile request is used for precompiled statements and for any character data returned in the SQLCA.

Executing an Application

At execution time, the active code page of the user application when a database connection is made is in effect for the duration of the connection. All data is interpreted based on this code page; this includes dynamic SQL statements, user input data, user output data, and character fields in the SQLCA.

A Note of Caution

Failure to follow these guidelines may produce unpredictable results. These conditions cannot be detected by the database manager, so no error or warning message will result. For example, a C application contains the following SQL statements operating against a table T1 with one column that is defined as C1 CHAR(20):

```
(0)  EXEC SQL CONNECT TO GLOBALDB;
(1)  EXEC SQL INSERT INTO T1 VALUES ('a-constant');
     strcpy(sqlstmt, "SELECT C1 FROM T1 WHERE C1='a-constant');
(2)  EXEC SQL PREPARE S1 FROM :sqlstmt;

Where:
application code page at bind time = x
application code page at execution time = y
database code page = z
```

At bind time '*a-constant*' in statement (1) is converted from code page x to code page z. This conversion can be noted as (x to z).

At execution time, '*a-constant*' (x to z) is inserted into the table when statement (1) is executed. However, the WHERE clause of statement (2) will be executed with '*a-constant*' (y to z). If the codepoints in the constant are such that the two conversions (x to z and y to z) yield different results, the SELECT in statement (2) will fail to retrieve the data that is inserted by statement (1).

This appendix contains information about the National Language Support (NLS) provided by DataJoiner, including information about countries, languages, and code pages (code sets) supported and how to configure and use DataJoiner NLS features in both your applications and databases.

Conversion Between Different Code Pages

Ideally, for optimal performance, your applications should always use the same code page as your database. However, this is not always practical or possible. DataJoiner provides support for character conversion that allows your application and database to use different code pages. Characters from one code page must be mapped to the other code page in order to maintain meaning of the data.

When Does Character Conversion Occur?

Character conversion can occur in the following situations:

- When a client or application that accesses a database is running in a code page that is different from the code page of the database.

  This database conversion will occur on the database server machine for both conversions from the application code page to the database code page and from the database code page to the application code page.

  You can minimize or eliminate client/server character conversion in some situations. For example, you could create a DB2 for AIX database that uses code page 850 to match an OS/2 and DOS client application environment that predominately uses code page 850.

  Note: The DB2 for OS/2 Version 1.0 or Version 1.2 database server does not support character conversion between different code pages. Ensure that the code pages on server and client are compatible. See Table 26 on page 329 for the code page conversions that are supported.

- When a client or application importing a PC/IXF file runs in a code page that is different from the file being imported.

  This data conversion will occur on the database client machine before the client accesses the database server. Additional data conversion may take place if the application is running in a code page that is different from the code page of the database (as stated in the previous point).

  Data conversion, if any, also depends on how the import utility was called. For more information, see the *DB2 Administration Guide*.

- When DDCS for AIX is used to access data on a DRDA server. In this case character conversion occurs by the receiver of data, as defined by the DRDA rules. For example, data that is sent to DB2 for MVS is converted to the appropriate MVS coded character set identifier (CCSID) by DB2 for MVS. The data that is sent back to the DDCS for AIX machine from DB2 for MVS is converted by DDCS for AIX. For more information, see the *DDCS Installation and Configuration Guide* for your platform.

Character conversion will **not** occur for:

- File names. You should either use the ASCII invariant set for file names or provide the file name in the hexadecimal values that are physically stored in the file system.
- Data that is targeted for or comes from a column assigned the FOR BIT DATA attribute, or data used in an SQL operation whose result is FOR BIT or BLOB data. In these cases, the data is treated as a byte stream, and no conversion occurs. [11] See the *DB2 SQL Reference* for unequal code page rules for assigning, comparing, and combining strings.
- Access to a DB2 for OS/2 Version 1.0 or Version 1.2 database server.
- A DB2 product or platform that does not support, or that does not have support installed, for the desired combination of code pages. In this case, an SQLCODE -322 (SQLSTATE 57017) is returned when you try to run your application.

Character Substitutions During Conversions

When your application converts form one code page to another, it is possible that one or more characters are not represented in the target code page. If this occurs, DataJoiner inserts a *substitution* character into the target string in place of the character that has no representation. The replacement character is then considered a valid part of the string. In situations where a substitution occurs, the SQLWARN10 indicator in the SQLCA is set to ’W’.

Note: Any character conversions that result from using the WCHARTYPE CONVERT precompiler option will not flag a warning if any substitutions take place.

Supported Character Conversions

When data conversion occurs, conversion will take place from a **source code page** to a **target code page**.

[11] However, a literal that is inserted into a column that is defined as FOR BIT DATA could be converted if that literal was part of an SQL statement which was converted.

The source code page is determined from the source of the data; data from the application has a source code page equal to the application code page, and data from the database has a source code page equal to the database code page.

The determination of target code page is more involved. Where the data is to be placed, as well as rules for intermediate operations, is considered:

- If the data is moved directly from an application into a database, with no intervening operations, the target code page is the database code page.
- If the data is being imported into a database from a PC/IXF file, there are two character conversion steps:
  1. From the PC/IXF file code page (source code page) to the application code page (target code page)
  2. From the application code page (source code page) to the database code page (target code page).

  Exercise caution in situations where two conversion steps might occur. Make sure that you follow the supported character conversions that are listed in Table 26 to avoid a possible loss of character data.
- If the data is derived from operations performed on character data, where the source may be any of the application code page, the database code page, FOR BIT DATA, or for BLOB data, data conversion is based on a set of rules. Some or all of the data items may have to be converted to an intermediate result, before the final target code page can be determined. See the *DB2 SQL Reference* for a summary of these rules and for specific application with individual operators and predicates.

Table 26 on page 329 shows the code page conversions that are supported. Any code page can be converted to any other code page that is listed in the same row of the table. For example, code page 437 can be converted to 819, 850, 863, 1004, 1051, 1252, or 1257.

Note: Character string conversions between multi-byte code pages, for example DBCS and EUC, may result in either an increase or a decrease in the length of the string.

Table 26. Supported Code Page Conversions

Code Pages	Countries
437, 819, 850, 863, 1004, 1051, 1252[a], 1275	Austria, Australia, Belgium, Brazil, Canada, Denmark, Finland, France, Germany, Iceland, Italy, Latin America, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, UK, USA
813, 869, 1253[a], 1280	Greece
852, 912, 1250[a], 1282	Croatia, Czech Republic, Hungary, Poland, Romania, Serbia / Montenegro (Latin), Slovakia, Slovenia
855, 915, 1251[a], 1283	Bulgaria, FYR Macedonia, Serbia / Montenegro (Cyrillic)
857, 920, 1254[a], 1281	Turkey
862, 916, 1255[a]	Israel
864, 1046, 1089, 1256[a]	Arabic countries
866, 915, 1251[a], 1283	Russia
932, 942[b], 943, 954	Japan[e]
938, 948[c], 950, 964	Taiwan[e]
949, 970[d]	Korea[e]
1381, 1383[d]	People's Republic of China[e]

Note:

a Code pages 1250, 1251, 1252, 1253, 1254, 1255, and 1256 are treated as server code pages by the Windows NT database manager.

b Code page 932 and 942 are treated as equivalent by DataJoiner, so no character conversion is performed between them.

c Code page 938 and 948 are treated as equivalent by DataJoiner, so no character conversion is performed between them.

d Code pages 949 and 970, and code pages 1381 and 1383 are identical except that they support different numbers of User Defined Characters. Some User Defined Characters will not be converted based on the character conversions provided by the operating system APIs.

e Conversion support is optionally installable for these countries.

Character Conversion Expansion Factor

When your application successfully completes an attempt to connect to a DataJoiner database server, you should consider the following fields in the returned SQLCA:

- The second token in the SQLERRMC field (tokens are separated by X'FF') indicates the code page of the database. The ninth token in the SQLERRMC field indicates the code page of the application. Querying the application's code page and comparing it to the database's code page informs the application whether it has established a connection that will undergo character conversions.

- The first and second entries in the SQLERRD array. SQLERR(1) contains an integer value equal to the maximum expected expansion or contraction factor for the length of mixed character data (CHAR data types) when converted to the database code page from the application code page. SQLERRD(2) contains an integer value equal to the maximum expected expansion or contraction factor for the length of mixed character data (CHAR data types) when converted to the application code page from the database code page. A value of 0 or 1 indicates no expansion; a value greater than 1 indicates a possible expansion in length; a negative value indicates a possible contraction. Refer to the *DB2 SQL Reference* for details on using the CONNECT statement.

The considerations for graphic string data should not be a factor in unequal code page situations. Each string always has the same number of characters, regardless of whether the data is in the application or the database code page.

For more information on dealing with unequal code page situations, see the *DB2 Application Programming Guide*.

Locale, Code Page, and Host CCSID Support

Table 27 on page 331 shows the locales, code sets, code pages, and CCSIDs that are supported by DataJoiner. Each row of this table shows the supported conversions between the database code page and the application code page (code set). These are the code pages under which the database and application must be running for client/server connectivity to take place.

The following list explains each column of the table:

1. **Supported Locale** lists the locales that are supported by DataJoiner. The locale (territory) is mapped to the country code.
2. **Country Code** shows the country code that maps to the territory of the supported locale.
3. **Supported Code Set** shows the AIX code set that is associated with the supported locale. The code set is mapped to the Database Code Page.
4. **Database Code Page** shows the IBM-defined code page under which the database is created.
5. **AIX Application Code Page** shows the IBM-defined code page that an AIX application can use when connecting to a DataJoiner database.

6. **OS/2 Application Code Page** shows the IBM-defined code page that an OS/2 client application can use when connecting to a DataJoiner database.
7. **DOS Application Code Page** shows the IBM-defined code page that a DOS client application can use when connecting to a DataJoiner database.
8. **Windows Application Code Page** shows the IBM-defined code page that a Windows client application can use when connecting to a DataJoiner database.
9. **Host Supported CCSID** shows the IBM-defined Coded Character Set Identifier (CCSID) representing the encoding of data coming from the host that is supported by DataJoiner.

   Note: Check the manuals for your host database product to determine which CCSIDs are supported in your host environment.
10. **Country Name** shows the name of the country.

Table 27. Supported Locales, Code Sets, Code Pages, and CCSIDs

Supported locale	Country code	Supported code set	Database code page	AIX application code page	OS/2 application code page	DOS application code page	Windows application code page	Host supported CCSID	Country name
ar_AR	785	IS08859-6	1089	1089	864	864	-	Note 5	Arabic
Ar_AR	785	IBM-1046	1046	1046	864	864	-	420 (Note 5)	Arabic
C (Note 1)	001	IBM-850	850	819, 850	850, 437	850, 437	Note 2	037	Australia
C (Note 1)	001	IBM-850	850	819, 850	850, 437	850, 437	Note 2	284	Latin America
C (Note 1)	001	IBM-850	850	819, 850	850, 437	850, 437	Note 2	037	United States
cs_CS	042	ISO8859-2	912	912	852	852	-	870	Czech
da_DK	045	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	277	Denmark
Da_DK	045	IBM-850	850	819, 850	850, 437	850, 437	Note 2	277	Denmark
de_CH	041	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	500	Switzerland (German)
De_CH	041	IBM-850	850	819, 850	850, 437	850, 437	Note 2	500	Switzerland (German)
de_DE	049	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	273	Germany
De_DE	049	IBM-850	850	819, 850	850, 437	850, 437	Note 2	273	Germany
el_GR	030	ISO8859-7	813	813	-	-	-	875	Greece

Table 27. Supported Locales, Code Sets, Code Pages, and CCSIDs (continued)

Supported locale	Country code	Supported code set	Database code page	AIX application code page	OS/2 application code page	DOS application code page	Windows application code page	Host supported CCSID	Country name
en_GB	044	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	285	United Kingdom
En_GB	044	IBM-850	850	819, 850	850, 437	850, 437	Note 2	285	United Kingdom
en_US	001	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	037, 500 (Note 3)	United States, Canada
En_US	001	IBM-850	850	819, 850	850, 437	850, 437	Note 2	037, 500 (Note 3)	United States, Canada
es_ES	034	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	284	Spain
Es_ES	034	IBM-850	850	819, 850	850, 437	850, 437	Note 2	284	Spain
fi_FI	358	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	278	Finland
Fi_FI	358	IBM-850	850	819, 850	850, 437	850, 437	Note 2	278	Finland
fr_BE	032	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	500	Belgium (French)
Fr_BE	032	IBM-850	850	819, 850	850, 437	850, 437	Note 2	500	Belgium (French)
fr_CA	002	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	037, 500 (Note 3)	Canada (French)
Fr_CA	002	IBM-850	850	819, 850	850, 437	850, 437	Note 2	037, 500 (Note 3)	Canada (French)
fr_CH	041	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	500	Switzerland (French)
Fr_CH	041	IBM-850	850	819, 850	850, 437	850, 437	Note 2	500	Switzerland (French)
fr_FR	033	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	297	France
Fr_FR	033	IBM-850	850	819, 850	850, 437	850, 437	Note 2	297	France
is_IS	354	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	871	Iceland
hr_HR	038	ISO8859-2	912	912	852	852	-	870	Croatia
hu_HU	036	ISO8859-2	912	912	852	852	-	870	Hungary
Is_IS	354	IBM-850	850	819, 850	850, 437	850, 437	Note 2	871	Iceland
it_IT	039	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	280	Italy

Table 27. Supported Locales, Code Sets, Code Pages, and CCSIDs (continued)

Supported locale	Country code	Supported code set	Database code page	AIX application code page	OS/2 application code page	DOS application code page	Windows application code page	Host supported CCSID	Country name
It_IT	039	IBM-850	850	819, 850	850, 437	850, 437	Note 2	280	Italy
iw (Note 4)	972	ISO8859-8	916	916	862	862	-	424	Israel
Ja_JP	081	IBM-932	932	932	942, 932	932	932	930, 939, 5026, 5035	Japan (Note 6)
ko_KR	088	IBM_eucKR	970	970	934, 944	-	-	-	Korea
mk_MK	389	ISO8859-5	915	915	855	855	-	1025	FYR Macedonia
nl_BE	032	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	500	Belgium (Dutch)
Nl_BE	032	IBM-850	850	819, 850	850, 437	850, 437	Note 2	500	Belgium (Dutch)
nl_NL	031	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	037	Netherlands
Nl_NL	031	IBM-850	850	819, 850	850, 437	850, 437	Note 2	037	Netherlands
no_NO	047	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	277	Norway
No_NO	047	IBM-850	850	819, 850	850, 437	850, 437	Note 2	277	Norway
pl_PL	048	ISO8859-2	912	912	852	852	-	870	Poland
pt_PT	351	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	037	Portugal
Pt_PT	351	IBM-850	850	819, 850	850, 437	850, 437	Note 2	037	Portugal
ro_RO	038	ISO8859-2	912	912	852	852	-	870	Romania
ru_RU	007	ISO8859-5	915	915	866	866	-	1025	Russia
sh_YU	038	ISO8859-2	912	912	852	852	-	870	Serbia (Latin)
si_SI	038	ISO8859-2	912	912	852	852	-	870	Slovenia
sk_CS	042	ISO8859-2	912	912	852	852	-	870	Slovakia
sp_YU	381	ISO8859-5	915	915	855	855	-	1025	Serbia/Montenegro (Cyrillic)
sv_SV	046	ISO8859-1	819	819, 850	850, 437	850, 437	Note 2	278	Sweden
Sv_SV	046	IBM-850	850	819, 850	850, 437	850, 437	Note 2	278	Sweden
th_TH	066	TIS620.2533	874	874	-	-	-	Note 7	Thailand
tr_TR	090	ISO8859-9	920	920	857	857	-	1026	Turkey
zh_CN	086	IBM_eucCN	1383	936, 946	-	-	-	-	PRC

Table 27. Supported Locales, Code Sets, Code Pages, and CCSIDs (continued)

Supported locale	Country code	Supported code set	Database code page	AIX application code page	OS/2 application code page	DOS application code page	Windows application code page	Host supported CCSID	Country name
zh_TW	088	big5	950	950	938, 948	-	-	-	Big-5

Table Notes:

1. DataJoiner uses the AIX environment locale name to determine the territory. The AIX environment defaults to the C (POSIX**) locale when one is not specified, the territory defaults to USA, and the code set defaults to IBM-850. These defaults are also applicable to countries that do not have locales defined yet, such as Australia and the Latin American countries. These defaults will be used until these countries have their own locales. The C locale is 7-bit ASCII-based, and therefore is a proper subset of locales that are based on IBM-850 and ISO8859-1. The national language support that is provided by DataJoiner for the United States is based on the IBM-850 and ISO8859-1, which are the same code sets that are used to support Australia and the Latin American countries. As a result, the default value is acceptable for Australia and the Latin American countries. Other countries must use their own locales to obtain proper national language support.

   If you are in Australia or Latin America and are using the default locale, consider setting your locale to En_US (IBM-850) or en_US (ISO8859-1) before installing DataJoiner. This ensures that your code set requirements are matched with the language of the messages that you selected during installation. If you are in Latin America, you could also choose to set your locale to Es_ES (IBM-850) or es_ES (IS8859-1) if you will be using Spanish as the language for the messages. It is also possible to use a Spanish locale and have English as the language for the messages. The default date format that is provided by DataJoiner will be the one for the United States or Spain, depending on the locale that is being used.
2. The Latin-1 Windows code page is 1004, which is a superset of code page 819 (ISO8859-1). DataJoiner performs round-trip conversion of character data that is encoded on code page 1004. However, for purposes of other NLS functions such as composing SQL identifiers, DataJoiner treats code page 1004 as equivalent to code page 819.
3. Canada uses CCSID 037 for DB2 for VSE or VM and DB2 for MVS, and CCSID 500 for OS/400 databases.
4. Neither AIX nor X/Open has registered territories to use with the Hebrew language (as identified by locale iw). Future additions of territory for this locale name will not impact DataJoiner NLS function.
5. DataJoiner does not support code page 1089 because code page 1089 does not encode all the characters of the IBM-defined EBCDIC Arabic code page

420. Converting character data that is encoded on code page 420 to code page 1089 would result in some characters being lost. DataJoiner supports code page 1046 since code page 1046 encodes all the characters of the EBCDIC Arabic code page 420.

DataJoiner does not support client/server connectivity between code page 1089 and code page 1046. As a result, Arabic language AIX database clients must use the same code page as the database that is defined on the server. Arabic language OS/2 and DOS database clients running on code page 864 can connect to a DataJoiner server that is running in either 1089 or 1046.

6. Although the SBCS code page 850 is available in Japan, DataJoiner does not support this code page for a DBCS environment.
7. IBM Thailand provides its own sorting and collating routines to work with DataJoiner. Thai users should contact their local country support for additional information.

DBCS Code Page Support

Combined SBCS/DBCS code pages contain both single- and double-byte code points, and thus imply both an SBCS and DBCS code page. Table 28 on page 336 summarizes the relationships between the supported code pages (which is a combined SBCS/DBCS code page), and the SBCS and DBCS code pages implied by this combined code page.

Table 28. Relationships between SBCS and DBCS Code Pages

Supported Code Page	Platform	Implied SBCS Code Page	Implied DBCS Code Page	Name
932	OS/2, AIX	897	301	Japanese
942	OS/2	1041	301	Japanese Extended
943	Windows, Windows NT, Windows 95	1041	941	Japanese
938	OS/2	904	927	Traditional Chinese
948	OS/2	1043	927	Trad. Chinese Extended
949	OS/2	1088	951	Korean
950	OS/2 AIX HP	1114	947	Trad. Chinese big5
1381	OS/2	1115	1380	Simplified Chinese

Notes:

- Code pages 942 and 948 are only supported for database clients running under OS/2.
- Code pages 932 and 942 are treated as identical, as are code pages 938 and 948.
- Platform indicates the operating system where the code page is supported.

EUC Code Page Support

Extended UNIX Code (EUC) code pages are actually a combination of up to four character groups, known as G0, G1, G2, and G3. Each group can be identified with a code page in addition to the combined EUC code page identifier. The table that follows summarizes the relationships between the supported code pages (which are EUC code pages) and the G0 through G3 code pages that are implied by the EUC code page.

Table 29. Relationships between EUC Code Pages and Character Groups

Supported Code Page	Platform	Implied Code Pages				Name
		G0	G1	G2	G3	
954	AIX, HP-UX, Solaris	895	952	4992	953	Japanese EUC
964	AIX, HP-UX, Solaris	367	960	961	n/a	Traditional Chinese EUC
970	AIX, HP-UX, Solaris	367	971	n/a	n/a	Korean EUC

Table 29. Relationships between EUC Code Pages and Character Groups (continued)

Supported Code Page	Platform	Implied Code Pages				Name
		G0	G1	G2	G3	
1383	AIX, Solaris	367	1382	n/a	n/a	Simplified Chinese EUC

Database and client application support for graphic (pure double-byte character) data while running under EUC code pages with character encodings that are greater than two bytes in length is limited. DataJoiner has strict rules for graphics data that require all data to be exactly two bytes-wide. These restrictions essentially rule out many characters from both the Japanese and Traditional Chinese EUC code pages.

To overcome this restriction, support is provided at both the application level and the database level to represent Japanese and Traditional Chinese EUC graphic data that uses another coding scheme. A database that is created under Japanese and Traditional Chinese EUC code pages will store and manipulate graphic data by using the UCS-2 code set. Similarly, applications that run under those code pages will send graphic data to the database server as UCS-2-encoded data.

This support enables applications that run under EUC code pages to access the same type of data as those running under DBCS code pages.

For information regarding application development in EUC environments, refer to the *DB2 Application Programming Guide* and the *DB2 SQL Reference*.

The IBM-defined code page identifier associated with UCS-2-encoded data for DataJoiner 13844. The support for UCS-2-encoded data is at Level 1 of the standard (no combining characters).

DataJoiner Code Page Support

Because DataJoiner deals with multiple data sources, unequal code pages can exist in a heterogeneous database environment. In this situation, DataJoiner identifies the DataJoiner code page to the data source client code. The data source character set transformation logic returns the data to DataJoiner in the DataJoiner code page. If you use an invalid configuration (for example, if you attempt to transfer data from Sybase running the Sybase code page equivalent to 850 to DataJoiner in code page 8859), then the client code must either detect this error or map the data by using their own semantics. DataJoiner relies on non-IBM data sources for translation at the data source.

Specifying Code Pages

DataJoiner supports single-byte (SBCS) and double-byte (DBCS) character sets. These include the IBM PC code sets and a subset of the ISO code sets that are identified in Table 30.

Table 30. Supported Code Sets

Code Set	Code Page	SBCS or DBCS
IBM-850	850	SBCS
IBM-932	932	Combined SBCS 897/DBCS 301
IBM-1046	1046	SBCS
ISO8859-1	819	SBCS
ISO8859-2	912	SBCS
ISO8859-6	1089	SBCS
ISO8859-7	813	SBCS
ISO8859-8	916	SBCS
ISO8859-9	920	SBCS

Table 31 provides mappings between code pages and Oracle and Sybase options. Your data sources must be configured to correspond to these mappings, or else the client code must be able to detect the mismatch and flag it as an error, or map the data by using its own semantics. For example, Sybase maps all unmappable characters to '?'. See your data source documentation for information about other non-IBM data sources.

Table 31. Code Page Mappings

Code Page	Oracle	Sybase
850	NLS_LANG=American_America.US7ASCII	cp850
932	NLS_LANG=Japanese_Japan.JA16SJIS	sjis
1046	NLS_LANG=Arabic_UnitedArabEmirates.US7ASCII	n/a
819	NLS_LANG=German_Germany.WE8ISO8859P1	iso_1
912	NLS_LANG=German_Germany.EE8ISO8859P2	iso_2
1089	NLS_LANG=Arabic_UnitedArabEmirates.AR8ISO8859P6	iso_6
813	NLS_LANG=Greek_Greece.EL8ISO8859P7	iso_7
916	NLS_LANG=American_America.IW8ISO8859P8	iso_8
920	NLS_LANG=Turkish_Turkey.TR8ISO8859P9	iso_9
950	NLS_LANG=Chinese_Taiwan.ZHT16BIG5	
970	NLS_LANG=Korean_Korea.KO16KSC5601	

Table 31. Code Page Mappings (continued)

Code Page	Oracle	Sybase
1383	NLS_LANG=Chinese_China.ZHS16CGB231280	

For data sources that use the generic ODBC API, DataJoiner administrators must enter the code set value for each data source that is different from the default value in the SYSCAT.SERVERS catalog view. For more information about SYSCAT.SERVERS, see Appendix B, “System Catalog Views”, in *DataJoiner Application Programming and SQL Reference Supplement*.

For character data conversion to take place between a DataJoiner and a DRDA data source, conversion tables from the source code page to the target code page must exist at DataJoiner and at the DRDA data source. DataJoiner relies on non-DRDA data sources to provide all code page translations. Data sources using the same code set as DataJoiner require no translation.

If no conversion table can be found from the source code page to the target code page, DataJoiner issues an error message.

Administrators must enter the format of date, time, and timestamp variables for each data source in the SYSCAT.SERVERS catalog view. DataJoiner performs the necessary conversion to support the datetime values across multiple databases.

Character Sets

DataJoiner does not, in general, restrict the character set available to an application except as noted below.

DBCS Character Sets

Each combined SBCS/DBCS code page allows for both single- and double-byte character code points. This is accomplished by reserving a subset of the 256 available code points of each implied SBCS code page identifier for single-byte characters, with the remainder of the code points either undefined or allocated to the first byte of double-byte code points. These code points are shown in the following table.

Table 32. Mixed Character Set Code Points

Supported Mixed Code Page	Code Points for Single-byte Characters	Code Points for First Byte of Double-Byte Characters
932	x00-7F, xA1-DF	x81-9F, xE0-FC
942, 943	x00-80, xA0-DF, xFD-FF	x81-9F, xE0-FC
938	x00-7E	x81-FC
948	x00-80	x81-FC
949	x00-7F	x8F-FE
950	x00-7E	x81-FE
1381	x00-7F	x8C-FE

Code points that are not assigned to either category above are not defined, and are processed as single-byte undefined code points.

Within each implied DBCS code page, there are 256 code points available as the second byte for each valid first byte. These code points are also partitioned into valid and invalid second byte ranges for the purpose of determining whether a DBCS character is properly formed. Note that in DBCS environments, DataJoiner does not perform validity checking on individual double-byte characters.

Extended UNIX Code (EUC) Character Sets

Each EUC code page allows for both single-byte and up to three different sets of multi-byte character code points. This is accomplished by reserving a subset of the 256 available code points of each implied SBCS code page identifier for single-byte characters, with the remainder of the code points either undefined, allocated as an element of a multi-byte character, or allocated as a single-shift introducer of a multi-byte character. These code points are shown in the following tables.

Table 33. Japanese EUC Code Points

Group	1st Byte	2nd Byte	3rd Byte	4th Byte
G0	x20-x7E	n/a	n/a	n/a
G1	xA1-xFE	xA1-xFE	n/a	n/a
G2	x8E	xA1-xFE	n/a	n/a
G3	x8F	xA1-xFE	xA1-xFE	n/a

Table 34. Traditional Chinese EUC Code Points

Group	1st Byte	2nd Byte	3rd Byte	4th Byte
G0	x20-x7E	n / a	n / a	n / a
G1	xA1-xFE	xA1-xFE	n / a	n / a
G2	x8E	xA1-xFE	xA1-xFE	xA1-xFE
G3	n / a	n / a	n / a	n / a

Table 35. Korean EUC Code Points

Group	1st Byte	2nd Byte	3rd Byte	4th Byte
G0	x20-x7E	n / a	n / a	n / a
G1	xA1-xFE	xA1-xFE	n / a	n / a
G2	n / a	n / a	n / a	n / a
G3	n / a	n / a	n / a	n / a

Table 36. Simplified Chinese EUC Code Points

Group	1st Byte	2nd Byte	3rd Byte	4th Byte
G0	x20-x7E	n / a	n / a	n / a
G1	xA1-xFE	xA1-xFE	n / a	n / a
G2	n / a	n / a	n / a	n / a
G3	n / a	n / a	n / a	n / a

Code points that are not assigned to either category above are not defined, and are treated as single-byte undefined code points.

UCS-2 Character Sets

The UCS-2 encoding scheme is based on a straight assignment of a unique two-byte code point for each character. There are no modes or escape characters to specify modified characters or special cases. There are no protected ranges or shift characters, as is the case with DBCS or EUC encoding schemes. All code points are considered significant.

Supported Code Page	1st Byte	2nd Byte
13488[1]	x00-xFF	x00-xFF
Note: (1) Support for UCS-2 encoded data is provided only as an internal graphic data representation for applications and databases running under Japanese or Traditional-Chinese EUC code pages.		

Character Set for Identifiers

The basic character set that may be used in database names consists of the single-byte uppercase and lowercase Latin letters (A...Z, a...z), the Arabic numerals (0...9) and the underscore character (_). This list of letters is augmented with the three special characters #, @, and $ to provide compatibility with host database products. However, these special characters should be used with care in an NLS environment because they are not included in the NLS host (EBCDIC) invariant character set.

When naming database objects (such as tables and views), program labels, host variables, cursors, and statements alphabetics from the extended character set may also be used. For example, those letters with diacritical marks. The available characters depend on the code page in use and if you are using the database in a multiple code page environment, you must ensure that all code pages support any alphabetics you plan on using from the extended character set. See the *DB2 SQL Reference* for a discussion of delimited identifiers which can be used in SQL statements and can contain characters outside the extended character set.

Extended Character Set Definition for DBCS Identifiers

In DBCS environments, the extended character set consists of all the characters in the basic character set, plus those that are identified as a letter or digit as follows:

- All double-byte characters in each DBCS code page, except the double-byte space, are valid letters.
- The double-byte space is a special character.
- The single-byte characters available in each mixed code page are assigned to various categories as follows:

Category	**Valid Code Points within each Mixed Code Page**
Digits	x30-39
Letters	x23-24, x40-5A, x61-7A, xA6-DF (A6-DF for code pages 932 and 942 only)
Special Characters	All other valid single-byte character code points

Coding of SQL Statements

The coding of SQL statements is not language dependent. SQL is a programming language, and like other programming languages such as C, SQL is language invariant. The SQL keywords must be typed as shown in this book, although they may be typed in uppercase, lowercase, or mixed case. The names of database objects, host variables, and program labels that occur in an SQL statement cannot contain characters outside the DataJoiner extended character set as described above.

Collating Sequences

When the collating (or sort) sequence is not specified, DataJoiner selects a sequence that is based on the code page and country code of the database.

The DataJoiner collating sequence follows the order that is indicated by the “Character ID” column in the tables that are documented in the *National Language Design Guide: Volume 2, National Language Support Reference Manual,* SE09-8002. This guide is subsequently referred to as the *NLS Reference Manual*.

Other weight considerations stated in the *NLS Reference Manual* are not used by DataJoiner, which duplicates the order that is given by the “Character ID” column using internal tables.

Notes:

1. In order to provide consistent sort processing in a client/server and distributed database environment, DataJoiner uses a common sort sequence for all Latin-1 countries. As a result, the sort sequence for Spain and Latin American Spanish-speaking countries does not provide special sort considerations for letters “ch” and “ll”.
2. Because code points are likely to be assigned to different characters in a country’s EUC, PC DBCS, and UCS-2 code sets, you may obtain different results when you sort the same characters. For more information concerning collation within the database, please refer to the *DB2 SQL Reference Manual*.

For more information about the default collating sequences, see the *NLS Reference Manual*.

Datetime Values

The datetime data types are described below. Although datetime values can be used in certain arithmetic and string operations and are compatible with certain strings, they are neither strings nor numbers.

Date A *date* is a three-part value (year, month, and day). The range of the year part is 0001 to 9999. The range of the month part is 1 to 12. The range of the day part is 1 to *x*, where *x* depends on the month.

The internal representation of a date is a string of 4 bytes. Each byte consists of 2 packed decimal digits. The first 2 bytes represent the year, the third byte the month, and the last byte the day.

The length of a DATE column, as described in the SQLDA, is 10 bytes, which is the appropriate length for a character string representation of the value.

Time A *time* is a three-part value (hour, minute, and second) that designates a time of day under a 24-hour clock. The range of the hour part is 0 to 24; while the range of the other parts is 0 to 59. If the hour is 24, the minute and second specifications will be zero.

The internal representation of a time is a string of 3 bytes. Each byte is 2 packed decimal digits. The first byte represents the hour, the second byte the minute, and the last byte the second.

The length of a TIME column, as described in the SQLDA, is 8 bytes, which is the appropriate length for a character string representation of the value.

Timestamp

A *timestamp* is a seven-part value (year, month, day, hour, minute, second, and microsecond) that designates a datetime as defined above, except that the time includes a fractional specification of microseconds.

The internal representation of a timestamp is a string of 10 bytes, each of which consists of 2 packed decimal digits. The first 4 bytes represent the date, the next 3 bytes the time, and the last 3 bytes the microseconds.

The length of a TIMESTAMP column, as described in the SQLDA, is 26 bytes, which is the appropriate length for the character string representation of the value.

String Representations of Datetime Values

Values whose data types are DATE, TIME, or TIMESTAMP are represented in an internal form that is transparent to the SQL user. Dates, times, and timestamps can, however, also be represented by character strings, and these representations directly concern the SQL user since there are no constants or variables whose data types are DATE, TIME, or TIMESTAMP. Thus, to be retrieved, a datetime value must be assigned to a character string variable. The character string representation is normally the default format of datetime values that are associated with the country code of the database, unless overridden by specification of the *F* format option when the program is precompiled or bound to the database. See Table 39 on page 347 for a listing of the string formats for the various country codes.

When a valid string representation of a datetime value is used in an operation with an internal datetime value, the string representation is converted to the internal form of the date, time, or timestamp before the operation is performed. The following sections define the valid string representations of datetime values.

Date Strings

A string representation of a date is a character string that starts with a digit and has a length of at least 8 characters. Trailing blanks may be included; leading zeros may be omitted from the month and day portions.

Valid string formats for dates are listed in Table 1. Each format is identified by name and includes an associated abbreviation and an example of its use.

Table 37. Formats for String Representations of Dates

Format Name	**Abbreviation**	**Date Format**	**Example**
International Standards Organization	ISO	yyyy-mm-dd	1991-10-27
IBM USA standard	USA	mm/dd/yyyy	10/27/1991
IBM European standard	EUR	dd.mm.yyyy	27.10.1991
Japanese Industrial Standard Christian era	JIS	yyyy-mm-dd	1991-10-27
Site-defined (Local)	LOC	Depends on database country code	—

Time Strings

A string representation of a time is a character string that starts with a digit and has a length of at least 4 characters. Trailing blanks may be included; a leading zero may be omitted from the hour part of the time and seconds may be omitted entirely. If you choose to omit seconds, an implicit specification of 0 seconds is assumed. Thus, 13.30 is equivalent to 13.30.00.

Valid string formats for times are listed in Table 38. Each format is identified by name and includes an associated abbreviation and an example of its use.

Table 38. Formats for String Representations of Times

Format Name	**Abbreviation**	**Time Format**	**Example**
International Standards Organization	ISO	hh.mm.ss	13.30.05
IBM USA standard	USA	hh:mm AM or PM	1:30 PM
IBM European standard	EUR	hh.mm.ss	13.30.05

Table 38. Formats for String Representations of Times (continued)

Format Name	Abbreviation	Time Format	Example
Japanese Industrial Standard Christian Era	JIS	hh:mm:ss	13:30:05
Site-defined (Local)	LOC	Depends on database country code	—

Notes:

1. In ISO, EUR and JIS format, .ss (or :ss) is optional.
2. In the case of the USA time string format, the minutes specification may be omitted, indicating an implicit specification of 00 minutes. Thus 1 PM is equivalent to 1:00 PM.
3. In the USA time format, the hour must not be greater than 12 and cannot be 0 except for the special case of 00:00 AM. Using the ISO format of the 24-hour clock, the correspondence between the USA format and the 24-hour clock is as follows:

   12:01 AM through 12:59 AM corresponds to 00.01.00 through 00.59.00.

   01:00 AM through 11:59 AM corresponds to 01.00.00 through 11.59.00.

   12:00 PM (noon) through 11:59 PM corresponds to 12.00.00 through 23.59.00.

   12:00 AM (midnight) corresponds to 24.00.00, and 00:00 AM (midnight) corresponds to 00.00.00.

Timestamp Strings

A string representation of a timestamp is a character string that starts with a digit and has a length of at least 16 characters. The complete string representation of a timestamp has the form *yyyy-mm-dd-hh.mm.ss.nnnnnn*. Trailing blanks may be included. Leading zeros may be omitted from the month, day, and hour part of the timestamp, and microseconds may be truncated or entirely omitted. If you choose to omit any digit of the microseconds portion, an implicit specification of 0 is assumed. Thus, 1991-3-2-8.30.00 is equivalent to 1991-03-02-08.30.00.000000.

MBCS Considerations

Date and timestamp strings must contain only single-byte characters and digits.

Date and Time Formats

The character string representation of date and time formats is the default format of datetime values that are associated with the country code of the application. This default format may be overridden by specification of the *F* format option when the program is precompiled or bound to the database.

The following is a description of the input and output formats for date and time:

- Input Time Format
  - There is no default input time format
  - All time formats are allowed as input for all country codes.
- Output Time Format
  - The default output time format is equal to the local time format.
- Input Date Format
  - There is no default input date format
  - Where the local format for date conflicts with an ISO, JIS, EUR, or USA date format, the local format is recognized for date input. For example, see the UK entry in Table 39.
- Output Date Format
  - The default output date format is equal to the local date format.

Table 39 shows a listing of the string formats for the various country codes.

Table 39. Date and Time Formats by Country Code

Country Code	Local Date Format	Local Time Format	Default Output Date Format	Input Date Formats
785 Arabic	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
001 Australia (1)	mm-dd-yyyy	JIS	LOC	LOC, USA, EUR, ISO
061 Australia	dd-mm-yyyy	JIS	LOC	LOC, USA, EUR, ISO
032 Belgium	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
055 Brazil	dd.mm.yy	JIS	LOC	LOC, USA, EUR, ISO
359 Bulgaria	dd.mm.yyyy	JIS	EUR	LOC, USA, EUR, ISO
001 Canada	mm-dd-yyyy	JIS	USA	LOC, USA, EUR, ISO
002 Canada (French)	dd-mm-yyyy	ISO	ISO	LOC, USA, EUR, ISO
385 Croatia	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
042 Czech Republic	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
045 Denmark	dd-mm-yyyy	ISO	ISO	LOC, USA, EUR, ISO

Table 39. Date and Time Formats by Country Code (continued)

Country Code	Local Date Format	Local Time Format	Default Output Date Format	Input Date Formats
358 Finland	dd/mm/yyyy	ISO	EUR	LOC, EUR, ISO
389 FYR Macedonia	dd.mm.yyyy	JIS	EUR	LOC, USA, EUR, ISO
033 France	dd/mm/yyyy	JIS	EUR	LOC, EUR, ISO
049 Germany	dd/mm/yyyy	ISO	ISO	LOC, EUR, ISO
030 Greece	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
036 Hungary	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
354 Iceland	dd-mm-yyyy	JIS	LOC	LOC, USA, EUR, ISO
972 Israel	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
039 Italy	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
081 Japan	mm/dd/yyyy	JIS	ISO	LOC, USA, EUR, ISO
082 Korea	mm/dd/yyyy	JIS	ISO	LOC, USA, EUR, ISO
001 Latin America (1)	mm-dd-yyyy	JIS	LOC	LOC, USA, EUR, ISO
003 Latin America	dd-mm-yyyy	JIS	LOC	LOC, EUR, ISO
031 Netherlands	dd-mm-yyyy	JIS	LOC	LOC, USA, EUR, ISO
047 Norway	dd/mm/yyyy	ISO	EUR	LOC, EUR, ISO
048 Poland	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
351 Portugal	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
086 PRC	mm/dd/yyyy	JIS	ISO	LOC, USA, EUR, ISO
040 Romania	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
007 Russia	dd/mm/yyyy	ISO	LOC	LOC, EUR, ISO
381 Serbia/Montenegro	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
042 Slovakia	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
386 Slovenia	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
034 Spain	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
046 Sweden	dd/mm/yyyy	ISO	ISO	LOC, EUR, ISO
041 Switzerland	dd/mm/yyyy	ISO	EUR	LOC, EUR, ISO
088 Taiwan	mm-dd-yyyy	JIS	ISO	LOC, USA, EUR, ISO
066 Thailand (2)	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
090 Turkey	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
044 UK	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO

Table 39. Date and Time Formats by Country Code (continued)

Country Code	Local Date Format	Local Time Format	Default Output Date Format	Input Date Formats
001 USA	mm-dd-yyyy	JIS	USA	LOC, USA, EUR, ISO
Note: (1) Countries using the default C locale are assigned country code 001. (2) yyyy is in Buddhist era: Gregorian + 543 years.				

Appendix D. Resolving Problems Encountered by Applications That Predate Version 2.1.1

This appendix explains how to resolve problems that arise when certain applications, such as those based on DataJoiner Version 1.2, try to perform operations that are no longer valid in Version 2.1.1, to query or modify catalog tables that were updated for Versions 2.1 and 2.1.1, or to query catalog views that were updated for Version 2.1.1.

The word *applications* here refers to a wide range of programs and instructions; for example:

- Application program code
- Third-party utilities
- Interactive SQL queries
- Commands
- API invocation

This appendix does not describe:

- DataJoiner operations that are less likely to generate an error in Version 2.1.1 than in Version 1.2. These operations can have only a positive impact on existing applications.
- Inter-version differences that are common to DataJoiner and DB2. For a discussion of problems that can result from them, see “Appendix I. Incompatibilities between Releases”, in the *DB2 SQL Reference for common servers.*

The problems that this appendix addresses are those that can arise when applications that predate DataJoiner Version 2.1.1 try to:

- Link DataJoiner libraries to certain clients and data sources in AIX
- Use the **db2start** and **db2stop** commands to start and stop Classic Connect processes
- Query DataJoiner Version 2.1.1 catalog tables, or query DB2 for CS catalog views that have been updated for DataJoiner Version 2.1.1
- Modify DataJoiner Version 2.1.1 catalog tables

Linking DataJoiner Libraries to Clients and Data Sources in AIX

This section indicates:

- How the method for linking DataJoiner libraries to clients and data sources has changed in Version 2.1.1

- What problem can result when a Version 1.2 application tries to link Version 2.1.1 libraries to certain clients and data sources
- How to resolve this problem

Change

When you use Version 1.2 in the AIX environment, you edit djxlink.makefile and run the djxlink shell script to link DataJoiner libraries to Oracle and Sybase client libraries or to DRDA data sources accessed through APPC. When you use Version 2.1.1 in the AIX environment, you do not need to edit djxlink.makefile for most data sources. Simply running djxlink.sh will link DataJoiner libraries to libraries of nearly all data sources.

Problem

Running the Version 1.2 djxlink shell script in Version 2.1.1 does not work. This script is not shipped with Version 2.1.1.

Resolution

Run the Version 2.1.1 djxlink.sh script. For the small number of data sources that cannot be link-edited using djxlink.sh, edit djxlink.makefile to contain the data source library information. Then use djxlink.makefile to link libraries.

Starting and Stopping Classic Connect Instances

This section indicates:

- How the method for starting and stopping Classic Connect instances has changed in Version 2.1.1
- What problems can result when a Version 1.2 application tries to start and stop Classic Connect instances
- How to prevent this problem

Change

In Version 1.2, the **db2start** command starts Classic Connect processes and the **db2stop** command stops these processes. In Version 2.1.1, the **djxstart** and **djxstop** commands start and stop Classic Connect processes.

Problem

If, in Version 2.1.1, you issue a **db2start** command, the Classic Connect processes will not start. If you issue **db2stop**, the Classic Connect processes will not stop.

Resolution

To start and stop Classic Connect processes for a DataJoiner instance in Version 2.1.1, issue the **djxstart** and **djxstop** commands.

Querying System Catalog Tables and Views

This section explains:

- How DataJoiner catalog tables and DB2 for CS catalog views have been updated to support DataJoiner Version 2.1.1
- What problems can result when certain applications, such as those based on DataJoiner Version 1.2, try to query these tables or views
- How to resolve these problems

Changes

Changes have been made to several DataJoiner system catalog tables, and to certain DB2 for CS catalog views that support DataJoiner. This section discusses:

- Changes that could cause problems for applications designed to access catalog tables that were used by DataJoiner Version 1.2
- Changes that could cause problems for applications designed to access DB2 for CS views that have been updated to support the Spatial Extender.

Changes in Tables Used by DataJoiner Version 1.2

DataJoiner Version 1.2 uses three DB2 for CS catalog tables—SYSCOLUMNS, SYSINDEXES, and SYSTABLES—and two tables specific to DataJoiner—SYSREMOTEUSERS and SYSSERVERS. The following changes, listed by table, were made for DataJoiner Version 2.1 and retained in Version 2.1.1:

The SYSCOLUMNS Table: The following changes, listed by column, were made to this table:

HIGH2KEY
Non-character values are now in printable format rather than binary format.

LOW2KEY
Non-character values are now in printable format rather than binary format.

NULLS
The value D (not null with default) has been changed to N (not nullable).

REMOTE_TYPE
In Version 1.2, values denoted data types of columns of data source tables that DataJoiner

referenced by nickname. In Version 2.1.1, these values are stored in REMOTE_TYPENAME.

The SYSINDEXES Table: In Version 1.2, the value in the CLUSTERRATIO column of this table was -1 if statistics were not gathered. In Version 2.1.1, the value is -1 either if statistics are not gathered or if detailed index statistics are gathered. In the latter case, an appropriate value is added to the CLUSTERFACTOR column.

The SYSREMOTEUSERS Table: The data type for this table's AUTHID column was changed from CHAR to VARCHAR.

The SYSSERVERS Table: The following changes, listed by column, were made to this table:

COLSEQ
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value (colseq) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

CONNECTSTRING
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value (connectstring) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

CPURATIO
Data type changed from DOUBLE to FLOAT.

DATEFORMAT
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value (DATEFORMAT) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

FOLDID
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value (fold_id) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

IORATIO
Data type changed from DOUBLE to FLOAT.

PASSWORD
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value (password) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

TIMEFORMAT
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value (TIMEFORMAT) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

TIMESTAMPFORMAT
Deleted from SYSSERVERS. In Version 2.1.1, this server option is denoted by a value

(TIMESTAMPFORMAT) in the OPTION column of the SYSCAT.SERVER_OPTIONS catalog view.

The SYSTABLES Table: The following changes, listed by column, were made to this table:

PACKED_DESC
Data type changed from LONGVARCHAR to BLOB.

REL_DESC
Data type changed from LONGVARCHAR to BLOB.

VIEW_DESC
Data type changed from LONGVARCHAR to BLOB.

Changes in DB2 for CS Views That Support the Spatial Extender

The following DB2 for CS catalog views were changed to support the Spatial Extender, an optional facility that became available with DataJoiner Version 2.1.1. For information about the Spatial Extender, see *IBM DB2 Spatial Extender Administration Guide and Reference*.

The SYSCAT.DATATYPES View: The following columns were added to this view: EXTRA_LENGTH, TYPE_PRECEDENCE, and INSTANTIABLE.

The SYSCAT.FUNCPARMS View: The following columns were added to this view: PARMNAME, TYPE_PRESERVING, and MUTATED.

The SYSCAT.FUNCTIONS View: The following columns were added to this view: CONTAINS_SQL, DBINFO, RESULT_COLS, BODY, EFFECT, TYPE_PRESERVING, FUNC_PATH, and SELECTIVITY.

The SYSCAT.TRIGDEP View: A column named DTYPE was added to SYSCAT.TRIGDEP.

Problems

A variety of problems could occur. For example:

- If a DataJoiner Version 1.2 application does a qualified search on a column that takes a different value than it did before (for example, a search on NULLS in SYSIBM.SYSCOLUMNS for a value of D), the application might react differently than expected.
- If a DataJoiner Version 1.2 application queries a column whose data type has changed (for example, CPURATIO in SYSIBM.SYSSERVERS), too much or too little data might be returned.
- If a DB2 for CS application uses star notation (SELECT *) to query a view with new columns that the application doesn't recognize (for example,

SYSCAT.DATATYPES, which has several new columns to support the Spatial Extender), the application will receive an error.

Resolution

Review the changes listed above to decide whether they affect your applications and, if so, what corrective action to take (for example, updating the application). So that any problems in accessing or maintaining catalog tables can be avoided, we strongly recommend that instead of querying these tables, you query the catalog views derived from them.

If you need a rough approximation of the degree of clustering, select both CLUSTERRATIO and CLUSTERFACTOR in the SYSCAT.INDEXES catalog view and choose the greater of the two values that you retrieve.

Modifying System Catalog Tables

This section explains:

- How the method for modifying system catalog tables changed in Version 2.1.1
- What problems can result when Version 1.2 applications try to modify Version 2.1.1 catalog tables
- How to resolve these problems

Change

For DataJoiner to perform operations on a specific data source, DataJoiner must associate an identifier (specifically, a server name) with that data source. In Version 1.2, you could create such an association by inserting appropriate values into the table SYSIBM.SYSSERVERS. You could also modify an association by updating SYSIBM.SYSSERVERS, and terminate an association by deleting a server name from SYSIBM.SYSSERVERS. In Versions 2.1 and 2.1.1, you use DDL to perform these same operations indirectly. Specifically, you create DataJoiner-to-data source associations with the CREATE SERVER MAPPING statement, modify them with the ALTER SERVER MAPPING statement, and terminate them with the DROP statement. These statements operate on SYSCAT.SERVERS, a catalog view derived from SYSIBM.SYSSERVERS. The changes that you make to the view are propagated to SYSIBM.SYSSERVERS.

For a user to access data sources from DataJoiner, DataJoiner must associate the ID under which the user connects to DataJoiner with the IDs under which the user connects to these data sources. In Version 1.2, you could create such an association by inserting appropriate values into the table SYSIBM.SYSREMOTEUSERS. You could also modify an association by updating SYSIBM.SYSREMOTEUSERS, and terminate an association by deleting an ID from SYSIBM.SYSREMOTEUSERS. In Versions 2.1 and 2.1.1, you use DDL to perform these same operations indirectly. Specifically, you create associations between IDs with the CREATE USER MAPPING statement, modify them with the ALTER USER MAPPING statement, and terminate them with the DROP statement. These statements operate on SYSCAT.REMOTEUSERS, a catalog view derived from SYSIBM.SYSREMOTEUSERS. The changes that you make to the view are propagated to SYSIBM.SYSREMOTEUSERS.

Problem

If you issue an INSERT, UPDATE, or DELETE statement against SYSIBM.SYSSERVERS, SYSIBM.SYSREMOTEUSERS, or any of DataJoiner's other system catalog tables, the statement will fail.

Resolution

To modify SYSIBM.SYSSERVERS or SYSIBM.SYSREMOTEUSERS, use the SERVER MAPPING or USER MAPPING DDLs, as described in "Change" on page 356.

Appendix E. Viewing Documentation

Before you install DataJoiner, you might want to view the documentation that is included on the CD-ROM. To do so, you need to mount the CD-ROM using the following steps.

1. Insert the CD into the CD-ROM drive.
2. Login as root.
3. Type the following command on the AIX command line:

   ```
   mkdir -p /usr/cdrom
   ```

   where */usr/cdrom* is the name of your root directory.
4. Allocate a CD-ROM file system by typing:

   ```
   smitty storage
   ```
5. Select **File Systems**.
6. Select **Add / Change / Show / Delete File Systems**.
7. Select **CDROM File Systems**.
8. Select **Add a CDROM File System**.
9. Select the device name.

   Requirement:
   Device names for CD-ROM file systems must be unique. You might need to delete a previously defined CD-ROM file system or use another name for your directory if there is a conflict.
10. At the MOUNT POINT prompt, type the name of your root directory. For example:

    ```
    /usr/cdrom
    ```
11. Exit from smitty.
12. Mount the CD-ROM file system by typing the following command on the AIX command line:

    ```
    smitty mountfs
    ```
13. Select (or enter) the FILE SYSTEM name (for example, `/dev/cd0`).
14. Select (or enter) the DIRECTORY name: For example `/usr/cdrom`.
15. Select (or enter) the TYPE of file system: `cdrfs`.
16. Set Mount as READ ONLY system to YES.
17. Select **OK**.
18. Exit from smitty and logout as root.

Appendix F. Where to Learn More about DataJoiner, DB2 for CS, and Replication Products

This appendix lists IBM books about DataJoiner, DB2 for CS, and Replication Administration; states how to obtain these books; and tells you where to go on the Internet to learn more about DataJoiner.

DataJoiner, DB2 for CS, and Replication Publications

Table 40 on page 362 lists the DataJoiner, DB2 for CS, and Replication books applicable to installing, configuring, administrating, using, and running applications against DataJoiner. The *DataJoiner for UNIX Systems Planning, Installation, and Configuration Guide* and the *DataJoiner for Windows NT Systems Planning, Installation, and Configuration Guide* are provided in hardcopy with DataJoiner. In addition, these two books and all other DataJoiner books are provided in softcopy formats (PostScript, HTML, and PDF) on the product CD-ROM. All other books in Table 40 on page 362 are provided in PostScript; most are also provided in HTML (the two exceptions are the DB2 for CS Software Developer Kit publications). Additionally, most of the DB2 for CS books are provided in INF format (see Table 40 on page 362).

To understand how the DataJoiner books in Table 40 on page 362 are organized, it is important to understand how DataJoiner and DB2 for CS are interrelated. DataJoiner provides a "superset" of DB2 for CS. The two products share common functions and syntax; therefore, information that is common to DataJoiner and DB2 for CS is documented in the DB2 for CS books. The DataJoiner books listed in Table 40 on page 362 document the function and syntax that DataJoiner has *in addition to* the function and syntax that it shares with DB2 for CS.

Table 40 on page 362 does not list all of the DB2 for CS books. View or print a DB2 for CS book to see the publications list for all DB2 for CS books.

If you order Classic Connect, you will receive additional documentation:

- The *DataJoiner Classic Connect Installation, Configuration, and Reference Guide*, V2.1.1
- The *DataJoiner Classic Connect Data Mapper Installation and User's Guide*, V2.1.1
- A program directory

Table 40. DataJoiner, DB2 for CS, and Replication publications applicable to DataJoiner

Book Name	Form Number	File Prefix	INF
DataJoiner Version 2.1.1 Books			
DataJoiner for Windows NT Systems Planning, Installation, and Configuration Guide	SC26-9150	DJXN3	no
This book covers capacity planning, resource management, installation, and configuration tasks for IBM DataJoiner on Microsoft Windows NT operating systems.			
DataJoiner for UNIX Systems Planning, Installation, and Configuration Guide	SC26–9145	DJXG7	no
This book covers capacity planning, resource management, installation, and configuration tasks for IBM DataJoiner on AIX and Solaris operating systems.			
DataJoiner Administration Supplement	SC26–9146	DJXD5	no
This book provides information that assists DBAs and other system administrators of DataJoiner with performing administrative tasks. It includes a product overview section, security considerations, data source identification steps, database utility notes, performance considerations, database system monitor reference data, large object information, and explain tool examples.			
DataJoiner Application Programming and SQL Reference Supplement	SC26–9148	DJXK5	no
This book provides SQL statements, descriptions of system catalog data, guidelines, and other information for application programmers. With this information, application programmers can use DataJoiner to perform multiple tasks in a distributed database environment—tasks such as creating nicknames by which to reference tables and views, invoking functions and stored procedures, passing SQL directly to databases for processing, and using server options to optimize query performance.			
DataJoiner Generic Access API Reference	SC26–9147	DJXM4	no
This book explains how to create a generic access module that allows you to use existing drivers or to create new drivers to gain access to an unlimited set of data sources.			
DataJoiner Classic Connect Installation, Configuration, and Reference Guide	SC26–9319	DJXC5	no

Table 40. DataJoiner, DB2 for CS, and Replication publications applicable to DataJoiner (continued)

Book Name	Form Number	File Prefix	INF
This book provides information about the DataJoiner Classic Connect for MVS product. The audience for this information includes application programmers, database administrators, network administrators, system administrators, and system programmers. The book documents key tasks required to set up Classic Connect including: planning your setup; installing components using SMP/E; configuring the AIX and Windows NT client, data servers, enterprise servers, and network communications; managing instances; and creating relational data maps for IMS and VSAM data. This book also contains all system messages that pertain to Classic Connect.			
DataJoiner Classic Connect Data Mapper Installation and User's Guide	SC26–9318	DJXZ3	no
This book provides information about the DataJoiner Classic Connect data mapper utility for Windows. The audience for this information includes system programmers, database administrators, or anyone that needs to produce relational equivalent structures for IMS and VSAM data. The book documents key tasks required to set up and use the data mapper in the Windows environment: installing product files, starting the product, and generating meta data grammar for input to the DataJoiner Classic Connect meta data utility.			
DataJoiner Messages and Problem Determination Guide	SC26–9149	DJXP4	no
This book describes the messages and codes issued by DataJoiner and Classic Connect instances. For messages that report errors, the book explains the cause of the errors and recommends corrective actions. The book also provides guidelines on using diagnostic tools to isolate and understand problems.			
DB2 Spatial Extender Administration Guide and Reference	SC26–9316	DJXS1	no
This book provides instructions for spatially enabling a DataJoiner database, an introduction to spatial capabilities using geometry data types and functions, descriptions of spatial data exchange formats, an SQL and message reference for spatial data, and appendices containing the standard representations of spatial reference systems.			
DB2 for CS and Replication Books			
DB2 Information and Concepts Guide	S20H–4664	SQLG0	no
Provides product and conceptual information to anyone who needs a comprehensive overview of the DB2 products. It is useful when deciding which DB2 products suit your environment. It also includes a glossary of terms used in the book.			

Table 40. DataJoiner, DB2 for CS, and Replication publications applicable to DataJoiner (continued)

Book Name	Form Number	File Prefix	INF
DB2 Administration Guide	S20H-4580	SQLD0	yes
Contains information required to design, implement, and maintain a database to be accessed either locally or in a client/server environment.			
DB2 Database System Monitor Guide and Reference	S20H–4871	SQLF0	yes
Includes a description of how to use the Database System Monitor and a description of all the data elements for which information can be collected.			
DB2 Command Reference	S20H–4645	SQLN0	yes
Provides the reference information needed to use system commands and the DB2 command line processor to execute database administrative functions. Describes the commands that can be entered at an operating system command prompt or in a shell script to access the database manager. Explains how to invoke and use the command line processor, and describes the command line processor options. Provides a description of all the database manager commands.			
DB2 API Reference	S20H–4984	SQLB0	yes
Provides information about the use of application programming interfaces (APIs) to execute database administrative functions. Presents a description of APIs and the data structures used when calling APIs, as well as detailed information on the use of database manager API calls in applications written in the supported programming languages.			
DB2 SQL Reference	S20H–4665	SQLS0	yes
Is intended to serve as a reference for syntax and rules governing the use of SQL statements. Syntax diagrams, semantic descriptions, rules, and examples are provided for the SQL statements. Catalog views, product maximums, release-to-release incompatibilities, and a glossary are also included in this book.			
DB2 Application Programming Guide	S20H–4643	SQLA0	yes
Discusses the application development process and how to code, compile, and execute application programs that use embedded SQL to access the database. It includes discussions on programming techniques and performance considerations for the application programmer.			
DB2 Call Level Interface Guide and Reference	S20H–4644	SQLL0	yes

Table 40. DataJoiner, DB2 for CS, and Replication publications applicable to DataJoiner (continued)

Book Name	Form Number	File Prefix	INF
Is a guide and reference manual for programmers using the Call Level Interface. DB2 Call Level Interface is a callable SQL interface based on the X/Open CLI specification and is compatible with Microsoft Corporation’s ODBC.			
DB2 Messages Reference	S20H–4808	SQLM0	yes
Lists messages and explanations. Each explanation includes the action to be taken when a message or code is issued.			
DB2 Problem Determination Guide	S20H–4779	SQLP0	yes
Provides information that helps in determining the source of errors, recovering from problems, and describing and reporting defects.			
DDCS User’s Guide	S20H–4793	SQLC0	yes
Provides concepts, programming guidelines, and general information about the DDCS products.			
DB2 Replication Guide and Reference	S95H–0999	DB3E0	no
Describes how to plan, configure, administer, and operate IBM replication products, including the Apply and Capture programs.			
DB2 for CS Platform-Specific Books			
DB2 SDK for AIX Building Your Applications	S20H-4780	SQLA3	yes
This book provides environment setup information and step-by-step instructions to compile and link DB2 applications on the AIX operating system.			
DB2 SDK for Windows 95 and NT Building Your Applications	S33H-0310	SQLA6	yes
This book provides environment setup information and step-by-step instructions to compile and link DB2 applications on Windows 95 and NT operating systems.			
DB2 SDK for Solaris Building Your Applications	S34H-0890	SQLA5	yes

Table 40. DataJoiner, DB2 for CS, and Replication publications applicable to DataJoiner (continued)

Book Name	Form Number	File Prefix	INF
This book provides environment setup information and step-by-step instructions to compile and link DB2 applications on Sun’s Solaris operating system.			

How to Order, View, and Print Publications

Use order number SBOF-5289 to request one hardcopy of each of the DataJoiner, DB2 for CS, and Replication books shown in Table 40 on page 362.

To view online documentation, follow the instructions located in the README files on the CD-ROM. Most of the books in Table 40 on page 362 are provided as HTML files and can be viewed with an HTML browser. You can also view INF versions of many DB2 for CS books. Instructions for installing the INF reader on AIX are provided in the DB2 README files; on NT operating systems, the INF reader is installed automatically. DataJoiner and Replication information is not provided in INF format.

To print individual books, follow the instructions provided in the README files on the CD-ROM. PostScript files for all the books are provided.

Internet Resources

The following Internet resources provide additional information about DataJoiner.

World Wide Web

The following DataJoiner-specific Web site contains general and technical (frequently asked questions) product information. The address of the site is:

http://www.ibm.com/software/data/datajoiner

Also available online are the most current versions of books in the DB2 library. You can view books in the DB2 library by clicking the Library link from the following address:

http://www.ibm.com/software/data/pubs/techinfo.html

Internet Newsgroups

DataJoiner questions, answers, and discussions can be found in:

- bit.listserv.db2-l
- comp.databases
- comp.databases.ibm-db2

Appendix G. DataJoiner Classes and Services

This appendix describes:

- Classes you can take to learn about DataJoiner
- Services to help you plan to use DataJoiner, and to install and configure it

DataJoiner Classes

IBM offers classes that teach you how to install, use, and maintain DataJoiner. These classes are described in this section.

For more information, or to enroll in any IBM class, call 1-800-IBM-TEACH (1-800-426-8322) and refer to the IBM US Course Code. For locations outside the United States, contact your IBM representative.

Class descriptions will also be maintained at the DataJoiner Web site. The DataJoiner URL is:

http://www.ibm.com/software/data/datajoiner

Using DataJoiner

IBM US Course Code DW202

Duration
2 days

Format
Lecture with classroom exercises.

This course introduces the student to DataJoiner and its powerful multidatabase server capabilities. After completing this course, students should be able to effectively use DataJoiner to perform simple and complex distributed requests. They should also be able to monitor and tune SQL queries, accounting for the capabilities and characteristics of diverse DataJoiner data sources. Areas covered include:

- Global optimization
- Multi-vendor query considerations
- Nicknames
- Basic security
- An introduction to the DataJoiner catalog
- DataJoiner query performance
- The DataJoiner Explain tool

- The DataJoiner Database System Monitor

Who Should Take This Course
This course is appropriate for anyone who will be using, managing, installing, or maintaining a DataJoiner multiple database environment.

Prerequisite
SQL experience. You can obtain this experience by attending the “SQL Workshop,” IBM US Course Code CF120.

DataJoiner Administration

IBM US Course Code DW212

Duration
3 days

Format
Lecture with classroom exercises.

This course trains the student to install, configure, and manage a secure DataJoiner multidatabase server environment. Areas covered include:

- Installing DataJoiner
- Generating and managing the DataJoiner database
- Configuring DataJoiner
- Enabling DataJoiner client access to remote data sources
- DataJoiner security
- DataJoiner server performance

Who Should Take This Course
This course is appropriate for anyone who will be managing, installing, or maintaining a DataJoiner multiple database environment.

Prerequisite
DataJoiner knowledge or experience. You can obtain this experience by attending “Using DataJoiner,” IBM US Course Code DW202.

DataJoiner Services

IBM provides services for DataJoiner that include assistance with planning, installing, and configuring the product. The assistance is customized to your individual environment and takes place in two phases.

First Phase: Planning

The first phase helps you plan the installation and configuration of DataJoiner, and to configure network systems so that DataJoiner can communicate optimally with all data sources and clients. This phase includes:

- Assessing general readiness

- Defining clients
- Defining data sources
- Assessing applications
- Defining backup and recovery strategies for DataJoiner
- Configuring DataJoiner database parameters
- Identifying test queries for system validation
- Defining security requirements

Second Phase: Implementation

The second phase focuses on implementing the plan developed in the planning phase. It includes:

- Installing DataJoiner
- Configuring data sources
- Providing access to data source tables and views
- Installing and configuring remote clients
- Validating and documenting the environment
- Providing final turnover to the customer

At the end of this phase, active remote and local clients can access multiple data sources through DataJoiner.

DataJoiner services can be combined with replication services if you are interested in replicating data across a heterogeneous database environment. For more information about DataJoiner and replication services, contact your IBM representative or see the DataJoiner Web page. The DataJoiner URL is:

http://www.ibm.com/software/data/datajoiner

Notices

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

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

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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.

COPYRIGHT LICENSE:

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

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, or other countries, or both:

ADSTAR
Advanced Peer-to-Peer Networking
AIX
APPN
AS/400
AT
CICS
CICS/6000
Client Acces
Current
DATABASE 2
DataGuide
DataJoiner
DataPropagator
DataRefresher
DB2
DFSMS
Distributed Relational Database Architecture
DPROP
DRDA
Extended Services for OS/2
HACMP/6000
IBM
IIN
IMS
IMS/ESA
Language Environment
MVS
MVS/ESA
MVS/XA
NetView
Operating System/2
Operating System/400
OS/2
OS/390
OS/400
RACF
RETAIN
RISC System/6000
RS/6000
RT
SP
SQL/DS
SQL/400
System/390
VisualAge
VTAM

Intel is a registered trademark of the Intel Corporation in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Solaris is a trademark of Sun Microsystems, Inc.

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

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

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

Index

Readers' Comments — We'd Like to Hear from You

**DB2 DataJoiner® for UNIX® Systems
Planning, Installation,
and Configuration Guide**

Publication No. SC26-9145-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

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

Name

Address

Company or Organization

Phone No.

Readers' Comments — We'd Like to Hear from You

SC26-9145-02

Cut or Fold
Along Line

Fold and Tape

Please do not staple

Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM CORPORATION
Department BWE/H3
PO Box 49023
San Jose, CA 95161-9945

Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line

Part Number: CT6DFIE

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

SC26-9145-02

CT6DFIE

Spine information:

IBM

DB2 DataJoiner® for UNIX® Systems

Planning, Installation, and Configuration Guide

Version 2
Release 1 Modification
1