Informix Enterprise Gateway with DRDA

User Manual

Version 7.31
January 2000
Part No. 000-6492

Published by Informix® Press

Informix Corporation
4100 Bohannon Drive
Menlo Park, CA 94025-1032

Documentation Team: Evelyn Eldridge, Barbara Nomiyama, Tom Noronha

Table of Contents

Chapter 5 The gwdba Utility

Introduction

In This Introduction

This chapter introduces the *Informix Enterprise Gateway with DRDA User Manual*. Read this chapter for an overview of the information that this manual provides and for an understanding of the conventions that are used throughout this manual.

About This Manual

The *Informix Enterprise Gateway with DRDA User Manual* is a guide to how to use Informix Enterprise Gateway with DRDA. This manual documents the features and functionality of the Informix Enterprise Gateway with DRDA product. Throughout the manual, Informix Enterprise Gateway with DRDA is frequently referred to as the Gateway.

Organization of This Manual

This manual includes the following chapters:

- This Introduction provides an overview of the manual and describes the documentation conventions that are used.
- Chapter 1, "Gateway Concepts," provides background information about Informix Enterprise Gateway with DRDA. This chapter provides a foundation for the techniques that later chapters discuss.
- Chapter 2, "Installation," gives instructions for installing a new version of Informix Enterprise Gateway with DRDA.
- Chapter 3, "Configuration," tells how to configure the Gateway and how to prepare the necessary files for network connections.

- Chapter 4, "Global Language Support," discusses the environment variables and error messages associated with the Global Language Support (GLS) features of the Gateway.
- Chapter 5, "The gwdba Utility," gives instructions for how to use the **gwdba** utility to prepare user records, bind packages, and prepare system catalogs.
- Chapter 6, "Programming with the Gateway," provides a summary of the behavior of Structured Query Language (SQL) statements in the Informix Enterprise Gateway with DRDA environment. It also gives programming suggestions for some SQL statements and information about error treatment.
- Chapter 7, "The bcheckgw Utility," discusses the **bcheckgw** utility, which checks the integrity of files.

Types of Users

This manual is written for Informix Enterprise Gateway with DRDA system administrators and application developers who use Informix Enterprise Gateway with DRDA.

This manual assumes that you have already installed and used other Informix products. It also assumes that you are familiar with Distributed Relational Database Architecture (DRDA)-compliant database servers and have a network environment that can run Informix Enterprise Gateway with DRDA.

Software Dependencies

Informix Enterprise Gateway with DRDA enhances the capabilities of Informix client applications by allowing access to IBM DRDA databases. Informix Enterprise Gateway with DRDA supports connectivity to DRDA-compliant databases such as:

- IBM DB2 for OS/390
- IBM DB2 for MVS/ESA
- IBM DB2 for OS/400
- IBM DB2 for VM (SQL/DS)

SQL/DS is also known as DB2/VM and DB2/VSE. The Gateway works only with DB2/VM. The name of the database manager for OS/400 is DB2 for OS/400. See the release notes file for information about supported data sources and versions.

You can use Informix Enterprise Gateway with DRDA with various Informix application-development tools and database servers. The Gateway provides distributed database access to DRDA-compliant application servers for the following Informix products:

- Informix Dynamic Server 2000
- Informix Internet Foundation 2000
- Informix Client Software Developer's Kit Version 2.30 and above
- Informix Dynamic Server with Universal Data Option Version 9.1x
- Informix Dynamic Server Version 7.3x
- INFORMIX-OnLine Dynamic Server Version 7.2x
- INFORMIX-OnLine Database Server Version 5.1x and above
- Informix ESQL/C Version 7.2x and INFORMIX-ESQL/COBOL Version 7.2x
- Informix ESQL/C Version 5.1x and INFORMIX-ESQL/COBOL Version 5.1x
- INFORMIX-4GL Version 7.2x and above

When the Gateway participates in a distributed transaction, the Gateway behaves as a subordinate database server to a coordinating Informix Dynamic Server.

For a discussion of how to use Informix products with Informix Enterprise Gateway with DRDA, see "Using Informix Products with the Gateway" on page 6-105. For the latest information on product compatibility, see the online release notes file.

New Features of This Product

This section highlights the new features that are implemented in Version 7.31 of Informix Enterprise Gateway with DRDA.

- Enhanced mapping of Informix built-in functions

  This release supports enhanced mapping of SQL functions. This mapping enables use of Informix built-in SQL functions that have corresponding DB2 equivalents with a different name and identical behavior.

- New **GWIFXSEMANTIC** environment variable

  This release supports the new **GWIFXSEMANTIC** environment variable. This environment variable enables you to specify the use of Informix behavior for the built-in SQL functions that have the same name as functions in the application server but have different behavior from the functions in the application server.

- Performance improvements in distributed queries

  This release supports performance improvements in distributed queries by providing more statistical information to the Informix Server optimizer.

- ANSI outer joins

  This release supports ANSI outer joins that involve Informix tables and application server tables.

- Displaying the software version number using **gwdba**

  You can display the name, version number, and serial number for the Gateway by using the **-V** option of the **gwdba** utility.

- Support for delimited identifiers

  This release of the Gateway supports delimited identifiers. Support for delimited identifiers means that you can access any column of any table at a target application server, no matter how the tables or columns are named.

- New **GWUSESYSCOLNAME** environment variable

  The **GWUSESYSCOLNAME** environment variable is used only when the Gateway is connected to OS/400 version 3.1 or higher. When this environment variable is set, and the Gateway queries for catalog information about columns of an object in **qsys2.syscolumns**, the Gateway will look up the SYS_CNAME (SYSTEM_COLUMN_NAME) field instead of the NAME (COLUMN_NAME) field to find the name of the object's column.

- New **GWKEEPDYNAMIC** environment variable

  The **GWKEEPDYNAMIC** environment variable is used only when the Gateway is connected to DB2 for OS/390 version 5.0 or higher. When a package is bound or rebound with the **Bind** option KEEPDYNAMIC=YES, the DB2 server keeps dynamic statements past commit points for an application process so that performance improvements can occur. To instruct the Gateway to take advantage of these performance benefits, you set the **GWKEEPDYNAMIC** environment variable in the Gateway environment.

- Mixed-mode database access

  The Gateway now allows access from non-logged Informix databases to DB2 databases.

Conventions

This section describes the conventions that this manual uses. By becoming familiar with these conventions, you can more easily gather information from this and other volumes in the documentation set.

The following conventions are discussed:

- Typographical conventions
- Icon conventions
- Command-line conventions

Typographical Conventions

This manual uses a standard set of conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth. The following typographical conventions are used throughout this manual.

Convention	Meaning
italics	Within text, new terms and emphasized words are printed in italics. Within syntax diagrams, values that you are to specify are printed in italics.
boldface	Identifiers (names of classes, objects, constants, events, functions, program variables, forms, labels, and reports), environment variables, database names, table names, column names, menu items, command names, and other similar terms are printed in boldface.
`monospace`	Information that the product displays and information that you enter are printed in a monospace typeface.
KEYWORD	All keywords appear in uppercase letters.
♦	This symbol indicates the end of product- or platform-specific information.

Tip: *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after the entry. When you are instructed to "type" the text or to "press" other keys, no* RETURN *is required.*

Icon Conventions

Throughout the documentation several different types of icons identify different types of text. Icons appear in the left margin of the page and pertain to the paragraph on the right. In some cases, the icon pertains to multiple paragraphs.

Comment icons identify three types of information, as described in the following table. This information is always displayed in italics.

Icon	Description
	Identifies paragraphs that contain vital instructions, cautions, or critical information.
	Identifies paragraphs that contain significant information about the feature or operation that is being described.
	Identifies paragraphs that offer additional details or shortcuts for the functionality that is being described.

Command-Line Conventions

Informix Enterprise Gateway with DRDA supports a variety of command-line options. You enter these commands at the operating-system prompt to perform certain functions. For example, Chapter 7 describes the **bcheckgw** utility command syntax.

This section defines and illustrates the format of the commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper left with a command. It ends at the upper right with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in italics.

You might encounter one or more of the following elements on a command-line path.

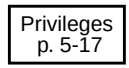
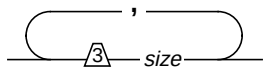
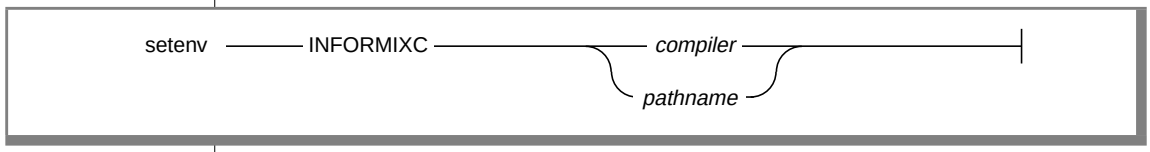
Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and must use lowercase letters.
variable	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.
-flag	A flag is usually an abbreviation for a function, menu, or option name or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as **.sql** or **.cob**, might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file and a period. The extension might be optional in certain products.
(.,;+*-/)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.
' '	Single quotes are literal symbols that you must enter as shown.
 	A reference in a box represents a subdiagram on the same page (if no page is supplied) or another page. Imagine that the subdiagram is spliced into the main diagram at this point.
	A shaded option is the default action.
	A gate (3) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. Here you can specify *size* no more than three times within this statement segment.

Figure 1 shows how you read the command-line diagram for setting the **INFORMIXC** environment variable.

Figure 1
An Example Command-Line Diagram



To construct a correct command, start at the top left with the command `setenv`. Then follow the diagram to the right, including the elements that you want. The elements in the diagram are case sensitive.

To read the example command-line diagram

1. Type the word `setenv`.
2. Type the word `INFORMIXC`.
3. Supply either a compiler name or pathname.

   After you choose *compiler* or *pathname*, you come to the terminator. Your command is complete.
4. Press ENTER to execute the command.

Additional Resources

The Informix Enterprise Gateway with DRDA documentation set includes printed manuals and on-line manuals.

This section describes the following pieces of the documentation set:

- Printed documentation
- On-line documentation
- Vendor-specific documentation
- Related reading

Printed Documentation

You need to refer to the related Informix product documents for the application-development tool that you are using. You might also want to refer to the following manuals:

- If you have an Informix database server, you or your system administrator need one of the following documents:
  - *INFORMIX-OnLine Administrator's Guide* (INFORMIX-OnLine versions prior to 6.0)
  - *INFORMIX-OnLine Dynamic Server Administrator's Guide* and *INFORMIX-OnLine Dynamic Server Archive and Backup Guide* (INFORMIX-OnLine Dynamic Server versions 6.0 through 7.2x)
  - *Administrator's Guide for Informix Dynamic Server* and *Archive and Backup Guide for Informix Dynamic Server* (Informix Dynamic Server versions beginning with 7.3)
  - *Administrator's Guide for Informix Dynamic Server 2000* and *Archive and Backup Guide for Informix Dynamic Server 2000* (Informix Dynamic Server 2000 and Informix Internet Foundation 2000)
- The *Informix Migration Guide* describes the procedures that you use when you move data from one location to another and when you migrate existing Informix databases to and from Informix Dynamic Server 2000.
- If you want additional information on the implementation of GLS in Informix products, see the *Informix Guide to GLS Functionality*.
- If you have never used SQL or an Informix application-development tool, you might want to read the *Informix Guide to SQL: Tutorial* for a task-oriented approach to learning SQL. You might also want to read the *Informix Guide to Database Design and Implementation* to learn basic database design and implementation concepts.

- Companion volumes to the *Informix Guide to SQL: Tutorial*, the *Informix Guide to SQL: Syntax* and *Informix Guide to SQL: Reference*, provide full information on the structure and contents of the demonstration database. These manuals include details of the Informix system catalog, describe Informix and common UNIX environment variables that should be set, and define column data types that Informix products support. In addition, they provide detailed descriptions of all the SQL statements that Informix products support and a glossary of useful terms.

On-Line Documentation

Different types of on-line documentation are available:

- On-line error messages
- Release notes, documentation notes, and machine notes

Error Message Files

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions. To read the error messages in the ASCII file, Informix provides scripts that let you display error messages on the screen (**finderr**) or print formatted error messages (**rofferr**).

Release Notes, Documentation Notes, and Machine Notes

In addition to the Informix set of manuals, the following on-line files, located in the **$INFORMIXDIR/release/en_us/0333** directory, might supplement the information in this manual.

On-Line File	Purpose
Documentation notes	Describes features that are not covered in the manuals or that have been modified since publication. The file is **GATEWAYDOC_7.31**.
Release notes	Describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file, **GATEWAYREL_7.31**, also contains information about any known problems and their workarounds.
Machine notes	Describes any special actions that are required to configure and use Informix products on your computer. Machine notes are named for the product that is described. For example, the machine-notes file for Informix Enterprise Gateway with DRDA is **GATEWAY_7.31**.

Please examine these files because they contain vital information about application and performance issues.

Vendor-Specific Documentation

In addition to the documentation provided by Informix, you need documentation about the remote DRDA-compliant database(s) that you will access. You also need documentation about how to install and maintain the SNA networking software for your hardware.

You might want to refer to the following IBM DRDA documentation:

- *Distributed Relational Database: Every Manager's Guide*, GC26-3195
- *Distributed Relational Database Application Programming Guide*, SC26-4773
- *Distributed Relational Database Connectivity Guide*, SC26-4783 (ISBN 0-3-398306-4)

- *Distributed Relational Database Problem Determination Guide*, SC26-4782
- *Distributed Relational Database Architecture*, SC26-4651
- *Planning for Distributed Relational Databases*, SC26-4650
- *DATABASE2 for MVS/ESA, Version 4, SQL Reference*, SC26-3270-00
- *DATABASE2 for OS/390, Version 5, SQL Reference*, SC26-8966-00

You might find the following materials useful when you make the LU 6. 2 connections:

- *SNA Configuration Reference Guide* (SC31-7014)
- *Systems Network Architecture: Formats* (GA27-3136)
- *SNA Formats and Protocol Reference Architecture and Logic for LU Type 6.2* (SC30-3269)
- *SNA Logical Unit Version 6 Reference: Peer Protocols* (SC31-6808)
- *SNA Transaction Programmers' Reference for LU Type 6.2* (GC30-3084)

Refer also to the SNA guide provided with your UNIX system.

Related Reading

For additional technical information on database management, consult the following books:

- *An Introduction to Database Systems* by C. J. Date (Addison-Wesley Publishing Company, Inc., 1994).
- *Transaction Processing: Concepts and Techniques* by Jim Gray and Andreas Reuter (Morgan Kaufmann Publishers, Inc., 1993)

To learn more about the SQL language, consider the following books:

- *A Guide to the SQL Standard* by C. J. Date with H. Darwen (Addison-Wesley Publishing Company, Inc., 1993)
- *Understanding the New SQL: A Complete Guide* by J. Melton and A. Simon (Morgan Kaufmann Publishers, Inc., 1993)
- *Using SQL* by J. Groff and P. Weinberg (Osborne McGraw-Hill, 1990)

This manual assumes that you are familiar with your computer operating system. If you have limited UNIX system experience, consult your operating-system manual or a good introductory text before you read this manual. The following texts provide a good introduction to UNIX systems:

- *Introducing the UNIX System* by H. McGilton and R. Morgan (McGraw-Hill Book Company, 1983)
- *Learning the UNIX Operating System* by G. Todino, J. Strang, and J. Peek (O'Reilly & Associates, 1993)
- *A Practical Guide to the UNIX System* by M. Sobell (Benjamin / Cummings Publishing, 1989)
- *UNIX System V: A Practical Guide* by M. Sobell (Benjamin / Cummings Publishing, 1995)

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.L35-1992), which is identical to ISO 9075:1992 on Informix Dynamic Server. The Gateway allows the use of SQL-92 Entry Level SQL syntax as well as certain extensions to the ANSI standard.

In addition, many features of Dynamic Server comply with SQL-92 Intermediate and Full Level and X/Open CAE (common applications environment) standards. Informix SQL-based products have been enhanced to comply with the X/Open SQL CAE specifications. The X/Open Schema views used by the Gateway are based on the X/Open XPG/4 SQL CAE specifications.

Further, IBM has issued a specification known as Distributed Relational Database Architecture (DRDA). The DRDA specification allows interoperability among different clients and servers. Informix Enterprise Gateway with DRDA complies with IBM DRDA Level 1 specifications.

The treatment of code-set conversions by the Gateway follows the IBM Character Data Representation Architecture (CDRA) and Formatted Data: Object Content Architecture (FD:OCA).The DRDA, LU 6.2, DDM, FD:OCA, and CDRA behaviors of the Gateway follow the standards as described in the following IBM publications:

- *Distributed Relational Database Architecture Reference* (SC26-4651-0)
- *DDM Architecture Reference Level 3* (SC21-9526-03)
- *Character Data Representation Architecture Level 1* (SC09-1390-00)
- *Formatted Data Object Content Architecture Reference* (SC31-6808-0)

The GLS functionality in Informix products is based on NLS standards as specified by X/Open in the XPG4 documents.

Informix Welcomes Your Comments

Please let us know what you like or dislike about our manuals. To help us with future versions of our manuals, tell us about any corrections or clarifications that you would find useful. Include the following information:

- The name, version, and page number of the manual you are using
- Any comments you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
SCT Technical Publications Department
4100 Bohannon Drive
Menlo Park, CA 94025

If you prefer to send electronic mail, our address is:

doc@informix.com

Or, send a facsimile to the Informix Technical Publications Department at:

650-926-6571

We appreciate your feedback.

Chapter

1

Gateway Concepts

In This Chapter

This chapter describes Informix Enterprise Gateway with DRDA and introduces some of the concepts you need to use the Gateway. Throughout this manual, Informix Enterprise Gateway with DRDA is referred to as the Gateway.

You can install the Gateway software on a computer that can support SNA LU 6.2 (SNA Logical Unit 6.2) communication with synchronous data link control (SDLC) or Token Ring. The Gateway can also use TCP/IP network protocol if it is supported by the DRDA application server.

The communications link between the Gateway computer and the IBM computer can be any connection that supports LU 6.2 or TCP/IP.

What Is DRDA?

The *Distributed Relational Database Architecture* (DRDA) defined by IBM is a set of protocols that software manufacturers can follow to develop connectivity solutions between heterogeneous relational database management environments. DRDA defines what you must exchange and how you must exchange it to coordinate communication between an *application requestor* (AR) and an *application server* (AS). Figure 1-1 illustrates this configuration.

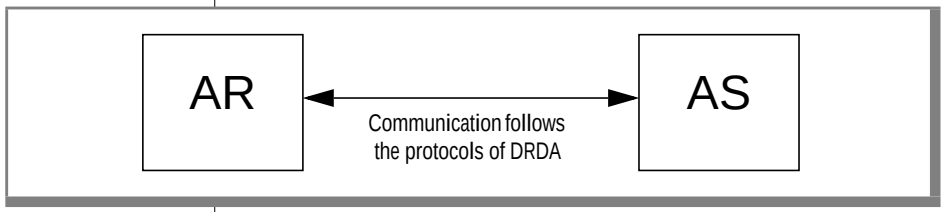


Figure 1-1
Relationship Between the Application Requestor and the Application Server

DRDA provides structure and guidelines, so that a properly constructed external application (in this case, an Informix client application) can interact with any database that follows the DRDA protocols.

The DRDA architecture includes the following definitions:

- Protocols for communication between hosts (SNA LU 6.2, TCP/IP)
- Definitions for distributed data-management commands (DDM)
- Definitions of the representation of data (FD:OCA)
- Description of character data (CDRA)

The current level of DRDA support that the Gateway provides is called DRDA-1, also known as Remote-Unit-of-Work (RUOW). This architecture dictates the behavior of both low-level and high-level functions. On a low level, DRDA dictates how to use the LU 6.2 or TCP/IP. On a high level, DRDA (specifically DRDA-1) dictates the behavior of distributed data management (DDM) commands.

For more information about DRDA, refer to the publications listed in "Vendor-Specific Documentation" on page 14 of the Introduction.

What Is Informix Enterprise Gateway with DRDA?

Informix Enterprise Gateway with DRDA acts as a bridge, or *gateway*, between an Informix client application and a DRDA-compliant database server. The Gateway allows you to access information from a DRDA-compliant database with an Informix client application.

The technology underlying the Gateway is based on the DRDA protocol, as defined by IBM. The Gateway enables an Informix client application to access and modify information in a database that supports the DRDA protocols. Conceptually, as Figure 1-2 on page 1-4 illustrates, the Gateway appears to the Informix application as a database server. The Gateway appears as a client application to the DRDA-compliant database server.

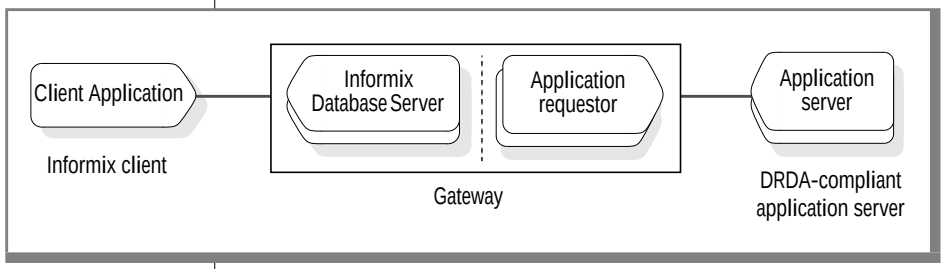


Figure 1-2
Role of the Gateway for Accessing a DRDA Server from an Informix Client Application

In the terminology defined and used in the IBM DRDA documentation, the Gateway appears to the DRDA-compliant database server as an AR and the DRDA-compliant database server, typically on an IBM computer, appears to the Gateway as an AS. In this manual, the DRDA-compliant database server is referred to as an *application server*. In contrast, the Informix database servers are referred to by name or as *database servers*.

What Does the Gateway Support?

Informix Enterprise Gateway with DRDA supports access to DRDA-compliant IBM data sources such as: IBM DB2 for OS/390, IBM DB2 for MVS/ESA, IBM DB2 for OS/400, and IBM DB2 for VM (SQL/DS). For further information about IBM data sources supported by the Gateway, see "Software Dependencies" on page 4 of the Introduction.

The Gateway provides connectivity to conforming DRDA databases for Informix products such as: Informix Dynamic Server 2000 and earlier database servers, Informix Client Software Developer's Kit, INFORMIX-ESQL/C, and INFORMIX-4GL. For further information about Informix products supported by the Gateway, see "Software Dependencies" on page 4 of the Introduction.

The Gateway is designed so that client applications that are written, for example, in INFORMIX-ESQL/C can issue requests to the Gateway as if it were an Informix database server. Details about special considerations that are required when you use the Gateway are covered in Chapter 6, "Programming with the Gateway."

How Does Informix Enterprise Gateway with DRDA Work?

The Gateway acts as a bridge between an Informix client application and an application server. The Gateway performs the following functions:

- Manages connections to the application server
- Uses formats and protocols that are acceptable to the application server to repackage SQL requests from Informix client applications
- Receives responses (data or status information) from the application server and repackages them in formats that are acceptable to the Informix client applications
- Handles implicit character code-set conversion when it encounters dissimilar code sets

Connecting to the Application Server

Informix client applications use CONNECT or DATABASE statements to connect to an application server. You use the Gateway name and an alias for the database name to specify the connection to the application server. The alias name (*alias_RDB_name*) refers to a unique configuration pair made up of the name of the DRDA-compliant database (the *real_RDB_name*) and a network communications mode. For instructions about how to prepare the configuration, see "Connection to the Application Server" on page 3-19.

Each Informix client application that connects to an application server is served by a separate Gateway process. For network connections, the Gateway administrator starts a daemon that spawns a separate process for each connection to the Gateway. For local connections, the process is initiated by the connection request from the client application. In a UNIX local-area network, a single instance of the Gateway daemon is usually sufficient to connect multiple client applications to multiple application servers.

Figure 1-3 on page 1-7 shows a conceptual diagram of an Informix client application that uses the Gateway to attach to an application server. The client application makes its initial connection request to the Gateway daemon (grey arrow).

The Gateway daemon starts a new Gateway process to serve the client application, connects the application to the Gateway process (black arrow), and then detaches itself. The Gateway then initiates a network conversation with the application server, using information that is supplied in the **$INFORMIXDIR/etc/sqlhosts** file. Preparation of the **sqlhosts** file is described in "Syntax of the sqlhosts File" on page 3-14.

The client application and the Gateway can be on different computers or on the same computer. The procedures for how to set up the Gateway in various configurations are discussed in Chapter 3, "Configuration."

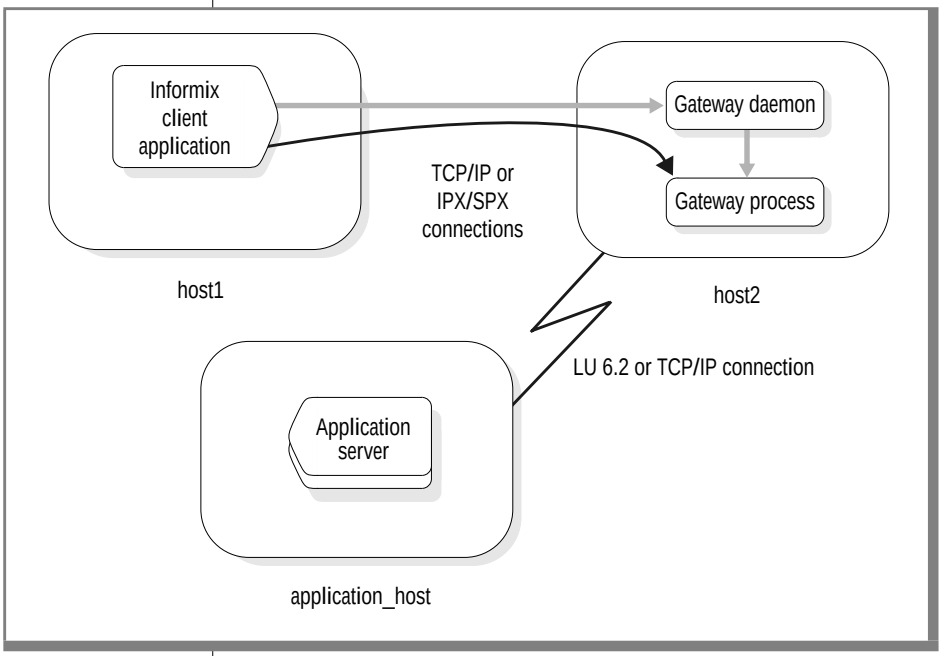


Figure 1-3 *Conceptual Illustration of a Gateway Connection*

Connecting to a DRDA Application Server Without a Password

When you use a user ID and password pair to connect to the Gateway, if a match for that user ID and target RDB name does not exist in the **gwuser** file, the Gateway uses the supplied password with the uppercased user ID to connect to the DRDA application server. For information about the **gwuser** file, see "The User Screen" on page 5-7.

If the remote relational database (RDB) is configured so that connections are allowed without a password (*already verified* security option), this requires the Gateway client to specify the correct password for the RDB user ID because the Gateway passes on the password to the RDB.

You can set the **GWNOPWD** environment variable to keep the password from being sent to the RDB if a matching entry for the user ID does not exist in the **gwuser** file. The user ID is uppercased and sent without a password to the RDB. This way, the password for the RDB user ID need not be maintained at the Gateway computer in the **gwuser** file or the operating-system password file to use an RDB account with the *already verified* security option.

SQL Statement Processing

The Gateway receives requests and responds to requests from the client application as if it were an Informix database server. However, instead of accessing the data directly, the Gateway converts the request to the corresponding DRDA protocol request and forwards the converted request to the application server.

The application server treats all SQL statements that the Gateway forwards as dynamic SQL statements, even statements that are coded as static statements in the application.

When you use the Gateway to access an application server, you can use any ANSI SQL statement and also some Informix extensions. For detailed information about how to prepare a client application for use with the Gateway, see Chapter 6, “Programming with the Gateway.”

Data Type Compatibility

The Gateway maps Informix data types to DRDA data types and vice versa, so that the application handles database data types as it normally would and so that the data on the application server side is handled correctly. Floating-point data is mapped to the correct format. Where necessary and appropriate, the format of the other data is also converted.

Information about data types that are used by the Gateway is provided in “Mapping Informix and IBM Data Types” on page 6-81.

Distributed Queries

A client application can connect directly to the Gateway or to a Dynamic Server, which connects to the Gateway as part of a distributed transaction, as Figure 1-4 on page 1-10 illustrates.

This manual uses the definitions in the following table to describe different types of connections and database access.

Term	Definition
Direct access	When the client connects directly to the Gateway to access a DRDA-compliant application server. To the client itself, the Gateway appears to be a database server. Figure 1-3 on page 1-7 shows an example of direct access. For direct access, the client can make either a local connection with unnamed pipes or a network connection.
Distributed access	When the client connects to a Dynamic Server and the server connects to the Gateway to access the application server. Figure 1-4 on page 1-10 shows an example of distributed access. For distributed access, Dynamic Server must use a network connection to the Gateway.
Distributed query	An SQL statement that is executed by a Dynamic Server that references one or more tables on database servers different than the Dynamic Server where the statement is being executed.
Distributed transaction	A transaction that holds resources at more than one database or application server.

Important: *The Gateway does not support two-phase commits. If an update statement (UPDATE, INSERT, or DELETE) is issued against an application server during a distributed transaction, all the other sites must be read-only during the transaction. That is, in one transaction, you can update only a single application server. However, if all application servers in a distributed transaction are read-only, then any number of Dynamic Servers can be updated within the same transaction.*

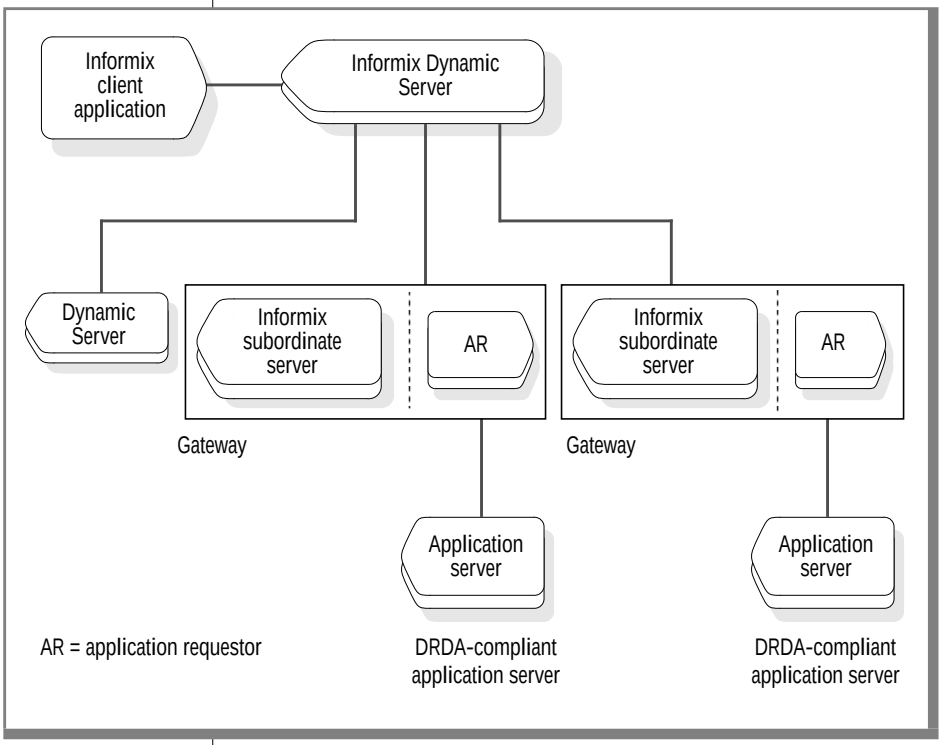


Figure 1-4
Distributed Query with Informix Dynamic Server as Coordinator

For a discussion of distributed queries, refer to "Considerations for Distributed Access to the Gateway" on page 6-43.

Catalog Information

In a distributed transaction that involves multiple databases, the coordinating Dynamic Server needs information about the remote objects that an application references. As with Dynamic Server, each application server that the Gateway supports has its own system catalog. This system catalog records schema information about the tables and columns that form the database.

The catalogs of application servers differ from each other and from the Informix catalog. Informix tools that require catalog information cannot operate with the Gateway unless Informix-style catalog information is available.

When the Gateway needs schema information for an object, depending on the **GWCATALOG** environment variable setting, the Gateway queries the appropriate catalog on the application server and passes the schema information to the Dynamic Server coordinator.

For more information on access to catalog information, see "Using the GWCATALOG Environment Variable to Access Catalog Information" on page 6-62.

The Informix Catalog

The catalog option of the **gwdba** utility (page 5-31) lets you create and populate an Informix catalog on an application server for Informix and third-party tools to use. The presence of an Informix catalog lets the Gateway appear more like an Informix engine to client applications.

The Informix catalog on the application server is itself a Dynamic Server system catalog, and contains all the required Dynamic Server catalog tables. Only the tables that the following list shows are actually populated. The **"INFORMIX".sysusers** table contains only one row, a row that lists **public** as having the database privilege:

- **"**INFORMIX**".systables**
- **"**INFORMIX**".syscolumns**
- **"**INFORMIX**".sysdepend**
- **"**INFORMIX**".syssyntable**
- **"**INFORMIX**".sysviews**
- **"**INFORMIX**".sysindexes**
- **"I**NFORM**IX".sysusers**

Informix system catalog tables are described in the *Informix Guide to SQL: Reference*.

Tools and applications can use any of the following formats to reference catalog tables:

- **"informix".sys*xxx***
- INFORMIX**.sys***xxx*
- **"INFORMIX".sys*xxx***
- **informix.sys*xxx***

Single Authorization ID for Informix System-Catalog Objects

The Version 6.1 **gwdba** utility used two authorization IDs, "**informix**" and "**INFORMIX**", to implement the Informix-style system catalog at the DRDA application server. Base tables were created under the authorization ID "**informix**"(noncollection library for OS/400) and views on these objects were created under the authorization ID "**INFORMIX**".

The Version 7.1 and later **gwdba** utility requires only the "**INFORMIX**" authorization ID. All catalog objects are created as base tables under the "**INFORMIX**" authorization ID. The Gateway also modifies incoming queries and replaces references to the "**informix**" authorization ID with the "**INFORMIX**" authorization ID.

Performance Improvements with the Informix Catalog

The Informix catalog also improves the performance of distributed queries that use the Gateway. The information that the Gateway requires to optimize distributed joins is readily available in the Informix catalog, making a search through the application-server catalog unnecessary. The Informix catalog contains only those tables and views that the user wants to access through the Gateway. Because the Informix catalog contains only the relevant tables and views, queries against it run faster than queries against the application-server catalog.

Static Information in the Informix Catalog

The Informix catalog has one limitation: the information in the catalog is static. Therefore, you must refresh the Informix catalog if you alter or drop tables, or if you create or drop indexes. The X/Open-style information schema, which is discussed in "The X/Open-Style Information Schema" on page 1-13, does not have this limitation, but it is *dramatically* slower to query.

The Application-Server Catalog

If no Informix catalog exists on the application server or if the necessary table or view is missing, the Gateway looks for the information in the application-server catalog.

Network Conversations for Catalog Access

When a client connects directly to the Gateway, the client establishes a single network conversation between the Gateway and the application server. When the Gateway participates in a distributed transaction, it can either issue catalog-access queries in the same conversation as the user's queries or start a separate conversation exclusively for the catalog queries.

The default is a single conversation in distributed mode. If the **GWTWOCON** environment variable is set, the Gateway uses the authorization ID **informix** to start a separate conversation. In either case, catalog queries that the Gateway issues are prepared and bound during package binding and are executed only when an application uses the Gateway in distributed mode.

Although it is efficient to use a single conversation, you might have instances where two conversations are necessary. Not all installations provide access to the system catalog to all users. If the DBA for the application server has not ensured that the native system catalog tables have public SELECT privileges, the **informix** user ID must be mapped to an authorization ID on the application server that has this privilege, and you must set the **GWTWOCON** environment variable.

The Gateway records the start and end of each conversation in the system-log file **$INFORMIXDIR/gw/log/gw.log**. For information about the system-log file, refer to "Errors Recorded in the gw.log File" on page 6-73.

The X/Open-Style Information Schema

The **Information Schema** option of the **gwdba** utility (page 5-41) lets you create a set of *schema definition objects* at the application server that conform to X/Open specifications. If you plan to write a client application that will use the X/Open-style information schema, you can create an X/Open catalog for the application to use.

Error Handling

The application server accepts or rejects a request from the Gateway based on the SQL syntax and data formats that the server supports. If the application server rejects the statement, the server returns information about the error to the application. The treatment of error messages from the application server is discussed in "Error Handling" on page 6-70.

Support for Multiple Character Sets

In many cases, the character set of the client application and that of the application server differ. The Gateway supports character conversion between both single-byte and multibyte character sets. Character data is converted when it is passed from the Gateway to the application server (*outbound* data) and when it is passed from the application server to the Gateway (*inbound* data). The Gateway always converts character data that the application server returns (inbound data) before it places the data into the host variables of the receiving application.

The Gateway can pass outbound character data (SQL statements that the client application sends and character data that is in host variables) unchanged to the application server. In this case, the application server handles the conversion of the outbound character data. Alternatively, the Gateway can perform character conversions on outbound data before the data is sent to the application server. Set the **GWOUTBCONV** environment variable to specify outbound conversion for single-byte CCSIDs. Refer to "The GWOUTBCONV Environment Variable" on page 4-6.

Packages and Sections

To access an application server, the Gateway requires that a *package* be available at that application server. A *package* is a control structure on the application server that contains, in effect, the compiled version of the SQL statements of an application. Use of the package is invisible to the application itself.

Each package is bound into a logical grouping of packages called a *collection*. The *collection_ID* is the name of a specific collection. For more information about collections, refer to "The Collection_ID" on page 3-13.

The process of preparing a package for use is called *binding*. You must bind two packages, one each for the isolation levels of Cursor Stability and Repeatable Read, when you configure the Gateway. These two packages support all the client applications that use the Gateway. The **gwdba** utility creates and binds the packages. The binding process is discussed in "The Bind-Package Screen" on page 5-17.

The Gateway uses the package whenever a client application references an SQL statement. A *section* is the entry in a package that represents a given SQL statement. The **gwdba** utility requires you to specify the number of sections that each package contains. "Calculating the Number of Required Sections" on page 5-25 tells how to calculate the number of sections you need.

The following sections describe how the behavior of packages and sections differs between IBM DRDA products and Informix Enterprise Gateway with DRDA.

Packages and Sections in DRDA

Packages on the application server contain the description of the static SQL statements and dynamic SQL capabilities of the application. In other words, a package is the compiled version of the SQL portion of the application. When the application is executed, the package is used whenever an SQL statement is referenced.

Each statement in the application does not necessarily correspond to a section because some statements resolve to the same section. Finding out which statements resolve into separate sections is somewhat complex, but a good general rule is that if a statement can be dynamically prepared, it resolves to a section. If a statement cannot be dynamically prepared, it shares the section of another statement. For example, the SELECT statement can be dynamically prepared, so it has its own section. However, the OPEN, FETCH, and CLOSE statements cannot be dynamically prepared, so they share a section with another statement that can be dynamically prepared (for example, the section of the SELECT statement). Dynamic statements are discussed in Chapter 6, "Programming with the Gateway."

Packages and Sections in Informix Enterprise Gateway with DRDA

Informix Enterprise Gateway with DRDA uses packages differently from IBM DRDA products. In DRDA, the package contains all the SQL statements of the application. The configuration process on the Gateway creates only *two* packages on each application server, and all the Informix applications that use the Gateway use those packages.

How can all the Informix applications that use the Gateway share two packages? The two Gateway packages contain only dynamic sections, that is, sections that are reserved for dynamic statements that are prepared later when the package executes. When an application runs, the Gateway maps the dynamic statements of the application to these dynamic sections in the Gateway packages.

The Gateway also maps *all* static SQL statements to dynamic sections. In other words, the Gateway turns the static SQL in the application into dynamic SQL. Both the application and the application server are unaware of this mapping. From the perspective of the application, some SQL statements are static and some are dynamic, but from the perspective of the application server, all the SQL statements are dynamic.

Environment Variables

The Gateway uses the Informix environment variables that are listed in alphabetical order and described in this section. For detailed information on how to set and use Informix environment variables, see the *Informix Guide to SQL: Reference*. Environment variables for Global Language Support (GLS) are found in the *Informix Guide to GLS Functionality*.

You can define all environment-variable values in a standard Informix environment-configuration file. The file must be named **informix.rc** and must be in the **$INFORMIXDIR/etc** directory of the Gateway. Values that are set in the environment or in your private environment-configuration file (**.informix** in the user's home directory) override the values that are set in **informix.rc**.

The Gateway ships with a file named **gw.rc** in the **$INFORMIXDIR/etc** directory. This file is renamed to **informix.rc** during the installation process. It contains commented-out entries for all the defined Gateway environment variables in an **informix.rc** file format.

If you do not already have an **informix.rc** file in the **$INFORMIXDIR/etc** directory, the installation process renames **gw.rc** to **informix.rc**. Otherwise, you can copy the **gw.rc** contents to your existing **informix.rc** file. Once these entries are in the **informix.rc** file, you can uncomment the Gateway environment variables you want to use and supply the appropriate values for them in the **informix.rc** file.

The Gateway process can override the values in environment-configuration files. The Gateway mode determines where you set the environment variables.

- If you access the Gateway in local-pipe mode, you must set all environment variables in the environment of the client product that you use to connect to the Gateway, before you invoke the client application.
- If you access the Gateway in network mode, before you start the daemon you must set environment variables in the environment from which you start the Gateway daemon.

The Gateway uses the following Informix environment variables:

- **DB_LOCALE**

  The Gateway uses the **DB_LOCALE** environment variable setting to determine the Informix GLS locale for code-set conversions for clients. If this environment variable is not set, the default database locale is **en_us.8859-1**. For more information, see "The DB_LOCALE Environment Variable" on page 4-5.

- **GL_PATH**

  Set the **GL_PATH** environment variable to override the default locations for GLS code-set conversion tables. For more information, see "The GL_PATH Environment Variable" on page 4-5.

- **GWASCCSID**

  The Gateway administrator should set the **GWASCCSID** environment variable to identify the application server if you encounter error -29032. For details, see *Informix Error Messages* in Answers OnLine.

- **GWBNDEXPLAIN**

  When you set the **GWBNDEXPLAIN** environment variable, this variable sets the **BNDEXPOPT** (**Bind Explain**) option during package binding. You perform package binding with the **gwdba** utility. The **BNDEXPOPT** option controls whether the target application server causes the target relational database to produce explanatory information for all explainable SQL statements in the package. An explainable SQL statement is any statement that begins with SELECT, INSERT, UPDATE or DELETE.

  Currently the option value that the Gateway sets for **BNDEXPOPT** causes the target application server to explain all explainable SQL statements. **BNDEXPOPT** is supported on DB2 for MVS and DB2 for OS/390. For information on generating the output for explainable SQL statements, see the application server product documentation. For further information on the binding process, see "The Bind-Package Screen" on page 5-17.

- **GWCATALOG**

  Use the **GWCATALOG** environment variable to control the source and amount of catalog information that the Gateway collects about objects in distributed queries. You normally do not need to set this environment variable but you might consider setting it if distributed queries take too long to complete. For more information, see "Using the GWCATALOG Environment Variable to Access Catalog Information" on page 6-62.

- **GWDBALOCALE**

  Set the **GWDBALOCALE** environment variable to allow the **gwdba** Informix catalog options to use locale-sensitive string-manipulation functions. Use it to explicitly set the Informix GLS locale of the **gwdba** utility. You can set **GWDBALOCALE** to the same values as **DB_LOCALE**. When **GWDBALOCALE** is set to a valid and available Informix GLS locale, characters that the user enters in the **gwdba** screens and in the **Add-tables** and **Purge-tables** files are processed according to the character definitions and classifications for that locale. If **GWDBALOCALE** is not set, the default locale is **en_us.8859-1**. The **gwdba** utility is described in Chapter 5. Locales are described in Chapter 4.

- **GWDEBUG**

  The **GWDEBUG** environment variable controls trace-event logging. For more information, see "The Trace File" on page 6-75.

- **GWIFXSEMANTIC**

  You can use the **GWIFXSEMANTIC** environment variable to specify Informix behavior for built-in functions that have the same name as their DB2 equivalents but different behavior. For more information, see "Specifying Informix Behavior for SQL Functions" on page 6-79.

- **GWISOLEVEL**

  In direct-access mode, if you set the **GWISOLEVEL** environment variable to a valid value, the Gateway uses the corresponding isolation level as the initial isolation level. If **GWISOLEVEL** is not set, the default is REPEATABLE READ.

  In distributed access mode, if you connect from an unlogged Informix database to OS/400 and you want to update non-journaled files on OS/400, you must use the **GWISOLEVEL** environment variable to set the isolation level to CURSOR STABILITY in conjunction with the **GWMAPCS** environment variable.

  For more information on the **GWISOLEVEL** environment variable, see "Using the GWISOLEVEL Environment Variable" on page 6-24.

- **GWKEEPDYNAMIC**

  The **GWKEEPDYNAMIC** environment variable is used only when the Gateway is connected to DB2 for OS/390 version 5.0 or higher. To use this environment variable, you need to rebind the Gateway package with the **Bind** option KEEPDYNAMIC=YES in a DB2 command. When a package is bound or rebound with **Bind** option KEEPDYNAMIC=YES, the DB2 server holds dynamic statements past commit points for an application process. For further information on the **GWKEEPDYNAMIC** environment variable, see "Using the GWKEEPDYNAMIC Environment Variable" on page 6-92.

- **GWMAPCS**

  Set the **GWMAPCS** environment variable to control access to OS/400 files that are not journaled. For more information, see "Access to Nonjournaled OS/400 Files" on page 5-29.

- **GWMAXROWS**

  Use the **GWMAXROWS** environment variable to limit the number of rows that a query can select from a remote table. For more information, see "Using the GWMAXROWS Environment Variable to Limit Rows Selected in a Query" on page 6-62.

- **GWNOBITDATA**

  The **GWNOBITDATA** environment variable is used only when the Gateway is connected to an OS/400 application server. When **GWNOBITDATA** is set to any value, the Gateway associates the AS/400 System CCSID with the OS/400 character data that is described by the OS/400 as having CCSID 65535 (0xFFFF). For more information, see "The GWNOBITDATA Environment Variable" on page 6-95.

- **GWNOPWD**

  When set, the **GWNOPWD** environment variable tells the Gateway not to send a password with the user ID that it sends to the database server, if the user ID has already been authenticated.

- **GWOUTBCONV**

  Set the **GWOUTBCONV** environment variable to specify outbound conversion for single-byte CCSIDs when the locale refers to them. For more information, see "The GWOUTBCONV Environment Variable" on page 4-6.

- **GWPROCINFOTABLE**

  Use the **GWPROCINFOTABLE** environment variable to specify the table with stored procedures information on a DB2 for OS/400 data source. For more information, see "The GWPROCINFOTABLE Environment Variable" on page 6-34.

- **GWREUSECACHE**

  In distributed mode, if the **GWREUSECACHE** environment variable is not set (the default), the Gateway issues catalog queries for any object the first time the object is accessed. When set, **GWREUSECACHE** allows Dynamic Server to reuse its cache entry for the object, if it exists. Setting **GWREUSECACHE** should significantly improve the performance of distributed queries with multiple Gateway users. For more information, see "Using GWREUSE-CACHE for the Dynamic Server Catalog Cache" on page 6-66.

- **GWTIMEOUT**

  The environment variable **GWTIMEOUT** is set to the number of minutes the Gateway process will remain idle after servicing the last request. After waiting for this specified interval the Gateway process will terminate. For more information, see "Using the GWTIMEOUT Environment Variable" on page 6-68.

- **GWTWOCON**

  If the **GWTWOCON** environment variable is set, the Gateway uses a separate conversation for catalog queries in distributed mode. If **GWTWOCON** is not set, the Gateway uses a single conversation.

- **GWUPPWD**

  Environment variable **GWUPPWD** allows users to enter passwords in either uppercase or lowercase. If **GWUPPWD** is set, then the selected password is uppercased prior to being sent to the application server. For more information, see "Using the GWUPPWD Environment Variable" on page 6-69.

- **GWUSESYSCOLNAME**

  You can use the **GWUSESYSCOLNAME** environment variable only when the Gateway is connected to OS/400 version 3.1 or higher. When **GWUSESYSCOLNAME** is set, and the Gateway is looking up catalog information about columns of an object in **qsys2.syscolumns**, the Gateway looks up the **SYS_CNAME (SYSTEM_COLUMN_NAME)** field instead of the **NAME (COLUMN_NAME)** field to find the name of the columns in that object. For more information on using this environment variable, see "The GWUSESYSCOLNAME Environment Variable" on page 6-96.

- **GWVTOT**

  When set to any value, the **GWVTOT** environment variable indicates to the Gateway that all remote objects that are views at the DRDA application servers should be represented as tables to the coordinating Dynamic Server during distributed access that involves remote views. For more information, see "Using the GWVTOT Environment Variable with Views" on page 6-58 and "Using GWVTOT with the gwdba Utility" on page 6-59.

Additional Informix environment variables that might apply, such as **DBPATH**, **INFORMIXSQLHOSTS**, **SQLEXEC**, **TERM**, and **TERMCAP** are described in the *Informix Guide to SQL: Reference*. GLS environment variables such as **DBLANG** and **CLIENT_LOCALE** are described in the *Informix Guide to GLS Functionality*.

Security Features

The user ID and password of the UNIX application user probably differ from the user ID and password that you need to access the database on the application server. You can use the following mechanisms to establish a map of user-ID and password pairs between UNIX and the application server:

- The **gwdba** utility
- The **.netrc** file
- The USER clause of the CONNECT statement

The **gwdba** utility maps a UNIX user ID to a user ID and password pair for an application server. The system administrator of the Gateway can invoke the **gwdba** utility to view or update the user ID maps. If any other users invoke the **gwdba** utility, the users can view and update only their own maps. In either case, passwords can only be updated, never viewed.

Passwords are protected by an encryption algorithm. The Gateway uses a different key for each installation of the Gateway. This key assures that if the file that contains the encrypted passwords is stolen, the file cannot be used at another Gateway installation.

Issues of security are discussed in more detail in "Enforcing Security" on page 5-12.

Performance

The performance of the Gateway is, of course, affected by the same considerations that affect any Informix database server. The following list shows some of these considerations:

- Structure of the database
- Efficiency of the programming of the client application
- Speed of the computer and the storage media
- Type of logging

In addition, the following factors can affect the performance of the Gateway:

- Speed of the network

  For a discussion of transfer rates, refer to "Effect of Network Speed on Performance" on page 3-38.
- Type of system catalog available at the application server

  For a discussion of catalogs that the Gateway uses, refer to "Catalog Information" on page 1-10.
- Programming considerations

  Refer to direct access "General Performance Considerations" on page 6-42 and distributed access "Performance Considerations" on page 6-61.
- Correctness of table statistics

  Refer to "Keeping Statistical Information Current" on page 3-40.

The Gateway administrator and designers of client applications need to work together to maximize the performance of the applications.

Chapter

2

Installation

In This Chapter

This chapter discusses the installation of Informix Enterprise Gateway with DRDA. It describes the files that are used in installation and includes the installation instructions.

Preparing for Installation

To simplify the installation process, make sure that you have all the necessary items and information before you begin.

Installation Items

Your Informix product materials include the following items:

- A serial number keycard that gives numbers that are required by the installation procedure
- Electronic media that contain all the product files

The electronic media that Informix provides for the installation of the Gateway include these files:

- Installation script: **installgw**
- The Gateway executable files: **gw**, **gwd**, **gwdba**
- Files that are required to support the Gateway, such as the character conversion tables
- Release notes

Hardware and Software Requirements for the Gateway Computer

The Gateway requires the hardware and software that the following sections describe.

SNA LU 6.2 Network Protocol

Appropriate LU 6.2 support is installed on the Gateway computer by the system administrator and configured by the network administrator. You must install the Gateway software on a computer that can support an SNA LU type 6.2 and Physical Unit (PU) type 2.1. This type of connection is also referred to as *advanced program-to-program communications* (APPC).

In SNA terminology, the computer that runs the Gateway must be configured as a *PU type 2.1 node* to support the APPC (LU 6.2) connection. In addition, a *connection file* (or an equivalent object, such as the *p2pconf input file* on a Sun workstation) must be completed for each connection to a remote application server. The connection file includes the necessary LU 6.2 information, such as the NetID, LU name, TP name of the partner; the unique session name; and the LU name of the Gateway. (The SNA term *partner* refers, in this case, to the application server.)

The physical communications link between the Gateway computer and the application server can be any connection that supports SNA, such as *Synchronous Data Link Control* (SDLC) or Token Ring.

TCP/IP Network Protocol

If the application server supports TCP/IP communications, the Gateway computer system and the application server computer system should have TCP/IP network connectivity established to communicate to each other.

Software Required at the Application Server

The Gateway supports various application servers such as DB2, SQL/DS, and OS/400. See your on-line release notes for the latest versions.

You do not need to install any other software on the host computer of the application server. You do need to ask the database administrator for a login for each user who can access the remote application server. For more information about user logins and the permissions they require, refer to "Information About the Application Server" on page 3-12.

Administrators Who Can Provide Help

Working with a relational database system in an environment that might involve multiple computers, multiple operating systems, and multiple network-connection types demands patience and cooperation among the managers of the different systems. As you install and use the Gateway, you need to talk to the following people:

- The UNIX system administrator

  The system administrator deals with questions of resource allocation and can assist with the initial installation of the Gateway software.

- The UNIX network administrator

  The network administrator takes care of the **/etc/hosts**, **/etc/services**, and **/etc/hosts.equiv** files, which are required for TCP/IP network connections, and the NetWare file, which is required for IPX/SPX network connections. The location of the NetWare file is vendor-dependent. The network administrator maintains the APPC software and deals with network failures.

- The SNA communications administrator (for LU.6 connections)

  The SNA administrator prepares and maintains the application-server part of the network connections between the Gateway computer and the application server.

- The security administrator for the computer where the application server resides.

  The security administrator for the application server updates tables to grant authorization for Gateway users to log on to the relational database on the application server.

- The database administrator of the application server

  The database administrator (DBA) of the application server provides an *RDB_user_ID* for each user of the application server. The administrator also grants specific database privileges for individual users.

Installation Directory

You must decide which directory to use for the Gateway files. The directory where you install the Gateway (or any other Informix product) is referred to as your **$INFORMIXDIR**. Informix suggests that you install your Informix products in a subdirectory (or subdirectories) of **/usr**, such as **/usr/informix**.

Important: *You must not install Informix Enterprise Gateway with DRDA in an NFS- or RFS-mounted directory. You must install the Gateway on a disk drive that is local to the Gateway computer.*

If you plan to use unnamed pipes to make local connections, you must install the Gateway in the same **$INFORMIXDIR** as the Informix client products that will use the Gateway. If you plan to use network connections exclusively, you can install the Gateway in a directory by itself.

Upgrading from a Previous Release of the Gateway

If you are upgrading from a release prior to Gateway Version 7.2, the package on each target application server must be rebound before you start using the Version 7.3x Gateway. For information on package binding, see "The Bind-Package Screen" on page 5-17.

If you already installed the Informix catalog on the application server by using a Gateway version prior to Version 7.31, the DB2 database administrator must update the catalog. Informix provides a script called **upgrade.sql** in the $I**NFORMIXDIR/gw/catalog/install** directory. To perform the update, the administrator can run the **gwdba Catalog-Install** option and choose the **upgrade** file. Alternatively, the administrator can use a tool like DB-Access to run the **upgrade.sql** script.

Changes to the $INFORMIXDIR Directory

The installation script changes the directory structure of the **$INFORMIXDIR** directory when you upgrade to Version 7.3x. It removes the **gw/ccsidtab** directory and adds the **gls/cm3**, **gls/cv9** and **gls/lc11** directories, as Figure 2-1 shows.

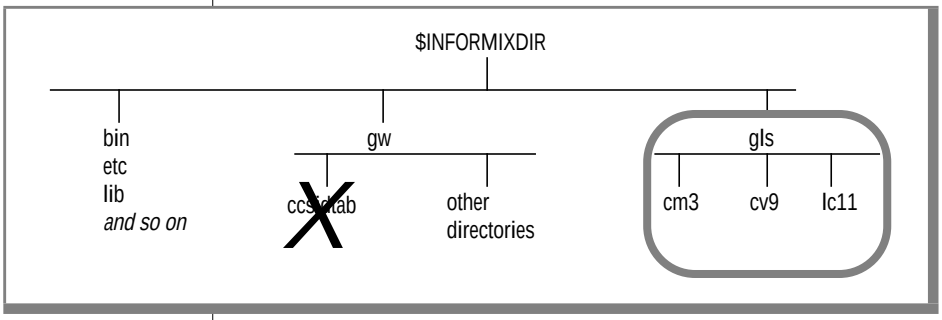


Figure 2-1
Directory Changes That the Installation Script for Version 7.3x Makes

Absence of gwlang Files

Because the Version 7.3x **gwdba** utility does not have a locale option, **gwlang** files are not created.

Upgrade from Gateway Version 7.2

If you are upgrading from Gateway Version 7.2, you must first use the Gateway Version 7.2 **gwdba** utility to unload existing package information, install Gateway Version 7.3x, and then load the new package with the Version 7.3x **gwdba** utility. If you follow this procedure, you do not need to rebind packages with Gateway Version 7.3x.

Installing Informix Enterprise Gateway with DRDA When Other Informix Products Are Present

If you are installing several Informix products, install them in the following order:

1. Application development tools, such as INFORMIX-4GL
2. SQL application-programming interfaces (APIs), such as ESQL/C or ESQL/COBOL
3. Database server products, such as Informix Dynamic Server and INFORMIX-OnLine
4. Networking products, such as INFORMIX-STAR or INFORMIX-NET
5. Gateway products, such as Informix Enterprise Gateway with DRDA, INFORMIX-Enterprise Gateway for EDA/SQL, and INFORMIX-Enterprise Gateway Manager

Installing Informix Enterprise Gateway with DRDA

Follow these steps to install Informix Enterprise Gateway with DRDA:

1. Copy the Gateway product onto your computer.
2. Update error-message files on each computer where client applications are installed.
3. Print out the release notes, documentation notes, and machine notes from the **$INFORMIXDIR/release** directory.

Copying the Gateway onto Your Computer

Your Informix product materials include a serial-number keycard and electronic media that contain all product files. Both are necessary for installation. If you do not have the serial-number keycard or the proper media, contact your supplier or Informix sales representative.

Before you load the Gateway, you must set the **PATH** and **INFORMIXDIR** environment variable. For convenience, add the following environment variable commands to your **.profile** (Bourne shell) or **.login** (C shell) file.

Bourne shell or Korn shell:

```
INFORMIXDIR=/usr/informix
export INFORMIXDIR
PATH=$INFORMIXDIR/bin:$PATH
export PATH
```

C shell:

```
setenv INFORMIXDIR /usr/informix
setenv PATH ${INFORMIXDIR}/bin:${PATH}
```

Loading Product Source Files

1. Load the media supplied with your software into the appropriate tape drive, disk drive, or other device in your computer.
2. If you are not currently in the **$INFORMIXDIR** directory, enter the following command:

   ```
   cd $INFORMIXDIR
   ```

3. To transfer the software from the media to the current directory, enter the appropriate `cpio`, `tar`, or other loading command listed on the serial-number keycard. In most cases, it lists a version of the `tar` or `cpio` command similar to one of the following forms:

   ```
   cpio -ivdBum < devicename
   ```

   or

   ```
   tar xvf[b] 20 devicename
   ```

   The *devicename* refers to the full pathname to that device. Devices are commonly in **/dev**, so the name is normally **/dev/*devicename***.

If diskettes are supplied, you might need to repeat the system command for each diskettes, or you might be prompted to insert each new diskette and press RETURN.

Important: *At this point, Informix recommends that you read the appropriate machine notes for the Gateway.*

Installing the Gateway

1. Enter the following command as super user and press RETURN:

   ```
   installgw
   ```

2. The following message appears on the screen:

   ```
   Installation Script

   This installation procedure must be run by root
   (super-user). It will change the owner, group, and
   mode of all files of this package in this directory.
   There must be a user "informix" and a group "informix"
   known to the system.

   Press RETURN to continue,
   or the interrupt key (usually CTRL-C or DEL) to abort.
   ```

   Press RETURN to continue the installation procedure.

3. The following prompt appears:

   ```
   Enter your serial number (for example, INF#X999999) >
   ```

   Enter the 11-character serial number, located on your serial-number keycard, and press RETURN. The serial number is three uppercase letters, followed by a pound sign (#), followed by one uppercase letter and six digits.

   If you need to erase and change any part of the serial number, you can only do so if you set your erase key to ^H.

4. The following prompt appears:

   ```
   Enter your serial number KEY
   (uppercase letters only) >
   ```

   Enter the six-letter software serial-number key, which is also located on the serial-number keycard, and press RETURN.

   A message appears similar to the following example:

   ```
   WARNING!
            This software, and its authorized use and
   number of users, are subject to the applicable license
   agreement with Informix Software, Inc. If the number
   of users exceeds the licensed number, the excess users
   may be prevented from using the software. UNAUTHORIZED
   USE OR COPYING MAY SUBJECT YOU AND YOUR COMPANY TO
   SEVERE CIVIL AND CRIMINAL LIABILITIES.
   ```

   If your software is licensed for use by an unlimited number of simultaneous users, you see a message to that effect.

   Press RETURN to continue the installation procedure.

5. At this point, the installation procedure begins. A series of messages appear on the screen as each directory is installed. The messages are similar to the following example:

```
Installing directory .
Installing directory bin
Installing directory lib
          .
          .
          .
```

The following message tells you that your product is fully installed.

```
Installation of product complete
```

In this example, *product* is the Informix product that you are installing. After the preceding message appears, the shell prompt appears. This prompt indicates that the installation procedure is finished.

If no error messages appear during the installation procedure, the installation is successful. If any error messages appear before the `Installation complete` message, see "Solving Installation Problems" on page 2-13.

Updating the Error-Message Files

In addition to the error messages that serve all Informix applications, the installation materials include new error messages that are specific to the Gateway. After you run the **installgw** installation script, if your client applications do not use the same **$INFORMIXDIR** as the Gateway, you must overwrite the old message files with the new files, as the following sections describe. The new message files contain both the old messages and the new messages.

If your client applications use the same **$INFORMIXDIR** as the Gateway, you do not need to copy any message files.

Important: *If you run Informix client applications from more than one computer, you must copy the message files to each computer where a client application resides. You must copy the message files as binary files if you use FTP.*

Files for Version 5.x Client Applications

If your client applications are on a different computer or do not use the same **$INFORMIXDIR** as the Gateway, copy the **net.iem** and **sql.iem** files from the **$INFORMIXDIR/msg/en_us/0333** directory of the Gateway into the **$INFORMIXDIR/msg** directories of *each* client application.

You need to copy the **net.iem** and **sql.iem** files only if the sizes of these files in the client installation are smaller than the same files in the Gateway installation.

Files for Version 6.x and Version 7.1UC1 Client Applications

If your client applications are on a different computer or do not use the same **$INFORMIXDIR** as the Gateway, copy the **drda.iem** file from the **$INFORMIXDIR/msg/en_us/0333** directory of the Gateway into the **$INFORMIXDIR/msg** directories of *each* client application.

You need to copy the **drda.iem** file only if the size of this file in the client installation is smaller than the same file in the Gateway installation.

Files for Version 7.1UD1, Version 7.2, 9.x, and Client SDK Client Applications

If your client applications are on a different computer or do not use the same **$INFORMIXDIR** as the Gateway, copy the **net.iem** file from the **$INFORMIXDIR/msg/en_us/0333** directory of the Gateway into the **$INFORMIXDIR/msg** (or **$INFORMIXDIR/msg/en_us/0333** for Version 7.2, Version 9.x, and Client SDK clients) directories of *each* client application.

You need to copy the **net.iem** file only if the size of this file in the client installation is smaller than the same file in the Gateway installation.

In addition, if the file **cnet.iem** exists in the client installation and the size of **cnet.iem** is smaller than the file **net.iem** in the Gateway installation, you need to copy **net.iem** over **cnet.iem**.

Solving Installation Problems

This section describes the more common installation problems and how to solve them. If any of the outlined problems persist, contact the Informix Technical Support Department. In North America, call toll-free (800) 274-8184 or send a FAX to (913) 599-8590. Outside North America, contact your distributor or the nearest Informix subsidiary.

Media-Loading Failures

The problems in this category refer to difficulties in loading the product files onto your computer from the media supplied by Informix:

- ***Problem.*** You attempt to load the files, but the `cpio`, `tar`, or other loading command fails with an error message similar to one of the following examples:

  ```
  invalid blocksize
  cannot open devicename
  unknown option
  tape read error
  ```

  Solution. The load command is most likely to fail because the wrong command arguments were entered or because the media is damaged. Check the serial-number keycard and verify that you are entering the `cpio`, `tar`, or other loading command exactly as written. Try the command again. If it continues to fail, contact the Informix Technical Support Department or the vendor from whom you purchased the product. You might need new media.

Product-Installation Failures

The problems in this category refer to difficulties that you might encounter while you are running the installation script.

- ***Problem.*** When you attempt installation, you see the following message:

  ```
  Please rerun this installation procedure as
  super-user
  ```

  Solution. Check that you are logged in as **root**.

- ***Problem.*** When you attempt installation, you see the following message:

  ```
  INFORMIXDIR is not set.
  ```

  Solution. There is no default **INFORMIXDIR** for installation. You must set the variable to the directory where the product is to be installed.

- ***Problem.*** When you attempt installation, you see the following message:

  ```
  INFORMIXDIR and working directory do not match.
  INFORMIXDIR is set to pathname
  Current working directory is pathname
  ```

  Solution. The user must be in the directory that corresponds to **INFORMIXDIR** to run the installation procedure. One way to ensure this correspondence is to first change directories to where the product is to be installed, and then set **INFORMIXDIR** by typing the following command:

  ```
  setenv INFORMIXDIR 'pwd'
  ```

- ***Problem.*** After you enter the six-letter software serial-number key, messages such as the following examples appear:

  ```
  chmod:  can't change filename
  etc/brand:  cannot open filename
  filename:  not owner
  ```

  Solution. This problem usually occurs because you are not logged in as **root**. Log out, and log back in as **root**. You must rerun the installation script and repeat all subsequent steps. This problem can also occur when you attempt installation on a cross-mounted file system. If so, log in as **root** on the computer where the cross-mounted file system resides.

- ***Problem.*** After you enter the six-letter software serial number key, the following message appears as different directories are installed:

  ```
  "WARNING: This is an invalid serial number.
  Exiting install script."
  ```

  Solution. This message is caused when you enter an incorrect serial number. Make sure you enter the correct serial number.

- ***Problem.*** After you enter the six-letter software serial-number key, the following message appears as different directories are installed:

```
etc/brand: invalid serial number and/or key.
 ** Verify serial number and key values. **
 ** Restart installation procedure.      **
 ** Please type carefully.                **
```

  Solution. This message indicates that the product cannot be installed because the serial numbers are not valid. In this case, it is likely that you did not enter either the 11-character serial number or the 6-letter serial number key correctly.

  If you did enter the serial numbers correctly, the error can occur because the `stty erase ^h` command was not run properly, causing certain keystrokes to be misinterpreted. For example, a pound sign might be read as a backspace. If you suspect this problem, rerun the **stty** command before you continue.

  You must rerun the installation script and repeat all subsequent steps. The installation might still fail. If so, reload the product files from the media and repeat all subsequent steps.

- ***Problem.*** After you enter the six-letter software serial number key, the following message appears as different directories are installed:

```
Unknown message number 32766.
```

  Solution. This problem occurs when **INFORMIXDIR** is not set before you run the installation script. You must set **INFORMIXDIR**, rerun the installation script, and repeat all subsequent steps.

Inability to Access Gateway Executables

The problems in this category refer to difficulties in accessing Informix products:

- ***Problem.*** You try to run a Gateway executable such as **gwdba** from the command line or an alternative method, and you receive a system prompt or a message similar to the following message:

  ```
  program:  Command not found.
  ```

  Solution. Such a response indicates that the executable file could not be found. This problem normally involves environment variables. Most likely, either **INFORMIXDIR** is not set or **PATH** is set incorrectly. Check the user's environment to verify that the environment variables are set properly. If you are running the Bourne shell, remember to export **INFORMIXDIR** and **PATH** after setting them.

  If the **INFORMIXDIR** and **PATH** environment variables are set correctly, check whether another file with the same name in the user's path would be accessed before the Informix executable file. If such a file exists, you must move, rename, or delete it. Alternatively, you can reorder the user's path to find the Informix executable file before the other file.

  This error can also occur if a user's **TERM**, **TERMCAP** (or **TERMINFO**), or **INFORMIXTERM** environment variable is set incorrectly. The listed **TERM** terminal type must be a valid entry in the **termcap** file (or the **terminfo** directory) that the user accesses. If you use **TERMINFO**, you must also set **INFORMIXTERM** to **terminfo**. (For more information, see the *Informix Guide to SQL: Reference*.)

  Check the user's environment to verify that **TERM**, **TERMCAP** (or **TERMINFO**), and **INFORMIXTERM** are set correctly. Reset them if necessary. You can check for environment difficulties by calling a system editor, for example **vi**. A distorted display indicates that these environment variables are not set correctly for your terminal.

- ***Problem.*** You try to call the Informix program from the command line (or through an alternative method), but you get the following (or a similar) message:

  ```
  Unknown message number 32766.
  ```

  Solution. This problem generally occurs when **INFORMIXDIR** does not point to the correct directory or the directory is misspelled. Check what **INFORMIXDIR** is set to, and reset it to the correct directory. (This problem also can occur when you use the **DBLANG** or **CLIENT_LOCALE** environment variables, if either is not set correctly. As with **INFORMIXDIR**, check your **DBLANG** or **CLIENT_LOCALE** environment variable, and reset it if necessary.)

- ***Problem.*** You try to call the Informix program from the command line (or through an alternative method), but you get the following (or a similar) message:

  ```
  Invalid serial number.  Please consult your
  installation instructions.
  ```

  Solution. This message generally indicates that the product is not installed. Either the installation script was not run, or it failed. You must rerun the installation script and repeat all subsequent steps.

  Another possibility is that the installed product files were copied into another directory that is in the **PATH** before **$INFORMIXDIR/bin**. Try placing **$INFORMIXDIR/bin** first in the **PATH** to determine if this is the problem.

Summary of Files

Figures 2-2 through 2-5 list the files necessary for the Gateway process.

Figure 2-2
Files and Utilities That Are Supplied by Informix

Name	Purpose	References
installgw	Used during installation	page 2-9
gwdba utility	Creates and maintains the Gateway system-control files	page 5-9
gw executable	Gateway executable file	page 3-24
gwd executable	Gateway-daemon executable file	page 3-24
gw.rc	Gateway-specific environment variable configuration file that lists all Gateway environment variables as commented-out Informix environment configuration file entries	page 1-16
libsnastub.a	SNA stub library. Might be required if TCP/IP is used for DRDA.	page 3-28
termcap	Provides terminal capability information that the **gwdba** utility uses	*Informix Guide to SQL: Reference*
GATEWAYDOC_7.31 **GATEWAYREL_7.31** **GATEWAY_7.31**	Provides supplemental information in the form of on-line documentation and release notes, and machine notes, respectively	page 14
CDRA conversion files	Provides information for character translation	*Informix Guide to GLS Functionality*
Message files (**drda.iem**, **net.iem**, and so on)	Provides information about error messages	page 2-11

Figure 2-3
Files That You Create or Manage

Name	Purpose	References
gwuser	Keeps information about user IDs	page 5-7
gwbind	Keeps information about packages	page 5-17
gwaplpkg	Keeps information about application-to-package mapping	page 5-22
gwenv.sh **gwenv.csh**	Environment variable configuration files used for the **gwdba** utility and the Gateway daemon **gwd**	page 3-24, page 5-3
sqlhosts	Describes connection characteristics	page 3-13
informix.rc	Environment variable configuration file	*Informix Guide to SQL: Reference*

Figure 2-4
Files that the Gateway Creates

Name	Purpose	References
The Gateway daemon log	Keeps information about Gateway invocations	page 3-25
gw.log	Gateway system log file	page 6-73
gwt.*euidpid*	Trace-event log file	page 6-75
prnccsid.dat	Cache file for conversion information	page 4-7

Figure 2-5
Network and Security Files

Name	Purpose	References
LU 6.2 configuration files	Provides vendor-specific information for the LU 6.2 connections	Vendor documentation
IPX/SPX network files	Provides vendor-specific information for IPX/SPX connections	Vendor documentation
/etc/hosts **/etc/services**	Provides connection information for UNIX TCP/IP network connections	UNIX on-line manual, page 3-9
/etc/hosts.equiv **~/.rhosts**	Provides security information for network connections	UNIX on-line manual, page 5-14
~/.netrc	Provides network connection information for an individual user	UNIX on-line manual, page 5-16

Chapter

3

Configuration

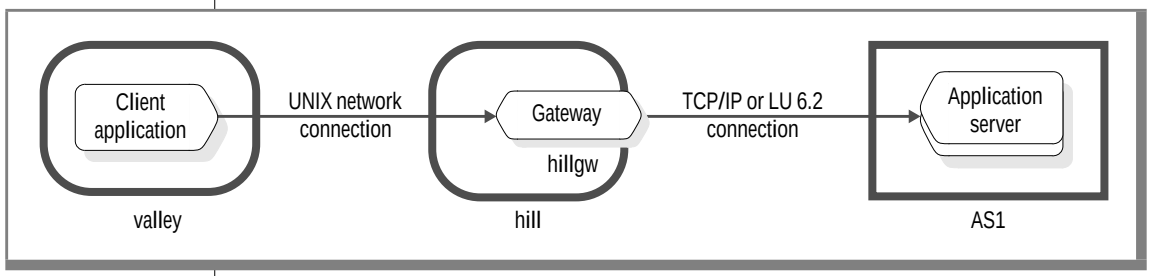
In This Chapter

This chapter covers information that you need to configure the Informix Enterprise Gateway with DRDA. It also provides information for the Gateway administrator on how to obtain the best possible performance from the Gateway.

Gateway Configuration

Figure 3-1 shows a simple network configuration of a client application, Informix Enterprise Gateway with DRDA, and an application server. The client application and the Gateway reside on different computers that are connected over a UNIX network. The computers on which the application server and the Gateway reside are connected through SNA LU 6.2 or TCP/IP connections.

Figure 3-1
A Typical Configuration of a Client, a Gateway, and an Application Server



You can also connect an Informix Dynamic Server to the Gateway and use Dynamic Server as the coordinator for distributed transactions, as Figure 3-2 illustrates.

Figure 3-2
Gateway Configuration for Distributed Transactions

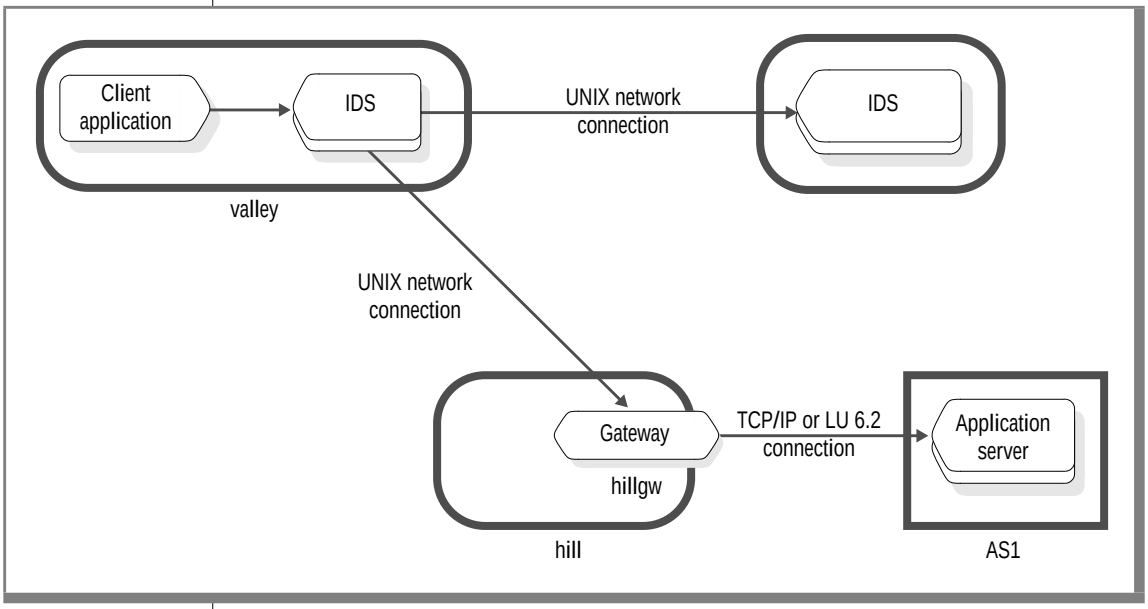


Figure 3-3 lists the tasks that you must complete to configure the Gateway. The tasks that directly relate to making connections between the client, the Gateway, and the application server are discussed in this chapter.

The remaining tasks are discussed in Chapter 4, "Global Language Support," Chapter 5, "The gwdba Utility," and the product-specific manuals for the network software and hardware.

Figure 3-3
Steps for Preparing the Gateway

Step	Task	Refer To
1	Prepare the network connection	Product-specific manuals
2	Gather information	page 3-8
3	Prepare the **$INFORMIXDIR/etc/sqlhosts** file	page 3-13
4	Prepare **gwenv.sh** or **gwenv.csh** file	page 3-24
	Start the Gateway daemon (**gwd**)	page 3-24
5	Use the **gwdba** utility for the following tasks:	
	Enter user-ID maps	page 5-8
	Test the connection to an application server (*optional*)	page 5-42
	Perform package binds	page 5-17
	Install Informix catalog (*optional*)	page 5-30
	Install X/Open-style information schema (*optional*)	page 5-40
6	Prepare for direct or distributed connection	page 3-29
7	Modify the system-startup script to bring up the Gateway daemon when the system is booted (*optional*)	page 3-34

Step 1: Prepare the Network Connection

You must prepare the configuration for the LU 6.2 or TCP/IP by following the instructions of the supplier of the SNA or TCP/IP product. Also, to make troubleshooting easier, the network administrator should verify that the connection between your computer and the target computer functions properly. The tools that the manufacturer provides with the APPC or TCP/IP software usually include tests to verify the connection.

This manual assumes that your system has a functioning TCP/IP or LU 6.2 connection to the application server. It also assumes that you are familiar with UNIX networking tools or that assistance is readily available.

Step 2: Gather Information

Before you start the configuration process for the Gateway, you need information about:

- The UNIX network
- For SNA connections, the APPC network
- For TCP/IP connections: TCP/IP network information between the Gateway computer and application server
- The application server
- The locale

Information About the UNIX Network

You might need the help of the network administrator to find information about the UNIX network. You use this information when you prepare the **sqlhosts** file.

The Communication Protocol

What communication protocol will the client application use to connect to the Gateway? The following connections are possible:

- A local connection with unnamed pipes
- A TCP/IP network connection
- An IPX/SPX (NetWare) network connection

You can use a local connection only if the client application and the Gateway are on the same computer.

The Communications Interface

What communications interfaces do the client and the Gateway use? The following interfaces are possible:

- Interprocess communication (IPC) with local connections
- Sockets
- Transport-level interface (TLI)

The hostname

The name by which the Gateway computer is known to your UNIX network is the **hostname** value. For example, in Figure 3-1 on page 3-5, the host name of the Gateway computer is **hill**. To find the host name, follow these steps:

- For a TCP/IP network connection, the host name is the name that is associated with the network address of the Gateway computer in the **/etc/hosts** file.
- For a NetWare network connection, the host name is the name of the NetWare file server.
- When you use a local connection (unnamed pipes) instead of a network connection, you use the local hostname of the computer.

Both the TCP/IP host-name entry in the **/etc/hosts** file and the NetWare file-server name are customarily the same as the host name of the computer, but that is not a requirement. You can learn the host name of the computer by entering the command `hostname` at the UNIX prompt.

The TCP/IP servicename Value

If the Gateway is on a TCP/IP network, the **servicename** value is an entry in the **/etc/services** file. Ask the network administrator which service name is to be associated with the Gateway daemon.

Information About the Advanced Program-to-Program Communication

For information about the APPC, ask the person who administers the APPC connection on the computer where the Gateway is installed. The UNIX system administrator probably also administers the APPC.

Much of the information is platform specific, so you need to refer to the machine notes and to the information supplied by the vendor of the APPC. The following sections discuss some of the information that you might need.

The modename Value

When you configure the APPC connection, one of the parameters you must set is the **modename** value. The definition of the mode name varies among vendors. You must refer to the machine notes for information about the correct **modename** value for your configuration. You use the mode name when you prepare the **$INFORMIXDIR/etc/sqlhosts** file.

You can configure the APPC connection to run in different modes for different purposes. For example, you might use one mode for applications that do batch processing and another mode for interactive applications. You need to know what the characteristics of the modes are and the mode name of each mode.

The sna_gateway_name

The *sna_gateway_name* is defined by some vendor software. You might need the *sna_gateway_name* when you prepare the **sqlhosts** file. Refer to the vendor documentation and to the machine notes to determine if you need the *sna_gateway_name*.

The Buffer Size

The *buffer size* is an optional parameter in the **sqlhosts** file that specifies the size of the communication buffer that the Gateway uses to communicate with the application server. The default value is 16 kilobytes. The Gateway can use buffer sizes that range from 512 bytes to 32,767 bytes. Informix recommends that, for performance reasons, the buffer be no smaller than 4096 bytes (4 kilobytes). Changing the buffer size can drastically change the performance of the Gateway.

Check the machine notes to see if a recommended buffer size exists for your configuration. See "On-Line Documentation" on page 13 of the Introduction.

The TPN

The *transaction program name* (TPN) is a parameter in the **sqlhosts** file.

For DB2 and OS/400 application servers, the TPN is an optional parameter in the **sqlhosts** file and is used as a service aid by Informix. The Gateway uses a default TPN value of 0x07F6C4C2.

Important: *For SQL/DS application servers, you must set the TPN to the RDB name of the SQL/DS application server, for example,* `t=SQLDS1`.

The length of the remote TPN that the **sqlhosts** file specifies using the `t=` parameter is limited to 36 characters. The Gateway recognizes only the first 36 characters in the specified name.

TCP/IP Information

If the communication between the Gateway computer and application server computer is configured to use TCP/IP network protocol, obtain the following information to service remote DRDA requests.

- The host name or IP address of the application server computer
- The service name or the port number used by the application server

Information About the Application Server

Obtain the information that the following sections describe from the database administrator of each application server that the Gateway will access.

Important: *Inform the administrator of the application server if the Gateway issues a GRANT EXECUTE TO PUBLIC statement on the packages it binds when that option is selected in the Bind-Package screen of the* ***gwdba*** *utility.*

The real_RDB_name

The *RDB_name* is the database server name, the globally unique name for an application server. In this manual, the *RDB_name* is called the *real_RDB_name*.

You need the *real_RDB_name* when you use the **gwdba** utility, as Chapter 5, "The gwdba Utility," describes, and when you prepare the **sqlhosts** file, as page 3-13 describes.

The RDB_user_ID for the Gateway Administrator

The administrator of the application server must create or provide an existing user ID for the Gateway administrator. This manual refers to the application server user IDs as the *RDB_user_IDs*. The *RDB_user_ID* that the Gateway administrator (UNIX user ID **informix**) uses must have the privileges to bind packages in a collection.

RDB_user_IDs for Users of Client Applications

To access an application server, the user must have a user ID that the application server recognizes. The user IDs that the UNIX network recognizes might not be the same as the user IDs that the application server recognizes. The administrator of the application server must create or provide an existing *RDB_user_ID* for each user who will access the application server with the Gateway.

The Collection_ID

On a DB2 application server, a *collection* is a logical grouping of bound packages. On an OS/400 application server, a collection specifies a group of tables and other database objects. The *collection_ID* is the name of a specific collection.

The system administrator of the application server must choose or create a collection to contain the packages that the Gateway uses and give you its *collection_ID*. On an SQL/DS application server, the *collection_ID* is the *RDB_user_ID* of the Gateway administrator (UNIX user ID **informix**).

You use the *collection_ID* when you bind packages with the **gwdba** utility, as described in "The Bind-Package Screen" on page 5-17.

Information About the Locale

The Gateway obtains locale information from several environment variables at the client and Gateway computers. For clients prior to Version 7.2, the Gateway administrator can specify a **DB_LOCALE** setting in the Gateway environment. If you are using U.S. English, you can probably leave the locale information set to the defaults. For information about locales, see the *Informix Guide to GLS Functionality*.

Step 3: Prepare the sqlhosts File

The **sqlhosts** file resides in the **$INFORMIXDIR/etc** directory. It contains information about local and network connections. A client application uses the information to connect to a database server. Database servers use the information to connect to other database servers in distributed transactions. For Informix Enterprise Gateway with DRDA, the **sqlhosts** file also includes information that the Gateway requires to establish a connection to the application server.

You can set the **INFORMIXSQLHOSTS** environment variable to specify the full pathname and filename of a file that contains connectivity information. When **INFORMIXSQLHOSTS** is set, the client or database server looks in the specified file for connectivity information. When **INFORMIXSQLHOSTS** is not set, the client or database server looks in the **$INFORMIXDIR/etc/sqlhosts** file. The file that the **INFORMIXSQLHOSTS** environment variable specifies has the same format as the **$INFORMIXDIR/etc/sqlhosts** file.

Each set of Informix products that is installed within a particular **$INFORMIXDIR** directory (for example, a Version 5.x or a Version 7.x installation) has only one **sqlhosts** file. That one file contains connection information for all of the Informix products in that **$INFORMIXDIR**. This manual only discusses connections that involve the Gateway. Other connections are discussed in the manuals of specific products, such as your *Administrator's Guide*.

To prepare a Gateway for use, you must put entries in the **sqlhosts** of both the Gateway and the client. These **sqlhosts** can be the same file or different files, depending on whether the client and the Gateway products are installed in the same or different **$INFORMIXDIR** directories.

Syntax of the sqlhosts File

Figure 3-4 shows a sample **sqlhosts** file. Each entry in the **sqlhosts** file is one line composed of five fields: the **dbservername** field, the **nettype** field, the **hostname** field, the **servicename** field, and the **options** field.

Figure 3-4
Example of an sqlhosts File

dbservername	nettype	hostname	servicename	options
valley_gw	ontlitcp	valley	valley_service	
river_gw	onipcpip	river	gw	
payroll	drapplu6	BIGBLUE	m=mode1,g=sunTR	
inventory	drtlitcp	db2host	db2port	RDB=BIGBLUE

Tip: *The descriptive field names were chosen to describe the* ***sqlhosts*** *values that Informix Dynamic Server uses. For some Gateway entries, the field names do not relate to the values in the field.*

You can separate the fields with white spaces. The following list describes the syntax rules for the five fields:

1. The first field, **dbservername**, must meet the following requirements:
   - It must be unique on your UNIX network.
   - It must have 18 or fewer characters.
   - The first character must be a letter.
   - It can contain only lowercase letters, numbers, and underscores.
2. The second field, **nettype**, is always 8 lowercase letters.
3. The third field, **hostname**, must meet the following requirements:
   - It cannot be longer than 64 characters.
   - It cannot contain white space or punctuation marks.
4. The fourth field, **servicename**, must meet the following requirements:
   - It cannot be longer than 128 characters.
   - It cannot contain white space.
5. The fifth field, **options**, must meet the following requirements:
   - It must be in the format:

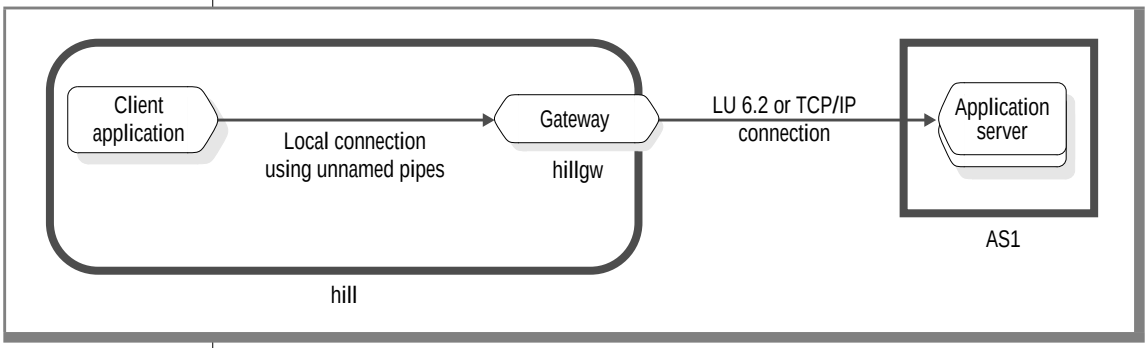
     letter=value or keyword=value
   - Combined options can be separated by commas or white space.

See your *Administrator's Guide* for the complete syntax of the **sqlhosts** file.

Preparing the sqlhosts File on the Gateway Computer

Figure 3-1 on page 3-5 shows a Gateway configuration where the client application and the Gateway are on different computers. In Figure 3-5, the client application and the Gateway are on the same computer. The italic text in the figures shows the names of the computers, the Gateway, and the application server. The examples in the following sections use the names that Figure 3-1 and Figure 3-5 show.

Figure 3-5
A Gateway Configuration That Uses Unnamed Pipes



On the computer where the Gateway resides (**hill**), the **sqlhosts** file must have the following entries:

- One or more entries that allow clients to connect to the Gateway
- An entry for each application server to which the Gateway can connect

Local Connection

When the client application and the Gateway are on the same computer, you can use a local connection with unnamed pipes. To find the correct values for the **sqlhosts** file entry that allows a local connection like that in Figure 3-5, use the following criteria:

1. For the first field, choose a name for the Gateway database server. This name is the Gateway server name, *gwservername*.

   As discussed in "What Is Informix Enterprise Gateway with DRDA?" on page 1-4, the Gateway looks like a database server to an Informix client. The *gwservername* is analogous to the database server name or *dbservername* of a Dynamic Server.

   In Figure 3-5, the name of the Gateway (*gwservername*) is **hillgw**.
2. The second field is the **nettype**. For an unnamed-pipes connection, the **nettype** value must be **onipcpip**.
3. The third field is the **hostname**. The host name of the computer in our example is **hill**.
4. The value in the fourth field, the **servicename**, is always **gw** for a local connection using unnamed pipes.

The entry for the **sqlhosts** file is shown in the following example:

```
hillgw      onipcpip     hill      gw
```

Network Connection

When the client application and the Gateway are on different computers, as in Figure 3-1 on page 3-5, you must use a network connection. You can also use a network connection when the client application and the Gateway are on the same computer.

To find the correct values for the entry in the **sqlhosts** file on the Gateway, which allows a network connection between the application server and the Gateway as Figure 3-1 on page 3-5 shows, use the following criteria:

1. For the first field, choose a name for the Gateway database server. This name is the Gateway server name, *gwservername*.

   As discussed in "What Is Informix Enterprise Gateway with DRDA?" on page 1-4, the Gateway looks like a database server to an Informix client. The *gwservername* is analogous to the database server name or *dbservername* of a Dynamic Server or SE database server.

   In Figure 3-1, the name of the Gateway (*gwservername*) is **hillgw**.

2. For the **nettype** field, select the correct **nettype** value from the list in Figure 3-6.

Figure 3-6
Possible nettype Values for Making a Network Connection

nettype	Description
ontlitcp	Network connection using TLI with TCP/IP protocol
onsoctcp	Network connection using sockets with TCP/IP protocol
ontlispx	Network connection using TLI with IPX/SPX protocol

   To choose the correct **nettype** value, refer to the information about the UNIX network that you gathered in "The Communication Protocol" and "The Communications Interface" on page 3-9.

3. The host name of the Gateway computer in our example is **hill**.

4. Select the value for the fourth field, the **servicename**, as described in the following list:

   - If the network protocol is TCP/IP, use the **servicename** value that you found in "The TCP/IP servicename Value" on page 3-10.
   - If the network protocol is IPX/SPX, you can select any service name, provided it is unique among the names of services available on the IPX/SPX network. It is convenient to use the *gwservername*.

5. For the example configuration in Figure 3-1 on page 3-5, enter one of the following lines in the **sqlhosts** file, depending on which **nettype** value and **servicename** value you select.

   For a TCP/IP network using TLI:

   ```
   hillgw      ontlitcp      hill      TCP/IPservicename
   ```

   For a TCP/IP network using sockets:

   ```
   hillgw      onsoctcp      hill      TCP/IPservicename
   ```

   For an IPX/SPX network:

   ```
   hillgw      ontlispx      hill      hillgw
   ```

Connection to the Application Server

To determine the correct values for the entry in the **sqlhosts** entry on the Gateway for the connection between the Gateway and the application server, perform the following steps:

If you use SNA LU 6.2 protocol

1. Choose a database name (called the *alias_RDB_name*) for the client application to use when connecting to the application server.

   In the example in Figure 3-5 on page 3-16, the application server is named **AS1** and you choose **alias_AS1** for the *alias_RDB_name*.

   The client application always uses the *alias_RDB_name* when it makes a connection to an application server. The *alias_RDB_name* can be the same as a *real_RDB_name*.
2. The **nettype** field for the connection to the application server is always **drapplu6**.
3. The **hostname** field is the *real_RDB_name* (that is, it is the *RDB_name* of the application server).

4. The **servicename** field contains information necessary to make the LU 6.2 connection. (Refer to "Information About the Advanced Program-to-Program Communication" on page 3-10.) The information includes some of the following items:

   - `m=modename`
   - `g=sna_gateway_name`
   - `b=buffer_size`
   - `t=TPN`

   These parameters are platform specific. For more information about which parameters you require and their possible values, refer to the online machine notes file**.** Refer to "Information About the Advanced Program-to-Program Communication" on page 3-10.

   When you enter the parameter values in the **sqlhosts** file, you must include the `letter=` identifier (such as `m=` or `g=`). The values are separated by commas and no spaces are allowed.

Using the values from our examples, the **sqlhosts** entry in the Gateway should be similar to the following example:

```
alias_AS1      drapplu6      AS1      m=modeA,g=SNAname
```

To connect to the AS1 application server through the Gateway on **hill** as on page 3-5, the client application can issue the following statement:

```
CONNECT TO 'alias_AS1@hillgw'
```

If you use TCP/IP protocol

1. Choose a suitable *alias_RDB_name* for the TCP/IP connection. Assume that this name is **alias2_AS1**.
2. The **nettype** field is:

   drtlitcp for TCP/IP networks that use TLI API

   drsoctcp for TCP/IP networks that use sockets API
3. The **hostname** field should contain the host name of the computer system running the application server. You can specify the IP address instead of the host name.

4. The **servicename** field should contain the service name that the application server uses. You can also specify the port number instead. The number 446 is a well-known port number for Informix Enterprise Gateway.
5. The **options** field should at least specify the *real_RDB_name* of the application server that uses the RDB= keyword.

The sqlhosts entry for the TCP/IP connection to the application server is similar to the following example.

```
alias_AS1      drsoctcp      db2host      446      RDB=AS1
```

Multiple Connection Modes

The administrator of the Gateway can use multiple alias names to set up multiple communication modes for each application server. To choose the connection characteristics for an application, the application programmer can specify the desired alias in the CONNECT or DATABASE statement. "The modename Value" on page 3-10 discusses modes.

The mode name and the *real_RDB_name* form a unique pair. If you use multiple modes, you must have an *alias_RDB_name* entry in the **sqlhosts** file for each mode.

The table in Figure 3-7 illustrates possible connections. The **sqlhosts** entries that correspond to Figure 3-7 might be as Figure 3-8 shows.

Figure 3-7
Illustration of alias_RDB_names for Different Modes

alias_RDB_name	Application Server Name (real_RDB_ name)	modename
nyc_A	NYCITY7	modeA
nyc_B	NYCITY7	modeB
blue1	BIGBLUE	C_mode
blue2	BIGBLUE	D_mode

(1 of 2)

alias_RDB_name	Application Server Name (real_RDB_ name)	modename
red1	BABYBLUE	BABYLU
green1	biggreen	Not applicable (TCP/IP)
green2	BIFGREEN	Not applicable (TCP/IP)

(2 of 2)

Figure 3-8
Sample sqlhosts Entries Showing Multiple Modes

servername	nettype	hostname	servicename	options
nyc_A	drapplu6	NYCITY7	m=modeA, g=SNAname	
nyc_B	drapplu6	NYCITY7	m=modeB, g=SNAname	
blue1	drapplu6	BIGBLUE	m=C_mode	
blue2	drapplu6	BIGBLUE	m=D_mode	
red1	drapplu6	BABYBLUE	m=LU62!IFMXLU!BABYLU	
green1	drsoctcp	db2host	db2port	RDB=BIGGREEN
green2	drsoctcp	158.58.10.175	446	RDB=BIGGREEN

Not all LU 6.2 implementations require the *sna_gateway_name*. The *buffer_size* is optional for all implementations. The TPN parameter is not usually used for DB2 or OS/400. For SQL/DS, TPN must be set to the *RDB_name*, for example, `t=SQLDS1`.

Preparing the sqlhosts File for the Client Application

On the computer where a client application resides, the **sqlhosts** file must have an entry for each Gateway to which the client can connect.

Connection to a Local Gateway Using Unnamed Pipes

When the client application is on the same computer as the Gateway, the **sqlhosts** entry for the client application is the entry that you prepared in "Local Connection" on page 3-17. You do not need to add anything else.

Network Connection

When the client application and the Gateway are on different computers or in different **$INFORMIXDIR** directories on the same computer, the **sqlhosts** entry is the same as the entry in "Network Connection" on page 3-17 with one exception: on a TCP/IP network one computer can use a sockets interface while the other computer uses a TLI interface. You must determine which interface the client-application computer uses and choose the correct value from the table in Figure 3-6 on page 3-18.

For example, if the Gateway computer uses sockets on TCP/IP and the client-application computer uses TLI on TCP/IP, the **sqlhosts** entries for the configuration in Figure 3-1 on page 3-5 would appear as shown in the following examples.

The **sqlhosts** entry on the Gateway computer:

```
hillgw      onsoctcp     hill      TCP/IPservicename
```

The **sqlhosts** entry on the client-application computer:

```
hillgw      ontlitcp     hill      TCP/IPservicename
```

The two **sqlhosts** entries are the same except for the middle three letters of the **nettype** entry.

Step 4: Start the Gateway Daemon

If any client applications use a network connection to the Gateway, you must start the Gateway daemon **gwd**. If you are not using the Gateway, you can skip this step. The **gwd** daemon enables the Gateway to receive a network connection request from a client application. In a network connection, the client attaches to the Gateway daemon, **gwd**. The Gateway daemon then spawns a Gateway process, attaches the client to the new Gateway process, and detaches itself.

Important: *In a local mode (unnamed pipes) client connection, the client application spawns the Gateway process directly. For this type of client connection, a Gateway daemon is not needed.*

To start the Gateway daemon process

1. Log in as **root** on the Gateway computer.
2. Update the **gwenv.sh** or **gwenv.csh** environment configuration files with the necessary values for the environment variables you will need.
3. Source the **gwenv.sh** or **gwenv.csh** files.

   For a Bourne shell, enter:

   ```
   $ . gwenv.sh
   ```

   For a C shell, enter:

   ```
   % source gwenv.csh
   ```
4. You can add optional environment variables to these files. See “Optional Environment Variables” on page 3-25.
5. Create a symbolic link to the SNA stub library if necessary. See “Using the SNA Stub Library” on page 3-28.
6. Enter the following command:

   ```
   gwd gwservername -s gw
   ```

The *gwservername* assigns a name to the Gateway server. The **-s** option tells the **gwd** daemon to spawn the Gateway executable file (**gw**). Use the *gwservername* that is in the **dbservername** field of the **sqlhosts** file. If an entry for *gwservername* does not exist in the **sqlhosts** file, you must add an entry. Refer to “Step 3: Prepare the sqlhosts File” on page 3-13.

Starting a Daemon Log File for the Gateway Process

You can start the **gwd** daemon with a log file by including the **-l** option, as shown in the following example:

```
gwd gwservername -s gw -l logfile
gwd gwservername -s gw -l full_logfile_pathname
```

If you specify only a filename, *logfile* is stored in the directory that was the current directory when the **gwd** command was executed. If you specify a full pathname, *full_logfile_pathname*, you can select the directory where you want to store the log file (for example, **/tmp/my_gw.daemonlog**).

Important: *The log file can grow to a large size. Use a log file only if you are trying to troubleshoot the connection between the application client and the Gateway.*

The log file contains information about Gateway invocations. Figure 3-9 shows the format of the log file and two sample entries.

Figure 3-9
Sample Excerpt from a Gateway Log File

Date	Time	Client Type	Client Hostname	User	Operation
1996-08-12	10:44:30.234352	sqlexec	server1	esther	-d / remote
1996-04-14	15:25:45.176553	sqlexec	abacus	cedric	-p

The following operations might appear in the log file.

-c, -r creates a database; removes a database
Trying to create or remove a database causes the Gateway to return a -349 error, but before that error is generated, the **-c** or **-r** entry is placed in the log file.

-d selects a database

-l lists available databases

-p allows access from a remote server or PC client

-x lists available databases through DB-Access

Important: *When the client application spawns the Gateway process with IPC-pipe mode, no* ***gwd*** *daemon is involved and you cannot start a log file for the* ***gwd*** *process. You can run both client applications that use IPC-pipe mode and client applications that use the* ***gwd*** *daemon (network mode) on the same computer. The* ***gwd*** *log file records only the Gateway processes that the* ***gwd*** *daemon started.*

Starting Multiple Gateway Daemons

In most cases, a single Gateway daemon is sufficient in a local UNIX network. You might want to start multiple Gateway daemons for the following reasons:

- To allow Dynamic Server to connect to multiple application servers for distributed access

  In this case, Informix recommends a Gateway daemon for each application server.

- To provide different locales (language characteristics) when using pre-Version 7.x client applications

  The locale of the Gateway is the UNIX locale that is in effect when the Gateway daemon starts. If you are using a pre-Version 7.2 client application, you must make the locale of the Gateway agree with the locale of the client application. Refer to the UNIX man pages for **setlocale()** and **locale()** for information about how to set or change the UNIX locale.

  For a Version 7.2 and later Gateway, set the **DB_LOCALE** environment variable in the daemon's environment.

- To provide different connection modes for different types of client applications

  Refer to "The modename Value" on page 3-10.

- To provide extra capacity when many users are connecting to the Gateway at the same time

You must start every Gateway daemon with a unique *gwservername*.

When to Explicitly Set the Locale of the Gateway Daemon

During execution, the Gateway uses the GLS locale (specified in the **DB_LOCALE** environment variable setting) of the client application. The Gateway uses locale information to specify the character code set that the current client application uses, and to classify the characters received from the client application into categories such as alpha, numeric, and punctuation.

Locale for Version 7.1 or Earlier Clients

If the client application is a pre-Version 7.1 client application, the client does not send its locale to the Gateway, and the Gateway uses the **DB_LOCALE** value set in its own environment. That is, the Gateway assumes that the locale of the client is the same as its own **DB_LOCALE** setting. If **DB_LOCALE** is not set in the Gateway environment, **en_us.8859-1** is used.

In this case, you must set the locale of the Gateway to match the locale of the client. For local connections, this happens automatically because the Gateway process is a child of the client-application process. For network connections, you must set the appropriate environment variables before you start the Gateway daemon.

Each Gateway daemon can support only one pre-Version 7.1 client locale. If different locales are used on pre-Version 7.1 clients, you must start a separate Gateway daemon to support each distinct client locale.

Locale for Version 7.2 or Later Clients and for Client SDK

If the client application is a Version 7.2 or later client application, the Gateway uses the locale that the client sends to the Gateway. Each Gateway daemon can support any number of different locales that Version 7.2 or later client applications use.

You do not need to set the locale of the Gateway daemon if all client applications are Version 7.2 or later.

Optional Environment Variables

Each spawned Gateway process inherits all the environment variables that were set when the daemon was started. For example, if you want to use a locale other than **en_us.8859-1** (the default), you must set the **DB_LOCALE** environment variable before you start the daemon. Other environment variables that you might want to set are **DBTEMP** and **GWDEBUG**. (See "The Trace File" on page 6-75.)

Using the SNA Stub Library

If you use TCP/IP protocol for DRDA, and if SNA is not installed on the Gateway computer, you might want to create a *one-time* symbolic link from **/usr/lib/sna_library_name** to **$INFORMIXDIR/lib/sna_library_name**. If **$INFORMIXDIR/lib/libsnastub.a** exists, a symbolic link is required.

If you are working on AIX, for example, enter the following command at the prompt:

```
ln -s $INFORMIXDIR/lib/libsna.a /usr/lib/libsna.a
```

You can find the command you need in the on-line machine notes file.

Step 5: Use the gwdba Utility

You use the **gwdba** utility to perform the following administrative tasks:

- Bind and drop packages at the application server
- Specify *RDB_User_ ID*s (*optional*)
- Specify *alias_RDB_name* to package-name mapping
- Install Informix catalogs (*optional*)
- Test the connection to the application server (*optional*)

You are not required to install the Informix catalog when you first start the Gateway, but you might want to install it later to potentially improve performance.

At this point, while you are configuring the Gateway, you must perform the following tasks:

1. If necessary, assign an *RDB_User_ID* for user
2. Bind packages at the application server

Instructions for using **gwdba** are in Chapter 5, "The gwdba Utility."

Step 6: Prepare for Direct or Distributed Connections

In a direct connection, the client application connects directly to the Gateway. In distributed connections, the client application connects to an Informix Dynamic Server, which connects to the Gateway.

Direct Connections

When the client application connects directly to the Gateway, you must set the environment variables that the client application requires and the environment variables that the Gateway requires.

For information about environment variables that the client application requires, refer to the application documentation. Information about environment variables used with GLS features is in "Environment Variables the Gateway Uses for GLS" on page 4-4.

The following sections describe the environment variables that the user of a client application must set for the following configurations:

- A Version 7.x / Version 9.x local connection (unnamed pipes)
- A Version 5.x local connection (unnamed pipes)
- A Version 7.x / Version 9.x network connection
- A Version 5.x network connection using INFORMIX-NET
- A Version 5.x network connection using INFORMIX-NET relay module
- A Version 5.x network connection using the Version 7.x relay module

A Version 7.x or 9.x Local Client Connection Using Unnamed Pipes

When you establish a connection from a Version 7.x or 9.x client application to the Gateway with a local connection (unnamed pipes), the client application and the Gateway must be installed in the same directory. The following table shows the environment variables that the user must set.

Environment Variable	Set To
INFORMIXDIR	Directory where the products are installed
INFORMIXSERVER	The Gateway server name (*gwservername*) (see page 3-17)

A Version 5.x Local Client Connection Using Unnamed Pipes

When you establish a connection from a Version 5.x client application to the Gateway using a local connection (unnamed pipes), the client application and the Gateway must be installed in the same directory. The following table shows the environment variables that the user must set.

Environment Variable	Set To
INFORMIXDIR	Directory where the products are installed
INFORMIXSERVER	The Gateway database server name (*gwservername*) (see page 3-17)
SQLEXEC	**$INFORMIXDIR/lib/sqlrm**

Network Connections

When the client application connects to the Gateway using network connections, the user of a client application must set the required environment variables before starting the client application.

A Version 7.x or 9.x Network Client Connection

The user of a Version 7.x or 9.x client application, which attaches to the Gateway using a network connection, must set the environment variables in the following table.

Environment Variable	Set To
INFORMIXDIR	Directory where the client product is installed
INFORMIXSERVER	The Gateway database server name (*gwservername*) (see page 3-17)

A Version 5.x Network Client Connection Using INFORMIX-NET

The user of a Version 5.x client application, which attaches to the Gateway using an INFORMIX-NET network connection, must set the environment variables in the following table.

Environment Variable	Set To
INFORMIXDIR	Directory where client application is installed
SQLEXEC	**sqlexec**

A Version 5.x Network Client Connection Using the INFORMIX-NET Relay Module

The user of a Version 5.x client application, which attaches to the Gateway in network mode using the INFORMIX-NET relay module, must set the environment variables in the following table.

Environment Variable	Set To
INFORMIXDIR	Directory where the client application is installed
SQLRMDIR	**$INFORMIXDIR/lib**
SQLRM	If the computer uses sockets: **sqlrmsoctcp** If the computer uses TLI: **sqlrmtlitcp**

A Version 5.x Network Client Connection Using the Version 7.x Relay Module

The Version 7.x Relay Module is a component of Informix Dynamic Server and Informix Enterprise Gateway with DRDA. The primary purpose of the Version 7.x Relay Module is to enable pre-Version 7.x client applications to connect to a local Version 7.x database server (Dynamic Server or Gateway) by using a local connection (unnamed pipes).

However, you can also use the Version 7.x Relay Module for network connections from a pre-Version 7.x client to the Version 7.x database servers or to the Gateway. Figure 3-10 illustrates a configuration that involves a pre-Version 7.x client application that connects to the Gateway in network mode using the Version 7.x Relay Module.

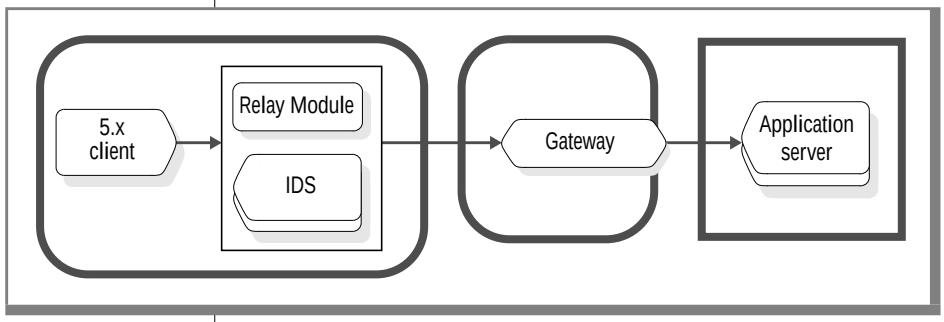


Figure 3-10
A Remote Connection Using the Version 7.x Relay Module

For this configuration, the user of a client application must set the environment variables in the following table.

Environment Variable	Set To
INFORMIXDIR	Directory where the client application is installed
INFORMIXSERVER	The Gateway database server name (*gwservername*) (see page 3-17)
SQLEXEC	**$INFORMIXDIR/lib/sqlrm** (the **INFORMIXDIR** of the Version 7.x database server)

Distributed Processing

When Informix Dynamic Server serves as the coordinator for distributed processing and the Gateway is a subordinate server, Dynamic Server is configured in the same way as for a direct connection to the Gateway, except for the following conditions:

- The connection must be a network connection.
- Dynamic Server does not require any environment variables.

Step 7: Modify System-Startup Scripts

You must restart the Gateway daemon whenever the Gateway computer is rebooted. If you want the Gateway daemon to start automatically, add the necessary environment-variable settings and the **gwd** command line to the appropriate UNIX start-up file (such as **/etc/rc** or **/etc/rc.local**).

Using the Gateway Daemon to Bypass User ID Authentication

The Gateway daemon provides a mechanism to bypass the authentication of the client user ID by using the operating system user ID and password on the Gateway computer. The authentication of the client user is deferred to the remote database computer.

Any valid user ID on the client computer can use the Gateway, even if that user ID or the user name specified in the USER clause of the SQL CONNECT statement is not valid on the Gateway computer. It eliminates having to add and maintain all of the client user IDs that use the Gateway on the Gateway computer.

This feature is implemented in the Gateway by additional options in the **gwd** daemon. For a description of the **gwd** command, see "Starting a Daemon Log File for the Gateway Process" on page 3-25. The following example shows how to use the **gwd** command:

```
gwd gwservername -l logfile -s gw_program -b [userid]
```

To bypass user ID authentication, use the following options:

-b bypasses the client's user ID authentification.

userid specifies the owner of the Gateway process. If you do not specify the user ID, **informix** is the owner of the process.

How to use the -b [userid] option

When you specify the **-b** option, **gwd** bypasses the authentication of the client's user ID against the operating system user ID and password pair. The client user is treated as a trusted user and is given access to the **gw** executable file.

If you do not specify a value for the user ID in conjunction with the **-b** option, the owner of the spawned **gw** process is made to be **informix**. If you do specify a value for user ID, the owner of the spawned **gw** process is changed to the specified user ID.

When you specify the **-b** option, the effective user ID of the client and the client-supplied password (which is mandatory) are used to connect to the application server. Refer to "Enforcing Security" on page 5-12 for further details.

Note that the owner of all the temporary files generated for handling scroll cursors and the Gateway trace are the same as the owner of the **gw** process.

If the user ID you specify does not exist on the system, the **gwd** aborts with the following message:

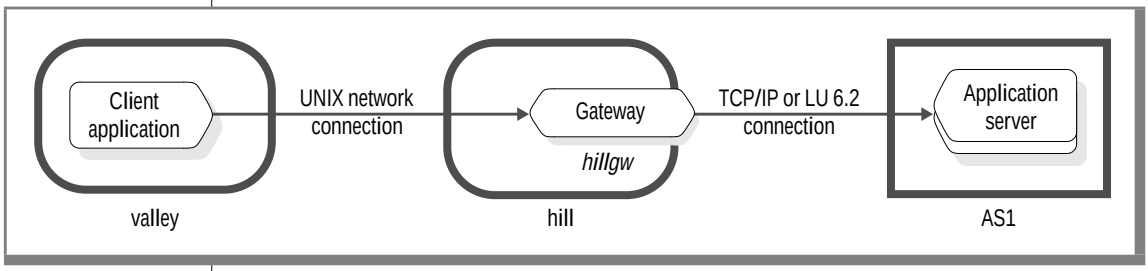
```
User username is not known on this computer. Please specify
a valid user for the -u option and reissue the command.
```

Refer to "Finding the User ID for the Application Server" on page 5-15, which shows how you can use the bypass feature.

Direct Mode Connection

Figure 3-11 illustrates a configuration that connects to the Gateway in direct mode.

Figure 3-11
A Direct-Mode Connection



If you want to bypass user ID authentication in a direct mode connection, you must specify the **-b** option in **gwd**. The following items are not necessary:

- You do not need the user ID mapping entries created with the **gwdba** tool on the Gateway computer **hill**.
- You do not need an entry in the **.netrc** file on the Gateway computer **hill**.
- The effective user ID for the client on **valley** does not need to be a valid operating system user ID on the Gateway computer **hill**.

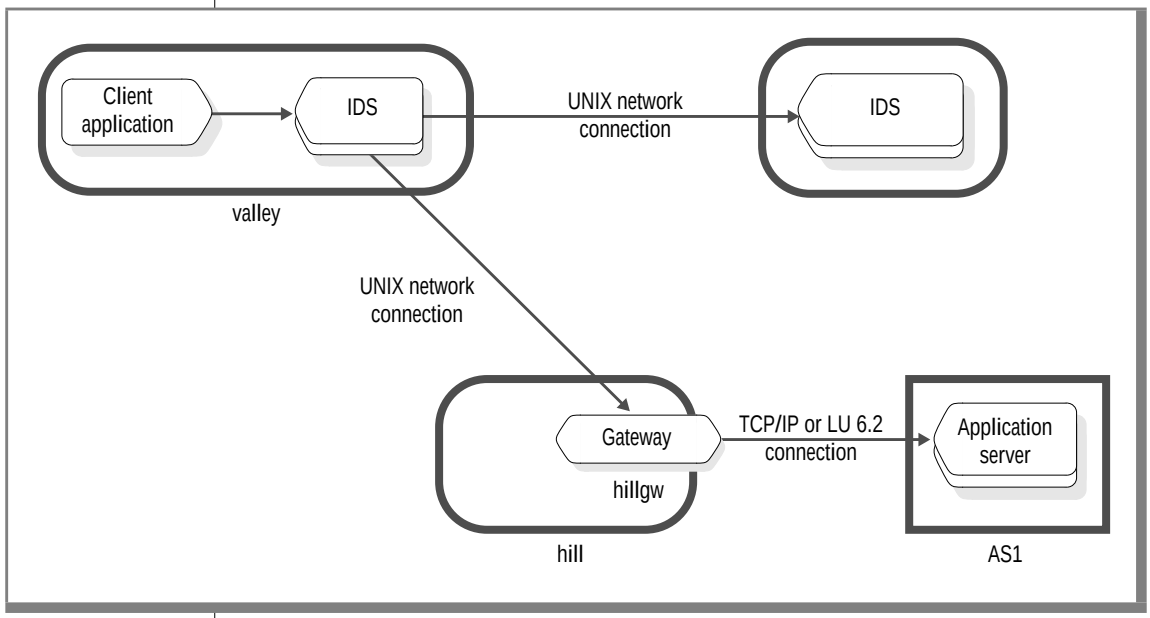
However, the client's user ID and password must be present on the remote application server.

For example, in Figure 3-11 the client application user on **valley** can connect to the remote application server **AS1** if the user ID and password that the client application user specifies on **valley** represents a valid user ID and password pair on the remote application server **AS1**.

Distributed Mode Connection

Figure 3-12 illustrates a distributed-mode connection.

Figure 3-12
A Distributed-Mode Connection



If you want to bypass user ID or password authentication in this mode, you must perform the following tasks:

- Specify the **-b** option in **gwd**.
- Make sure the effective user ID of the client is present as a valid operating system user ID on the Informix Dynamic Server computer **valley.** Note that the client's user ID does not need to exist on the Gateway computer **hill**.
- Create a **.netrc** file on the Dynamic Server computer **valley**.
- Enter the **login** and **password** fields that specify a valid user on the remote application server **AS1** into the Gateway entry **hill** in the **.netrc** file on the Dynamic Server computer **valley**.

With this configuration, you do not need the user ID mapping entries that the **gwdba** tool creates, or an entry in the **.netrc** file on the Gateway computer **hill**.

Gateway Administrator Performance Issues

The Gateway administrator needs to understand the following system-performance issues:

- Effect of network speed on performance
- Keeping statistical information current

Effect of Network Speed on Performance

Network factors that affect the performance of SQL queries, which transfer data over a network from an application server to the Gateway, are described in the following list:

- The raw transfer rate of the network
- The number of messages that are required to transport the data
- The buffer sizes that the software and hardware components that are involved in the transfer use
- Concurrent network traffic

The Raw Transfer Rate of the Network

The raw transfer rate of a network is determined by its slowest link. A typical Informix Gateway-to-IBM mainframe network has at least the following links:

- Link 1: the Informix Gateway is connected to an IBM controller.
- Link 2: the IBM controller is connected (channel attached) to the IBM mainframe.

Raw transfer rates for Link 1 networks range from 1,200 bits per second on a modem to 2 megabits per second on a Token Ring. Raw transfer rates for Link 2 networks range from 1 to 4.5 megabits per second.

Number of Messages That Are Required to Transport the Data

A fixed cost is associated with any message exchange on a distributed network. The fewer the messages required to transport a given amount of data, the less CPU and elapsed time is consumed.

The obvious way to minimize the number of messages that are used to send data across a network is to send more data with each message. The Gateway lets you specify how much data from an answer set should be sent in each message from the application server. The maximum amount of data that can be sent in a message is limited by the buffer size.

Buffer Sizes

In the DRDA protocol, when data is fetched from an application server in read-only mode, many rows of data are placed into a single buffer and sent to the Gateway. The Gateway does not send the application server another fetch request until it exhausts the data in the buffer. The larger the buffer, the better the throughput, because fewer messages need to be exchanged to receive the same amount of data.

However, if the underlying network protocol uses a smaller buffer size than the buffer size of the Gateway, increasing the buffer size does not help dramatically. Similarly, the physical buffer sizes that the network hardware uses can negate the effect of increasing the buffer size.

The Gateway buffer size defaults to 16 kilobytes for an LU 6.2 connection, and 4 kilobytes for a TCP/IP connection. You can set the buffer size in the **sqlhosts** file. For an LU 6.2 connection, set the *buffer_size* parameter `[b=]` in the **servicename** field of the entry for the application server. For a TCP/IP connection, specify the buffer size in the `[b=]` parameter of the **options** field. Buffer sizes range from 512 to 32,767 bytes for an LU 6.2 connection, and 518 to 32,773 bytes for TCP/IP. For more information, see "Connection to the Application Server" on page 3-19.

Concurrent Network Traffic

When a network is heavily loaded, less of its raw transfer rate is available to any one user.

Network Transfer-Time Example

Figure 3-13 shows the approximate theoretical network transfer time at different transfer rates for a table of the following size:

```
6,000,000 rows of 40 bytes per row
```

Figure 3-13
Theoretical Network Transfer Time for a Table That Contains 240 Million Bytes

Transfer Rate	Theoretical Transfer Time
Modem at 19,200 bits per second	28 hours
Token Ring at 4 megabits per second	8 minutes
Token Ring at 16 megabits per second	2 minutes

The actual network transfer time is longer, perhaps substantially longer, depending on buffer sizes, network traffic, and so on.

The total amount of time that is required to transfer the data from the application server to the client application or Dynamic Server system is not estimated here. Only the network transfer time from the application server to the Gateway is estimated in this example. The total transfer time involves the following factors:

- Application-server processing to retrieve the data
- Network transfer time to the Gateway
- Gateway processing
- Network- or unnamed-pipe-mode transfer time to the client application or Dynamic Server
- Client application or Dynamic Server processing

Keeping Statistical Information Current

Much of the query optimization that you can apply in distributed queries depends on the correctness of table statistics that are available from the system catalog information of the server. Dynamic Server, acting as the transaction coordinator, handles optimization issues for the Gateway. Dynamic Server has an efficient cost-based network-query optimizer, but it must have correct table statistics to perform properly. For example, depending on the system-catalog information that it receives, Dynamic Server might use indexed columns to access data or it might choose to perform a join remotely instead of locally.

The Gateway administrator must ensure that table statistics information on application servers and on Dynamic Servers is kept current for the tables that the Gateway applications use.

Updating Statistics on Informix Dynamic Server

To update statistics on a Dynamic Server, execute the SQL statement UPDATE STATISTICS. Use the SET EXPLAIN ON statement to find out about the choices that are available to the optimizer and their costs. For information about the syntax and use of these statements, refer to the *Informix Guide to SQL: Syntax*.

Updating Statistics on Application Servers

To update statistics on an application server, you need to work with the DBA of that application server. On some application servers, you update statistics by running a utility. For example, DB2 provides the utility **runstats** to update statistics.

If you have installed the Informix catalog on the application server, you must also update the statistics information in the Informix catalog, but only after the application server statistics are updated. You do this by executing the **Refresh** option on the **gwdba** catalog panel. (Refer to "The Refresh-tables Option" on page 5-37.)

Chapter

4

Global Language Support

In This Chapter

Global Language Support (GLS) refers to the tools that Informix provides for working with languages other than English. The tools include both conversion programs and the conversion tables that those programs require. GLS has two facets:

- Sharing data among computer systems that use different character representations
- Using multibyte character names for program objects

Informix Enterprise Gateway with DRDA supports the GLS conversion between database servers of single-byte and multibyte character data. The Gateway supports languages whose characters are represented by code points greater than 1 byte in length.

Multibyte characters are used primarily for Asian languages, but the capabilities can support any multibyte language. The Gateway support of multibyte languages includes the following characteristics:

- Multibyte character data code-set conversion (SQL statements, input host-variable values, output select list values) between the code set of the client and the code set of the application server
- Multibyte character SQL identifiers (table names, column names, and so on) in SQL statements
- Multibyte literal character string constants in SQL statements

This chapter describes environment variables used for GLS and lists GLS-related error messages. For full details of the features and functionality of GLS in Informix products, including locales and code-set conversion, see the *Informix Guide to GLS Functionality*.

Environment Variables the Gateway Uses for GLS

Informix Enterprise Gateway with DRDA uses the following environment variables to support GLS:

- **DBLANG**
- **DB_LOCALE**
- **GL_PATH**
- **GWASCCSID**
- **GWOUTBCONV**

The Gateway obtains locale information from various environment variables at the client and Gateway computers. For information about GLS environment variables, including rules of precedence, see the *Informix Guide to GLS Functionality*.

The DBLANG Environment Variable

The **DBLANG** environment variable specifies the language of the message output to the user. The **DBLANG** setting completes the directory specification that Informix products use to find files that contain the fixed error text of messages. Different directories contain files with messages in different languages.

Informix does not recommend a generic value for **DBLANG** because the directory structure that is used to contain message files varies according to the client product that is used. Consult the client product documentation to determine whether to set **DBLANG**, and if so, to what value. The Gateway administrator must copy the Gateway messages into the appropriate client directories. For more information, see "Updating the Error-Message Files" on page 2-11.

The DB_LOCALE Environment Variable

For clients prior to Version 7.2, the Gateway administrator can specify the **DB_LOCALE** setting in the Gateway environment to determine the Informix GLS locale to use for code-set conversions. Version 7.2 and later clients pass the **DB_LOCALE** setting in the client environment to the Gateway. If this locale maps to a mixed-byte CCSID (a combination double-byte and single-byte CCSID), the Gateway automatically performs the outbound conversion for character data.

For Version 7.2 and later clients, the **DB_LOCALE** setting at the client overrides the value at the Gateway. If the **DB_LOCALE** value is not inherited from the client environment and is not set in the Gateway environment, and the client version is Version 7.2 or later, the default Informix GLS locale is **en_us.8859-1**. This default value also applies to client versions prior to Version 7.2, if **DB_LOCALE** is not set at the Gateway.

The GL_PATH Environment Variable

The default location of the code-set conversion tables for GLS is the **$INFORMIXDIR/gls/cv** subdirectory. The default location of the Informix GLS locale files is **$INFORMIXDIR/gls/lcn**.

You can override the default locations by setting the **GL_PATH** environment variable to a search path that can locate both the conversion files and the Informix GLS locales. You can set **GL_PATH** to include multiple directory names using standard UNIX syntax, as the following example shows:

```
setenv GL_PATH directory_pathA:directory_pathB:directory_pathC
```

When **GL_PATH** is set, the directories are not subdivided into **lc** and **cv** subdirectories.

The files that contain the tables in the **$INFORMIXDIR/gls/cv** directory are described in the *Informix Guide to GLS Functionality*.

Warning: *Do not edit any files in the $INFORMIXDIR/gls/cv directory.*

The GWASCCSID Environment Variable

The Gateway administrator should set the **GWASCCSID** environment variable only if a -29032 error is encountered. If a -29032 error is encountered, then **GWASCCSID** must be set to the single-byte CCSID of the application server before you attempt the connection again. For details, see the *Informix Error Messages* manual.

The GWOUTBCONV Environment Variable

Use the **GWOUTBCONV** environment variable to specify outbound conversion for a single-byte CCSID.

If the Gateway obtains a locale that refers to a single-byte CCSID, the Gateway does not do any outbound conversion of character data. It lets the application server use its translation tables to convert the incoming character data. However, for some single-byte CCSIDs, you might want the Gateway to do outbound code-set conversion instead of having the application server perform inbound code-set conversion. For example, if the application server cannot perform a particular character conversion, you can configure the Gateway to perform outbound conversion by setting **GWOUTBCONV**.

The format of **GWOUTBCONV** is a list of Informix GLS locales and their *real_RDB_name*s. Lists are separated by colons, and, within each list, the locale and the *real_RDB_name* are separated by a comma. The following examples show a valid **GWOUTBCONV** setting:

```
setenv GWOUTBCONV 'en_us.819,PORTLAND'

setenv GWOUTBCONV 'en_us.819,PORTLAND:De_DE.88591,BONN'
```

The first setting tells the Gateway to perform outbound conversion when connecting to the application server with the *real_RDB_name* PORTLAND if the locale specification of the Gateway computer is **en_us.819**.

The second setting tells the Gateway to perform outbound conversion when connecting to the application server with the *real_RDB_name* PORTLAND if the locale specification of the Gateway computer is **en_us.819** and when connecting to the application server with the *real_RDB_name* BONN if the specified locale is **De_DE.88591**.

You can set **GWOUTBCONV** in the Informix configuration files. If you set it in your configuration file **$HOME/.informix**, this value overrides the settings in the centralized **$INFORMIXDIR/etc/informix.rc** configuration file of the Gateway. If you set the **GWOUTBCONV** environment variable before the Gateway daemon starts, this value overrides the settings in both the configuration files.

Tip: *For double- and mixed-byte CCSIDs, the Gateway always performs outbound conversion.*

For information about code sets and code-set conversion tables, see the *Informix Guide to GLS Functionality*.

The Application Server CCSID Cache File

When the Gateway performs outbound character code-set conversion, the Gateway creates and maintains a cache of the CCSIDs (code sets) that the application servers use. The Gateway maintains this cache in the file **$INFORMIXDIR/gw/sysinfo/prnccsid.dat**.

Warning: *Do not edit the **$INFORMIXDIR/gw/sysinfo/prnccsid.dat** file.*

GLS-Related Errors

The following error codes identify GLS-related errors:

- -29032
- -29033
- -29034
- -29036
- -29037

These errors and their corrective actions are explained in the *Informix Error Messages* manual.

Chapter

5

The gwdba Utility

In This Chapter

The **gwdba** utility is included in the Informix Enterprise Gateway with DRDA installation. The Gateway system administrator uses the **gwdba** utility to create and maintain the Gateway system control files. Users other than the Gateway administrator can use the **gwdba** utility to add or update their own user ID mapping entries.

This chapter discusses the following features of the **gwdba** utility:

- Adding user ID mapping entries
- Binding packages
- Installing and maintaining the Informix catalog
- Installing X/Open-style information schemas
- Testing the connection to the application server

For information on how to use the **GWVTOT** environment variable with the **gwdba** utility, see "Using GWVTOT with the gwdba Utility" on page 6-59.

Executing the gwdba Utility

To execute the **gwdba** utility, follow these steps:

1. Log in as user **informix**.
2. Set your **INFORMIXDIR** environment variable to point to the directory where your Gateway product is installed.
3. Set your **PATH** environment variable to include **$INFORMIXDIR/bin**.
4. Set **INFORMIXSERVER** to the Gateway server name.

5. Set your **DBPATH** environment variable to include **$INFORMIXDIR/forms**.

   The **gwdba** utility is written in INFORMIX-4GL. You need to prepare **DBPATH** so that 4GL can find the information it needs.

6. If you are not using a U.S. English character set, set the **GWDBALOCALE** environment variable to specify the Informix GLS Locale.

   If you enter characters into the **gwdba** screens from a locale that is not equivalent to the **en_us.8859-1** locale, or input such characters into the file for the **Add-tables** option, you must first set the **GWDBALOCALE** environment variable to the Informix GLS locale that is appropriate for the character set.

7. Set your **TERM** environment variable to `vt100`.

   If **gwdba** does not recognize your terminal type, try using an xterm window and setting **TERM** to `vt100`.

   The **gwdba** utility is a 4GL program that uses the settings of the **INFORMIXTERM**, **TERM**, **TERMCAP**, and **TERMINFO** environment variables to determine how to display information on your terminal based on the **termcap** file or **terminfo** directory.

   Informix Enterprise Gateway with DRDA ships with a **termcap** file in the **$INFORMIXDIR/etc** directory. To explicitly select this **termcap** file, use one of the environment variables listed in the previous paragraph. For further information on the **termcap** file, see the discussion of the **TERMCAP** environment variable in the *Informix Guide to SQL: Reference*.

8. You can set environment variables at the system prompt or in an environment-configuration file. If you set your environment variables in an environment-configuration file, source that file before you start **gwdba**.

   - For a C shell, enter:

     ```
     source gwenv.csh
     ```

   - For a Korn or Bourne shell, enter:

     ```
     . gwenv.sh
     ```

9. Enter the following command at the system prompt to execute the **gwdba** utility:

   ```
   gwdba
   ```

Multibyte Character Values in gwdba

You cannot enter multibyte character values into the screens of **gwdba**. The ultimate use of the information that you provide to **gwdba** (user IDs, passwords, collection IDs, relational database names, and so on) restricts the information to single-byte character data. Currently, most messages that the **gwdba** utility issues are hard-coded in ISO 8859-1 English.

You can set the **GWDBALOCALE** environment variable to maintain table names with multibyte characters in the Informix catalog. See "Entering Table Names with Multibyte Characters into the Informix Catalog" on page 5-39.

Files That gwdba Creates

The **gwdba** utility creates the **$INFORMIXDIR/gw/sysinfo** directory the first time you execute **gwdba**. The **User** and **Bind-Package** options each create two files, a data file (extension **.dat**) and an index file (extension **.idx**), and store them in the **$INFORMIXDIR/gw/sysinfo** directory. The **gwdba** utility creates files as follows:

- The **User** option creates **gwuser** files.
- The **Bind-Package** option creates **gwbind** files.
- The **Bind-Package** option creates **gwaplpkg** files if you choose the `Apl-to-Pkg` option in the **Bind-Package** screen.

Warning: *Do not use a text editor to edit the files that the **gwdba** utility creates. If you use a text editor on these files, they become unusable.*

Conventions That gwdba Uses

The **gwdba** utility uses the same conventions as other Informix utilities, such as DB-Access, as the following list describes:

- The active option is highlighted. To move the highlight to another option, you can press the SPACEBAR or the arrow keys.
- To select an option, highlight the desired option and then press RETURN or type the first letter of the option.
- The ESC key records a choice.

- CTRL-C cancels a choice.
- On most screens, CTRL-W summons help. You can also press CTRL-W while the cursor is positioned on any field in a data-entry screen to get help about the field.

Displaying the Software Version Number by Using gwdba

You can display the name, version number, and serial number for the Gateway by using the **-V** option of **gwdba**. When you specify the **-V** option, you cannot specify any other options with it.

The gwdba Main Menu

The menus and screens of the **gwdba** utility are illustrated in the next several sections. Figure 5-1 shows the main menu, which lets user **informix** choose one of the following tasks:

- Enter user ID mapping entries (page 5-7)
- Manage package binds (page 5-16)
- Install and maintain the Informix catalog (page 5-30)
- Install the X/Open-style information schema (page 5-40)
- Test the connection to the application server (page 5-42)

Figure 5-1
The gwdba Main Menu for User informix

```
Main:   User  Bind-Package  Catalog  Schema  Test-connect  Exit
Maintain user information

---------------------------- Press CTRL-W for HELP -----------------------
```

Figure 5-2 shows the main menu that regular users (those who are not user **informix**) see.

Figure 5-2
The gwdba Main Menu for Other Users

```
Main:   User  Test-connect  Exit
Maintain user information

---------------------------- Press CTRL-W for HELP -----------------------
```

The User Screen

With the User screen, map the UNIX user ID of each Gateway user to the user ID and password that serve to connect to an application server. The specified application server user ID and password can be different for each UNIX user ID / application-server pair. That is, if user **helen** uses two application servers, there will be two entries for user **helen** on the User screen. See Figure 5-4 on page 5-9. However, the Gateway and application server security can also be configured to give a user access to an application server without defining such a map. For more information, see "Enforcing Security" on page 5-12.

The User screen lets you add, update, or delete one or more user ID entries in the centralized user ID or password-lookup table that is stored in the **gwuser** file.

The Gateway administrator, that is, a user with user ID **informix**, can perform all the functions on the User screen for any user ID. Regular users that is, users who have user IDs other than **informix**, can also perform all the User screen functions, but only for their user IDs.

If you configure security to use the lookup table of the Gateway, each user must have an entry before using the Gateway. Either the Gateway administrator must make the entry for the Gateway user, or the user can make the entry. In one scenario, the Gateway administrator might make the first entry for a user because the entry requires knowledge of the *real_RDB_name*, which is not usually known by the user. At a later time, the user might update the password because the Gateway administrator does not want to maintain password information for each user.

Basic User-Screen Information

Users and the Gateway administrator should be aware of the following User-screen field information:

- The **Real RDB Name** and **RDB User ID** fields are shifted to uppercase when you enter them.
- Password field entries are never visible, even when being entered, and must be entered twice to ensure correctness.
- The case of the **Unix User ID** and **RDB Password** fields is preserved when you enter them.

Important: *This preservation of case is particularly important to note for the* ***RDB Password*** *field because passwords on application servers are often uppercase.*

Using the User Screen as informix

Figure 5-3 shows the screen that the Gateway administrator, or user with user ID **informix**, sees. You can use this screen to add an association between any UNIX user ID and *real_RDB_name* pair and any *RDB_user_ID* and *RDB_Password* pair to the Gateway lookup table. The *real_RDB_name* is the actual name of the application server, in contrast to the *alias_RDB_name*. The *real_RDB_name* and the *alias_RDB_name* are described in "Information About the Application Server" on page 3-12.

You can also use the **Find** option to search for specific records in the lookup table. Once you find a record, you can update it or delete it.

Figure 5-3
User Screen That User informix Sees

```
User:   Add-one   Many-add   Find   Exit
Add new user record

 ----------------------   Press Control-W for HELP   -------------------------



    Unix User ID          [                          ]
    Real RDB Name         [                          ]
    RDB User ID           [                          ]

    RDB Password          [                          ]
```

The information that the User screen stores in the **gwuser** file looks like Figure 5-4. In the illustration, the UNIX user **helen** uses two application servers. Therefore, she has two *RDB_user_ID*s, one on **SANJOSE** and one on **TUCSON**. When **helen** uses the **gwdba** utility to modify her entries, she must individually set a password for each *RDB_user_ID*.

Figure 5-4
Representation of the User Information in the gwuser File

UNIX User ID	RDB_name	RDB_user_ID	RDB_Password (Encrypted)
helen	SANJOSE	PAYROLL	encrypted password
helen	TUCSON	JACKSON	encrypted password
martin	SANJOSE	MWHITNEY	encrypted password
curt	SANJOSE	PAYROLL	encrypted password

Tip: *The UNIX user ID and the "RDB_name" form a unique key for entry into the User screen. The **gwdba** utility does not check the "RDB_user_ID" for uniqueness, so more than one UNIX user can use the same "RDB_user_ID" at any "RDB_name." For example, in Figure 5-4 both **curt** and **helen** map to the "RDB_user_ID" PAYROLL.*

Using the gwdba Utility as a Regular User

Regular users can only add to or update their own user entries with the **gwdba** utility. Figure 5-5 on page 5-10 shows the screen that regular users see. Users who have access to several different application servers can use the **Find** option to cycle through the entries.

Figure 5-5
User Screen for a Regular User

```
User:   Add-one   Many-add   Find   Exit
Add new user record
 --------------------   Press Control-W for HELP   ----------------------------


    Unix User ID          [ user-name               ]
    Real RDB Name         [                         ]
    RDB User ID           [                         ]

    RDB Password          [                         ]
```

For additional information, regular users should refer to "Basic User-Screen Information" on page 5-8.

Batch Mode User-ID Mapping Maintenance

The **gwdba** tool supports batch-mode user-ID mapping maintenance. The **gwdba** tool can update the **gwuser** file from an input text file. To exercise this feature, use the **-ua** option to add users, the **-um** option to modify users, and the **-ud** option to delete users. The table in Figure 5-6 summarizes the options.

Figure 5-6
gwdba User-ID Mapping Options

Command	Purpose
gwdba -ua *file name*	Adds users
gwdba -um *file name*	Modifies users
gwdba -ud *file name*	Deletes users

You must use the following conventions when you prepare the text file:

- The format for the text file is:

  unix_user_id RDB_name RDB_user_id RDB_password

  Where:

  - *unix_user_id* is the UNIX user ID
  - *RDB_name* is the relational database name
  - *RDB_user_id* is the relational database user ID
  - *RDB_password* is the relational database password
- The *RDB_name* and the *RDB_user_id* fields are shifted to uppercase regardless of their case in the text file. The *RDB_password* field is treated as case sensitive and is used unmodified.
- No other characters, such as punctuation marks, can occur between the fields in the preceding example. The only recognized white space characters are space and tab. The newline character indicates the end of a user record.
- For any user operation the two key fields, *unix_user_id* and *RDB_name*, are required to perform a lookup in the **gwuser** file. If any one of these fields is missing for a given user operation, an error message is logged in **/tmp/gwdba_err.log.** The user addition or modification file *must* contain all four fields for every user who needs to be added or modified. The user deletion file *must* contain at least the key fields.
- If the user already exists in the **gwuser** file, the user addition operation results in a `Duplicate Record` message. Otherwise it adds the user record to the **gwuser** file. The user modification operation modifies the record if it exists, or adds the record if it does not exist in the **gwuser** file. The user deletion operation does not require the *RDB_user_id* and *RDB_password*. If this information is present in the text file, it is ignored.

Logging Error Messages

All the error messages are logged into the file **/tmp/gwdba_err.log**.

Warning: *Since the error-message text files contain the password information in unencrypted, or clear text, these files must be either secured or deleted after use.*

Enforcing Security

When the Gateway begins execution at the request of a client application, it requires a user ID from the client, and sometimes a corresponding password. The Gateway then uses this user ID, and the password if required, to determine the user ID and password used to connect to the application server. When a client application accesses an application server with the Gateway, user ID authentication checks are made at the following times:

- When the client application connects to the Gateway
- When the Gateway connects to the application server

Connection Between a Client Application and the Gateway

The user ID that is sent to the Gateway from the client application is known as the *effective user ID* of the client. The effective user ID is defined as the following list describes:

- For a network connection, the effective user ID is the one the client application uses to connect to the Gateway.
- For a local connection, or unnamed pipes, the effective user ID is the one the client application uses to execute the Gateway.

The effective user ID of the client on a network connection comes from one of the following sources:

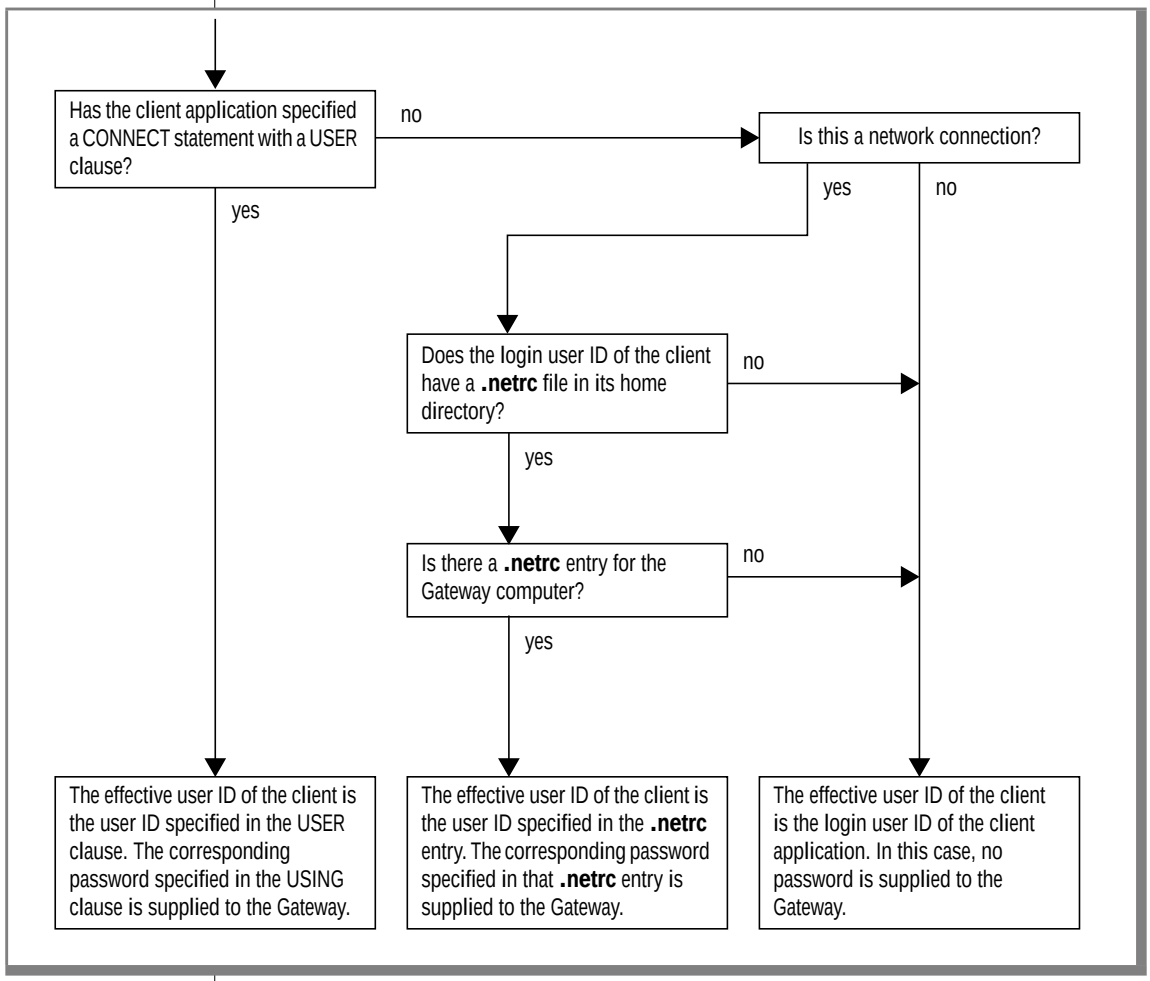
- The user ID that is specified in the USER clause of a CONNECT statement, as shown in the following example:

```
CONNECT TO 'alias_RDB_name@my_gw' USER 'P_GOMEZ'
       USING secret_password
```

- The UNIX login user ID of the client application
- A user ID from the **.netrc** file in the home directory of the login user ID of the client application

More specifically, the effective user ID of the client is determined using the steps that Figure 5-7 shows.

Figure 5-7
Determining the Effective User ID of the Client



Determining Whether You Need a Password

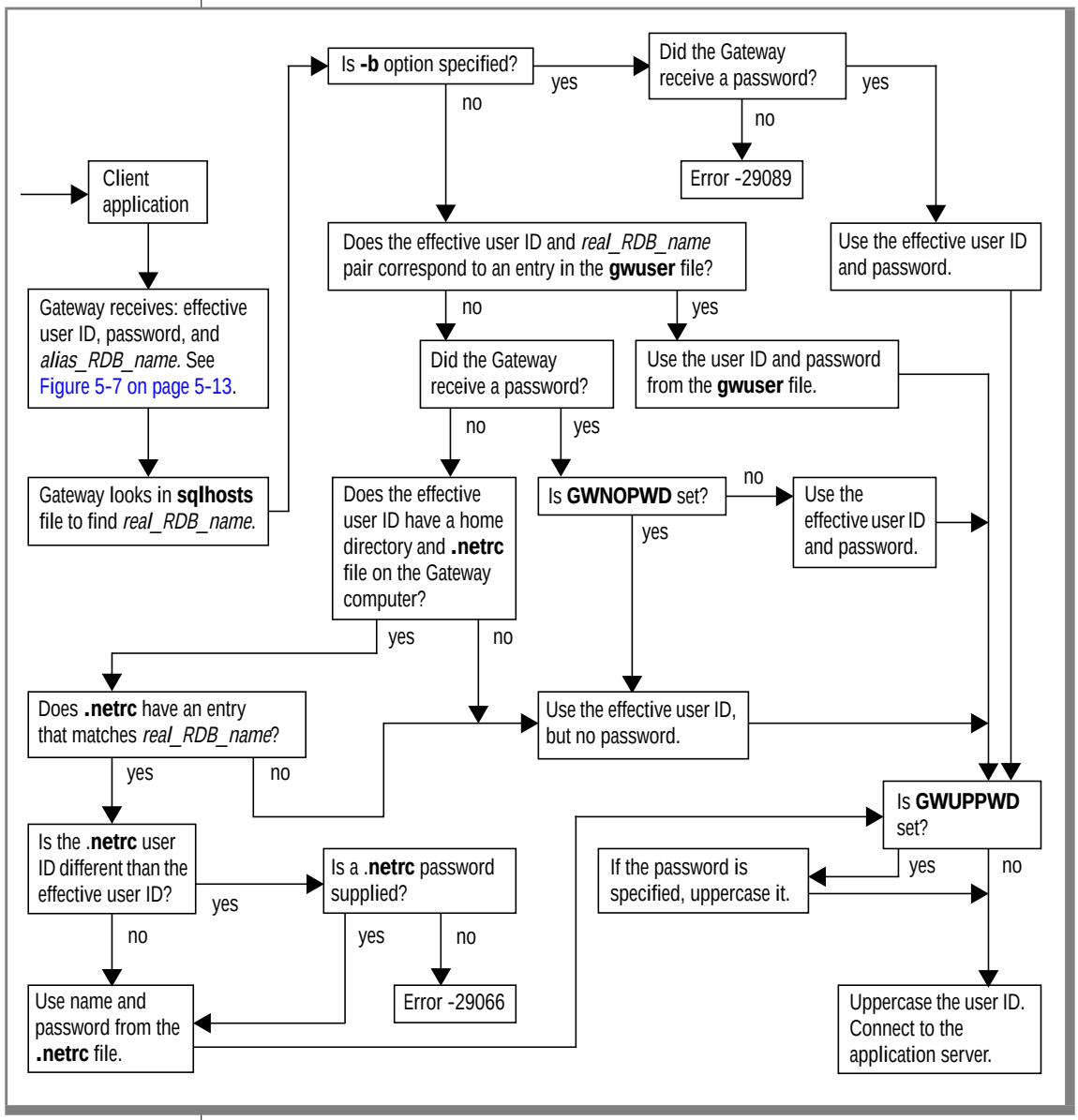
To see whether the client also supplies a password to the Gateway, see Figure 5-7 on page 5-13.

For more information about network security and the **.netrc** file, refer to your UNIX manuals or ask your network administrator.

Connection Between a Gateway and an Application Server

The user of the Gateway must have a valid user ID for the application server. This user ID is frequently not the same as the effective user ID of the Gateway user. The following discussion describes how the Gateway finds the user ID and password to use with the application server. Figure 5-8 on page 5-15 illustrates the steps used to find the user ID for the application server.

Figure 5-8
Finding the User ID for the Application Server



When the client application issues a DATABASE statement or a CONNECT statement, it specifies the *alias_RDB_name* of the application server. After a successful connection is made between the client application and the Gateway, the Gateway looks in the **hostname** field of the **sqlhosts** file to find the *real_RDB_name* that is associated with the *alias_RDB_name*. (Refer to "Preparing the sqlhosts File on the Gateway Computer" on page 3-16.) Then the Gateway uses the effective user ID and the *real_RDB_name* as a key into the **gwuser** file to find the user ID and password that should be passed to the application server. For information about the **gwuser** file, refer to "The User Screen" on page 5-7 and "Using the gwdba Utility as a Regular User" on page 5-9.

If no match exists for the effective user ID in the **gwuser** file and if a password was specified to connect to the Gateway, the effective user ID is uppercased and used with the password to connect to the application server.

The .netrc File

If no match exists for the effective user ID in the **gwuser** file and the client application does not specify a password, the Gateway searches for a **.netrc** file in the home directory of the effective user on the Gateway computer. The **.netrc** file is used only when the **gwdba** utility does not create a map and the user does not specify a password to connect to the Gateway.

If the effective user ID has a home directory on the Gateway computer and the directory has a **.netrc** file, the Gateway uses the *real_RDB_name* as a key into the **.netrc** file to find a user ID and password. If found, that user ID is uppercased and sent with the password (if present) to the application server. If the Gateway does not find a user ID and password in the **.netrc** file, the effective user ID is uppercased and then sent to the application server without a password.

If the Gateway daemon **gwd** was started with the **-b** option, the Gateway does not look up the **gwuser** file or the **.netrc** file. The client's effective user ID and password are used to connect to the application server. This user ID is not authenticated by **gwd**.

The Bind-Package Screen

Figure 5-9 shows the Bind-Package screen, which lets you (as user ID **informix**) set up the packages at each application server as part of the Gateway configuration process. A package is the control structure that is produced when SQL statements in an application are bound to a relational database in a DRDA-compliant server environment. The application server uses the control structure as it processes SQL statements. *Package binding* is the process of preparing the package. (Refer to "Packages and Sections in DRDA" on page 1-15.)

Warning: *You cannot use Version 7.31 Informix Enterprise Gateway with DRDA with packages that were bound at the application server with earlier versions of the Gateway. You must rebind old application server packages with the Version 7.31* ***gwdba*** *utility before you can use the Version 7.31 Gateway. You can use the* ***Load*** *and* ***Unload*** *options of the Bind-Package screen to transmit rebound package information from one Gateway node to another.*

Figure 5-9
The Bind-Package Screen

```
Bind-Package:   Package-(re)bind  Find  Drop  Load  Unload  Apl-to-pkg  Exit
(Re)Bind both CS/RR isolation level packages at the remote RDB

----------------------------Press CTRL-W for HELP --------------------------


Real RDB Name                          [                ]
Number Of Non-Cursor-Hold Sections     [      ]
Number Of Cursor-Hold Sections         [      ]
Collection ID                          [                ]
Package ID for isolation level CS      [                ]
       for isolation level RR          [                ]
Package Owner                          [      ]
Package Authorization Rule                    ]
Grant EXECUTE privilege to PUBLIC?     [      ]

Alias RDB Name                         [                ]
Gateway Server Name                    [                ]
```

For information about the fields of the Bind-Package screen, refer to the following pages:

- *real_RDB_ name*, page 3-12
- Number of non-cursor-hold sections, page 5-25
- Number of cursor-hold sections, page 5-26
- Collection ID, page 3-13
- Package ID, page 5-20
- Package owner, page 5-20
- Package authorization, page 5-21
- *alias_RDB_name*, page 3-19
- Gateway database server name, page 3-10

In an approximate description, the user **informix** must bind two types of general-purpose dummy packages when the Gateway is configured, one for Cursor Stability and one for Repeatable Read. All applications that use the Gateway use these packages repeatedly. This configuration is in contrast to the customary package management of DRDA-compliant database servers, in which each application is bound into its own, unique package. For more information about packages and sections, see "Packages and Sections in DRDA" on page 1-15.

The packages need to be set up only once for each application server. For more information, refer to "Using the Load and Unload Options" on page 5-28. You can use the same set of packages to gain access, provided the proper privileges are granted, to any database object.

Because the packages that the Gateway uses are bound only once for many applications, the administrator must enter the number of sections on the Bind-Package screen. Guidelines for how to estimate the number of required sections are given in "Calculating the Number of Required Sections" on page 5-25.

You do not need to understand package binds. Complete the Bind-Package screen, and the Gateway does the binding for you.

The **gwdba** utility stores Bind-Package information in the **$INFORMIXDIR/gw/sysinfo/gwbind** files. The table in Figure 5-10 shows the information stored by **gwdba** when you run the Bind-Package screen function. The first four columns of the table correspond to the first four lines of the Bind-Package screen. The **gwdba** utility generates the last three columns. For each *real_RDB_name* and *collection_ID* pair, two packages are bound: one for the Repeatable Read (RR) isolation level and one for the Cursor Stability (CS) isolation level. Both packages have the same number of sections. In addition, the **gwdba** utility generates consistency tokens that the Gateway uses internally.

Figure 5-10
Bind-Package Information Stored by gwdba

real_RDB_name	Number of Sections Non-Hold	Hold	Collection ID	Package Name	Isolation Level	Consistency Token (mmddhhmm)
SANJOSE	32	8	FOX	IFMXRR	RR	04081546
SANJOSE	32	8	FOX	IFMXCS	CS	04081546
NEWYORK	46	46	FOX	IFMXRR	RR	10310101
NEWYORK	46	46	FOX	IFMXCS	CS	10310101
TUCSON	32	12	EMPDB	IFMXRR	RR	02110945
TUCSON	32	12	EMPDB	IFMXCS	CS	02110945

Tip: *The **gwdba** utility adds 13 sections to the user-specified noncursor-hold section count. The Gateway uses these additional sections internally to issue static catalog-access queries and execute stored procedures.*

Bind-Package Features for the gwdba Utility

The **gwdba** utility supports the following Bind-Package features:

- Ability to name the package
- Ability to create and bind multiple packages
- Ability to specify the owner ID for a package at the time of binding
- Ability to *optionally* grant EXECUTE authority on a package to PUBLIC
- Support for controlling access to objects with packages

Naming the Package

The Bind-Package screen of the **gwdba** utility provides an option to explicitly name the packages. By default, the Gateway assumes IFMXCS and IFMXRR are the package IDs, respectively, for the two isolation levels supported: Cursor Stability and Repeatable Read.

Creating and Binding Multiple Packages

With the ability to name the package ID, the administrator can create and bind multiple packages for the same relational database. Note, however, that a single package should be uniquely identified by `RDBName.CollectionID.PackageID`. For more information on how to associate an application to a package, see "Using Multiple Packages and Multiple Applications" on page 5-22 .

Specifying the Owner ID for a Package at Bind Time

At bind time, the administrator can now specify the owner of a package that is different from the user ID that is used to bind the package. Note that the *owner* here should be a valid user name on the target relational database and any authentication and/or authorization of the user name is the responsibility of the target database system.

If the package being bound already exists at the target server, and the new owner specified is different than the previous package owner, then the following rules apply:

- The previous owner retains the EXECUTE privilege on the new package but loses the implicit privileges of ownership.
- All users who had been given EXECUTE privilege by the previous package owner retain the privilege, but the privileges are changed to indicate that they have been given by the new package owner.
- The new package owner is automatically granted the EXECUTE privilege on the new package and the implicit privileges of ownership.

  If the owner of the package is not changed, then all of the existing EXECUTE privileges on the existing package are retained unchanged by the new package.

Optionally Granting EXECUTE Authority on a Package to PUBLIC

For enhanced security, the administrator can control granting the EXECUTE authority on a package to PUBLIC. In versions of the Gateway prior to Version 7.3, EXECUTE authority on a package is automatically provided to PUBLIC. In Version 7.3 and later versions, the administrator can elect *not* to do so at the time of binding and instead, grant access to selected users by issuing SQL statements to the relational database through the DB-Access utility connected to the Gateway.

Using a Package to Control Access to Objects

Typically, the DB2 administrators control access to objects by:

- Explicitly granting and / or revoking privileges to users on objects
- Exercising privileges through a plan or a package

Versions of the Gateway prior to Version 7.3 allowed access to objects only by controlling access at the user level. In Version 7.3 and later versions of the Gateway, you can control a user's access to objects such as tables and views that are used in dynamic **sql** statements. This is done by specifying authorization rules for a package. The two rules that govern the authorization checking of a dynamic statement are defined as follows:

- RUN-TIME rules

  The DRDA server uses the authorization ID of the application process to check the authorization of the objects. Gateway is the *application process* and this is the only access control mechanism supported in versions of the Gateway prior to Version 7.3.

- BIND-TIME rules

  The DRDA Server uses the owner ID of the bound package to check for the authorization of objects. Version 7.3 and later versions of the Gateway support this access control mechanism.

For information about how your server supports these rules, check your DRDA server documentation. To support this new feature in the Gateway, the Bind-Package screen of the **gwdba** utility allows you to specify `B` for BIND-TIME rules, or `R` (the default) for RUN-TIME rules, for the package.

Using Multiple Packages and Multiple Applications

Because a relational database user may run more than one application, it is imperative that the user have execute permission on more than one package. However, at any given time, the administrator must be able to pick the single package that the application will *execute*. This is accomplished by considering the *alias-RDB-name* as the logical equivalent of an application and mapping the package that the *alias-RDB-name*, that is, the application will use. The **gwdba** utility provides this ability to associate the *alias-RDB-name* with the package for a given isolation level.

The **Apl-to-pkg** option of the Bind-Package screen allows you to maintain the application-to-package mapping entries. The utility stores this information in the **$INFORMIXDIR/gw/sysinfo/gwaplpkg** files. For example, the **Add** option of the *apl-to-pkg* screen has the following screen layout:

```
apl-to-pkg:    Add-one Many-add Find  Exit
Add a new apl-to-pkg mapping record
----------------------   Press Control-W for HELP   ----------------------
    Alias RDB Name    [                          ]
    Isolation Level   [                          ]

    Real RDB Name     [                          ]
    Collection ID     [                          ]
    Package ID        [                          ]
```

The examples in Figure 5-11 and Figure 5-12 assume that three applications, inventory, payroll, and sales, are accessed through the Gateway and that the administrator has mapped only the payroll and inventory applications to the respective packages. The **sqlhosts** entry could be as follows.

Figure 5-11
Example sqlhosts File

dbservername or alias_RDB_name	nettype	hostname	servicename
gwserver1	onsoctcp	valley	gw1
gwserver2	onsoctcp	valley	gw2
inventory	drapplu6	DB2CORP1	m=modeA
payroll	drapplu6	DB2CORP1	m=modeB
sales	drapplu6	DB2SALES	m=modeC

The Apl-to-pkg entries in **gwaplpkg** could be as shown in Figure 5-12 on page 5-24.

Figure 5-12
Example sqlhosts File

Alias-rdbname	Isolation Level	RDBname	Collection ID	Package ID
inventory	CS	DB2CORP1	INVENTORY	INV1
inventory	RR	DB2CORP1	INVENTORY	INV2
payroll	CS	DB2CORP1	PAYROLL	PAY1
payroll	RR	DB2CORP1	PAYROLL	PAY2

When the application connects to **inventory@gwserver1** and uses an isolation level of Repeatable Read, the Gateway looks up the map file, **gwaplpkg,** and retrieves the package name `DB2CORP1.INVENTORY.INV2`. The Gateway then uses this value to look up **gwbind** to get the package details.

Consider the scenario when the sales application connects using **sales@gwserver2** and sets an isolation level of Cursor Stability. Because no mapping exists for *alias_RDB_name* **sales**, the lookup in the map file **gwaplpkg** fails. The Gateway then sequentially scans the **gwbind** file for the matching *RDB_name* **DB2SALES** and *package ID* **IFMXCS** to obtain the default package details.

Required Privileges

The *RDB_user_ID* for the Gateway administrator, that is, a user with UNIX user ID **informix**, needs the following privileges to bind the packages on DB2 application servers:

- System privilege: BINDADD
- Package privilege: PACKADM on the collection

For the OS/400, the *RDB_user_ID* for the Gateway administrator needs the following privileges within the chosen collection to bind the packages:

- Object privilege: Opr
- Data privileges: Read and Add

If the application server is an SQL/DS server, Informix recommends that the Gateway administrator grant database administrator privileges to the *RDB_user_ID* to bind the packages.

Calculating the Number of Required Sections

The next sections discuss guidelines for calculating the number of non-cursor-hold sections and cursor-hold sections. The number of sections corresponds roughly with the number of statements that you expect to have active at one time. Do not make the number of sections too large because it wastes memory on the application server. If you run out of sections, you see the following error:

```
-29043 No more section_type sections left. Rebind packages
with more sections.
```

The *section _type* specifies either the cursor-hold or non-cursor-hold sections.

The Number of Noncursor-Hold Sections

All the applications that use a given installation share the Gateway packages. For this reason, the packages must contain enough sections to service the largest of these applications.

Use the following criteria to estimate the number of noncursor-hold sections that a given application requires.

1. Start with the maximum number of concurrently open noncursor-hold cursors that are declared on static SELECT statements.
2. Add one for each noncursor-hold cursor that is declared on dynamic SELECT statements.
3. Add one for each prepared statement that is not a dynamic SELECT statement.
4. Add one if the application contains any static noncursor SQL statements.

The Number of Cursor-Hold Sections

Use the following criteria to estimate the number of cursor-hold sections that a given application requires.

1. If the application server is a SQL/DS, the number of cursor-hold sections is zero. SQL/DS does not support cursor-hold cursors.
2. Start with the maximum number of concurrently open cursor-hold cursors that are declared on static SELECT statements.
3. Add one for each cursor-hold cursor that is declared on dynamic SELECT statements.

A More Exact Calculation of the Number of Sections

The number of sections in use rises and falls as you use an application because the Gateway acquires and releases sections in much the same way as an Informix database server acquires and releases resources associated with a cursor or statement. To more accurately calculate the maximum number of sections that a given application requires, you can trace the execution of the application.

For static SQL, apply the following rules:

- Each execution of a static non-cursor SQL statement acquires and then immediately releases a section.
- Each OPEN statement of a cursor that is declared on a static SELECT statement acquires a section. The section is released when the cursor closes.

For dynamic SQL, apply the following rules:

- Each PREPARE statement acquires a section.
- The first OPEN statement of each cursor that is declared on a dynamic SELECT statement acquires a section, except when the first OPEN of the first cursor that is declared on a given dynamic SELECT statement does not acquire a section. It uses the section that is acquired on the PREPARE statement.
- Each FREE of a statement that does not have a cursor declared on it releases a section. Also, each FREE of a cursor that is declared on a dynamic SELECT statement releases a section.

Additional Section for CALL Statement

Because the DRDA CALL statement for stored procedure invocation cannot be dynamically prepared, the **gwdba** utility internally adds one reserved section for executing stored procedures when binding packages. After **gwdba** completes the package binding, the following message appears on the bottom of the Bind-Package screen:

```
Packages bound. Added 13 reserved non-hold sections for
internal use.
```

Using the Find Option on the Bind-Package Screen

You can use the **Find** option on the Bind-Package Screen to search for records based on Real RDB Name, Collection ID, and Package ID.

The Real RDB Name, Collection ID and Package ID constitute a composite key for the bind package records. The **Find** option presents the bind-package records in key order. That is, for each Real RDB Name, records are ordered by Real RDB Name, then by Collection ID, and then by Package ID .

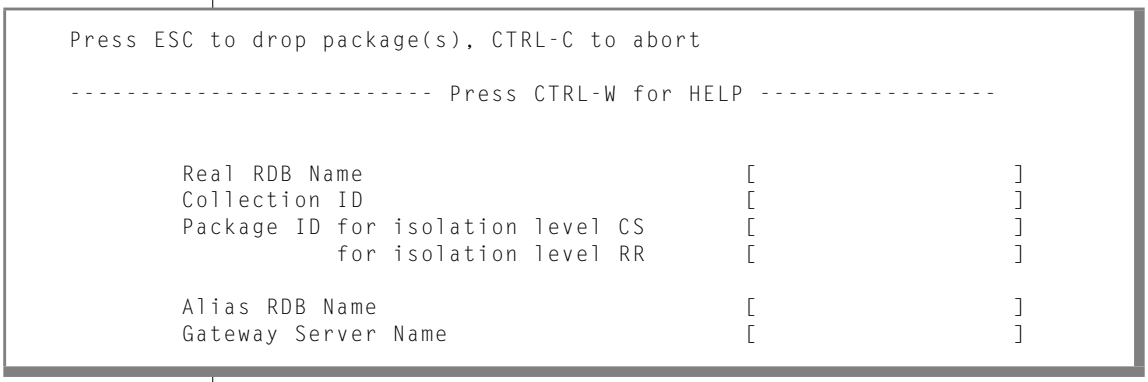
You can use only the Real RDB Name, Collection ID, and Package ID fields (individually) as search arguments for the **Find** option. Executing the **Find** option positions the screen at the first record (according to this defined order) that satisfies the search arguments that you enter.

If you have entered any two fields out of the three fields (Real RDB Name, Collection ID, Package ID), then the search is based on the Real RDB Name field if it is specified. If the Real RDB Name field is not specified, the Collection ID field is used for the search.

Dropping a Package

You use the **Drop Package** option of the Bind-Package screen to drop packages that are bound at an application server. Figure 5-13 on page 5-28 shows the Drop Package screen. To identify the packages to be dropped, specify the *real_RDB_name* of the application server and the *collection ID* where the packages exist. The *alias_RDB_name* and the Gateway server name (*gwservername*) specify which Gateway and application server to use when you process the drop. The *alias_RDB_name* must map to the *real_RDB_name* of the application server where the packages exist.

Figure 5-13
The Drop Package Screen



The **Drop Package** option deletes the specified packages at the application server and also deletes the information that the **gwbind** files store for the specified packages.

If the packages are dropped at the application server through some means other than the **gwdba** utility, delete the information that is stored in the **gwbind** files for the dropped packages.

To delete the information in the gwbind files

1. Choose **Find** from the Bind-Packages screen.
2. Press ESC to browse all the package records.
3. Choose **Delete** when you find the correct records.

   Two records need to be deleted, one for the Repeatable Read (RR) isolation level and one for the Cursor Stability (CS) isolation level.

Using the Load and Unload Options

In an installation with multiple Gateways, more than one Gateway can access an application server. In that case, only one Gateway needs to bind a package at the application server. All other Gateways that use the same application server, whether on other computers or in other **$INFORMIXDIR** directories on the same computer, must maintain identical package information in their respective **gwbind** files.

The **Unload** and **Load** options let you transfer bound-package data from one computer to another. When you choose the **Unload** option, the **gwdba** utility prompts for a filename into which it unloads the **gwbind** data. This file is a regular UNIX file that you can transfer to another computer. When you choose the **Load** option, the information from the specified file is loaded into the **gwbind** files.

During the loading process, if an entry that is to be loaded already exists in the target **gwbind** file (that is, the entries are for the same application server), the record is not loaded. You, as the Gateway administrator, must decide which record you want to keep. To overwrite an entry, you must delete it by using the **Delete** option and then perform the **Load** procedure again.

Warning: *When multiple Gateway installations access the same application server, either specify distinct collection IDs to create distinct packages or create the package once, unload the package-information entries, and load the entries at the subsequent Gateway installations. Do not specify the same collection ID to bind the package more than once per application server. Each bind package invalidates the previous Gateway bind package.*

Access to Nonjournaled OS/400 Files

If OS/400 files are journaled, both read and write operations are allowed with Repeatable Read and Cursor Stability isolation levels. If AS400 files are *not* journaled, the following considerations apply:

- If the isolation level is Repeatable Read, only read access is allowed.
- If the isolation level is Cursor Stability, access to the files depends on the setting of the **GWMAPCS** environment variable when the packages were bound:
  - If **GWMAPCS** is not set, or if it is set to an invalid value, the package is bound with Cursor Stability. In this case, neither read nor write access is possible, and AS400 returns error -7008.
  - If **GWMAPCS** is set to `1`, Cursor Stability maps to Uncommitted Read (AS400 `*CHG` values) when the package is bound. Read access is permitted, but write access is not permitted.
  - If **GWMAPCS** is set to `2`, Cursor Stability maps to the AS400 `*NONE` value when the package is bound. Both read and write access are permitted.

The Gateway supports only two isolation levels, Repeatable Read and Cursor Stability, by means of two packages named IFMXRR and IFMXCS. While binding the package IFMXCS, the Gateway maps the isolation level of the package to the AS400 COMMIT(*CS), COMMIT(*CHG), or COMMIT(*NONE), depending on whether **GWMAPCS** is not set, is set to 1, or is set to 2, respectively.

When a user sets the isolation level to Cursor Stability, the package selected is IFMXCS, although the actual package isolation level depends on **GWMAPCS** when the package is bound. Therefore, be aware of OS400 behavior with these commitment control options.

To change isolation-level mapping, reset the **GWMAPCS** environment variable and rebind the packages. (You also can set **GWMAPCS** in the Informix environment-configuration files.)

The Catalog Menu

The options on the **Catalog** menu let you create Informix system catalog tables on the application server. The **Catalog** menu shown in Figure 5-14 has the following options:

- **Install**
- **Add-tables**
- **Refresh-tables**
- **Purge-tables**
- **De-install**

The primary reasons for installing an Informix catalog are to improve the performance of distributed queries and to enable client products that require an Informix catalog to work with the Gateway.

Figure 5-14
The Catalog Menu

```
Informix Catalog:  Install  Add-tables  Refresh-tables  Purge-tables De-install exit
Install Informix Catalog on an Application Server

------------------------------ Press CTRL-W for HELP ----------------------
```

To prepare the Informix catalog

1. Execute the **Install** option to create the Informix catalog.

   The **Install** option creates the tables and other objects that make up the Informix catalog on the application server.

2. Prepare a file that lists application-server tables.

   You must list each table that should be included in the Informix catalog in a file that will be used by the **Add-tables** option. For instructions on how to prepare the file, refer to "Files for Add-tables and Purge-tables" on page 5-34.

3. Run the **Add-tables** option to add the information to the Informix catalog.

   The **Add-tables** option adds information about the tables in your list to the Informix catalog. This step is called *populating* the Informix catalog.

After you have created and populated the catalog for the first time, you must periodically run the **Refresh-tables** option to pick up changes in table and index information. For example, if the number of rows in a table changes significantly, or if the column definitions of a table changes, you should run the **Refresh-tables** option. You must keep the information in the catalog current because some client products use the catalog to get table descriptions and Dynamic Server coordinators use the catalog to optimize the performance of distributed queries.

After you create and populate the catalog, you can run the **Add-tables** option again to add information about other tables to the catalog, or you can run the **Purge-tables** option to delete information about existing tables (presumably tables that are no longer used) from the catalog.

Using the Catalog Options

Before you use the **Catalog** options, you must take the following steps:

- Make sure that you have the proper privileges on the application servers.
- Create objects on the application servers that can hold the catalog.
- Prepare a file of table names for some options.

Permissions That Are Required for Catalog Installation on DB2

On DB2, to execute the **Add-tables**, **Refresh-tables**, or **Purge-tables** option, the user ID that is associated with user **informix** must have the following authority:

- DBADM authority on database INFORMIX
- SELECT privilege on the following DB2 system catalog tables:
  - **sysibm.systables**
  - **sysibm.syscolumns**
  - **sysibm.sysindexes**
  - **sysibm.syskeys**
  - **sysibm.sysviews**
  - **sysibm.sysviewdep**
  - **sysibm.syssynonyms**

To execute the **Install** or **De-install** options, the user ID to which user **informix** maps must be **INFORMIX** (uppercase is required) and possess the privileges in the previous list or must be any user ID with **sysadm** authority.

Permissions Required for Catalog Installation on OS/400

On OS/400, to execute the **Add-tables**, **Refresh-tables**, or **Purge-tables** options, the user ID that is associated with user **informix** must be able to perform the following functions:

- Select, insert, and delete from all files in library **INFORMIX**.
- Select from **qsys.qadbxfil** and **qsys.qadbldep**.
- Select from each table or view that is added to the Informix catalog, if the table or view is in a library instead of a collection.

- For OS/400 versions prior to 3.1, select from the following collection catalog tables for each table or view that is added to the Informix catalog, if that table or view is in a collection:
  - ***collection-name*.syscolumns**
  - ***collection-name*.sysindexes**
  - ***collection-name*.syskeys**
  - ***collection-name*.sysviews**
- For OS/400 version 3.1 and above, the select privilege is required on the above catalog tables in collection **qsys2**.

To execute the **Install** or **De-install** options, the user ID to which user **informix** maps must be able to create and drop tables and indexes in library **INFORMIX**.

When a view is added to the Informix catalog with the **Add-tables** option, all tables or views on which it depends are also added. The user ID to which user **informix** maps must have the ability to select from these objects, if they reside in a library, or from the catalog of their collection, if they reside in a collection.

Using Informix Catalog Options on OS/400

On OS/400, make sure that the library **INFORMIX** is a noncollection library. Otherwise, the Informix catalog **Install** option will fail with OS/400 **sqlcode** -601 (object already exists). This failure occurs because some of the catalog table names for the collection are the same as the Informix catalog table names.

On OS/400, remember to enable journaling in the library **INFORMIX** before you execute the **Install** option. Otherwise, you will encounter the OS/400 **sqlcode** -7008 (Invalid operation) during installation of the Informix catalog.

Objects Required for Catalog Installation

You must create the following objects on the application server before you install the Informix catalog:

- A database named **INFORMIX** on DB2
- A noncollection library named **INFORMIX** on OS/400
- A public dbspace named **INFORMIX** on SQL/DS

Files for Add-tables and Purge-tables

Before you use the **Add-tables** or **Purge-tables** options, you must prepare a file that lists the names of tables or views that will be added or purged. The format of table and view names in the file that the **Add-tables** and **Purge-tables** options use must be *owner.objname*. The names must observe the following restrictions:

- The *owner* must be present.
- The *owner* and *objname* cannot be delimited identifiers.
- No spaces are allowed in *owner.objname*.
- Table and view names are separated by blanks, tabs, or newline characters.
- Aliases are not supported on DB2 or SQL/DS.

When the **Add-tables** or **Purge-tables** option asks for the filename, you must give the pathname of the file where the names are stored.

Case Leveling in Table Names

In accordance with Informix case-leveling standards, the Gateway converts lowercase owner names to uppercase, and table and column names to lowercase when it inserts the names into the Informix catalog. Similarly, the Gateway converts column names to lowercase if they are returned to the application via the DESCRIBE statement.

The Install Option

The **Install** option installs the Informix catalog on the application server. It creates tables and views that make up the Informix catalog on an application server. The **Catalog Install** menu, shown in Figure 5-15, lists the SQL script files available for installation at the application server.

Figure 5-15
The Catalog Install Menu

```
Catalog Install >>
Choose an install file with the Arrow Keys, or enter a name, then press Return.

--------------------------------------------------------------------------------

db2mvs
sqlds
os400
sqlds
upgrade
```

Select a file from those listed on the screen or type in the filename at the prompt and press RETURN. The **.sql** filename extension is not displayed.

Files listed on the **Catalog Install** menu are contained in the Gateway directory **$INFORMIXDIR/gw/catalog/install**. Each filename indicates the target DBMS where the catalog is to be installed and might also indicate the release level for the target DBMS. The **upgrade.sql** script is the upgrade script for all target DBMS. You need to run this script only if you already installed the Informix catalog on the target DBMS by using a Gateway version prior to Version 7.31.

The database administrator can alter these scripts. When there is no SQL script file for an application server, the Gateway administrator can add a DDL file to the directory for the Gateway. By examining the DDL in the supplied files, the Gateway administrator can write a suitable installation script for another data source.

Before you can execute the **Install** option, the application server database administrator must create the following objects at the application server:

- A database named **INFORMIX** for DB2
- A noncollection library named **INFORMIX** for OS/400
- A public dbspace named **INFORMIX** on SQL/DS

Enabling Journaling for the informix Library on OS/400

If the application server is an OS/400, you must enable journaling in the non-collection library **INFORMIX** before you execute the **Install** option. You can enable journaling for all objects in the library by creating a journal with the name **QSQJRN** in the library **INFORMIX**.

The OS/400 administrator can issue the following commands to enable journaling on an OS/400 application server:

```
CRTJRNRCV JRNRCV(INFORMIX/QSQJRN0001)
    TEXT("JRN RCVR FOR INFORMIX")

CRTJRN JRN(INFORMIX/QSQJRN) JRNRCV(INFORMIX/QSQJRN0001)
    TEXT("JRN FILE FOR INFORMIX")
```

The OS/400 administrator should ensure that the OS/400 *RDB_user_ID* associated with the UNIX user **informix** has the appropriate privileges on the journal receiver and the journal file in the library **INFORMIX**.

The Add-tables Option

The **Add-tables** option adds tables and views to an already-present Informix catalog. It requests a filename and then populates the Informix catalog on the application server with the tables and views listed in that file. The **Add-tables** option also adds any tables or views on which the tables and views in the file depend.

Executing the **Add-tables** option automatically refreshes the Informix catalog, so that the added tables and views are built on a consistent view of the object definitions of the application server.

The Refresh-tables Option

The **Refresh-tables** option updates the tables and views in the Informix catalog. Because the Informix catalog is implemented as a set of tables, rather than as a set of views on the system catalog of the application server, the table and view information in the Informix catalog can become outdated. You must refresh the Informix catalog if any of its tables or views are altered or dropped.

When the **Refresh-tables** option is executed, any table or view that was dropped is purged from the Informix catalog. Any object that depends on such a dropped table or view is also purged. Any table that was altered, or dropped and re-created, is purged and then added again to the Informix catalog, as is any object that depends on such a table or view. The **Refresh-tables** option also purges and adds *all* indexes again.

The OS/400 provides no time stamping or versioning for tables and views. However, instead of purging and re-adding *all* tables and views, the **Refresh-tables** option on OS/400 removes only dropped tables and views. In other words, if an OS/400 table or view is dropped and re-created with different attributes, the Gateway administrator must execute the **Purge-tables** and **Add-tables** options on this object to refresh its definition in the Informix catalog.

Another reason to execute the **Refresh-tables** option is to update the table statistics in the Informix **systables** system catalog table. On DB2 and SQL/DS, the **Refresh-table** option purges and adds all **systables** rows again, so that the columns that contain the row count and number of pages are updated. As noted earlier, the index statistics are also updated.

The Purge-tables Option

The **Purge-tables** option requests a filename and then removes the tables and views that are listed in that file from the Informix catalog on the application server. The **Purge-tables** option also removes from the Informix catalog any objects that depend on tables and views in the file. The format of the file that the **Purge-tables** option uses is described in "Files for Add-tables and Purge-tables" on page 5-34.

The De-install Option

The **De-install** option drops the tables and views that make up the Informix catalog on the application server, which destroys both the Informix catalog and the information that it contains. The **Catalog De-install** menu, shown in Figure 5-16, lists the available SQL script files for deinstallation at the application server.

Figure 5-16
The Catalog De-install Menu

```
Catalog De-install >>
Choose a de-install file with the Arrow Keys or enter a name then press Return.

---------------------------- Press CTRL-W for HELP ------------------------

generic-din.sql
os400.sql
```

Select a file from those listed on the screen or type in the filename at the prompt and press RETURN.

Files listed on the **Catalog De-install** menu are contained in the Gateway directory **$INFORMIXDIR/gw/catalog/deinstall**. Each filename indicates the target DBMS where the catalog is to be deinstalled and might also indicate the release level.

The database administrator can alter these scripts as needed. When no SQL script file exists for an application server, the Gateway administrator can add a DDL file to the directory for the Gateway. By examining the DDL in the supplied files, the Gateway administrator can write a suitable deinstallation script for another data source.

Entering Table Names with Multibyte Characters into the Informix Catalog

To enter table names that contain multibyte characters into the Informix catalog, set the **GWDBALOCALE** environment variable to the Informix locale of the client before you invoke **gwdba**. Setting **GWDBALOCALE** allows the **gwdba** Informix catalog options to use locale-sensitive string manipulation functions.

In addition, in the conversion of any application server code set to client code set, the character data length expansion factor for that specific application server to client is applied to the lengths of character and graphic columns as stored in the Informix catalog.

When a client uses a multibyte code set, character data can expand in length when it is converted from the code set of the application server into the code set of the client. Typically, this applies only to Asian-language users. To ensure that the Gateway can return character data even when it expands in length, the Gateway calculates expansion factors that are applied to the byte length of the application server for every CHAR or VARCHAR field in the output **sqlda** that is returned to the client for a DESCRIBE or PREPARE statement. The resulting *expanded* lengths that are reported for character-output fields depend on the specific code-set conversion that will be performed on the fields when the data is moved from the application server to the client.

In simpler terms, the Gateway misrepresents to the client that CHARACTER or VARCHAR columns are larger than they are at the application server so that enough room exists for the character data to expand when converted from the code set of the application server to the code set of the client. In the same manner, when the description of a character column on the application server is placed in the Informix catalog, the column length is expanded based on the particular application server to client code set conversion in effect when the **gwdba Add-tables** option is run.

The Schema Menu

You use the **Schema** menu options to install the X/Open-style information schema on a DB2 or SQL/DS application server. Installing the X/Open-style information schema is not required by the Gateway. However, if you want to run client applications that require the X/Open-style information schema to function, you must use this option to install the schema. Currently no client applications for the Gateway use the X/Open-style information schema.

The X/Open-style information schema is a catalog structure, based on X/Open specifications, which contains information about objects in a database. Client applications that use this X/Open catalog to access information about database objects can be executed against any database server that has implemented these X/Open catalogs without rewriting the catalog queries that are used in the application.

The X/Open-style information schema, as installed by the Gateway, is implemented as a set of views based on the system catalog tables of the application server. This set of views is based on, although it deviates slightly from, the X/Open specification. If you plan to write a client application to use the X/Open-style information schema, read the installation scripts in the **$INFORMIXDIR/gw/schminfo** directory to understand how the views implement the X/Open-style information schema.

Required Permissions for Schema Installation

Because of the complexity of the objects that are created and the sensitivity of the system catalog tables on which they are based, you must have system database administration authority at the application server to execute the **Schema** options. The following table shows the required permissions.

Application Server	Permissions Required
DB2	SYSADM
SQL/DS	DBA SELECT WITH GRANT OPTION on 5 catalog tables

Informix recommends that the database administrator of the application server carefully review the scripts that are executed by this option before granting the permissions. The scripts that the schema installation uses are located in **$INFORMIXDIR/gw/schminfo**. Once the required **Schema** option is executed, system database or administration authority at the application server is no longer required.

Required Objects for Installing the Schema

Before you can use the **Schema Install** option, the administrator of the application server must create objects as the following list describes:

- On a DB2 server, create database **INFORMIX**.
- On an SQL/DS server, create public dbspace **INFORMIX**.

Schema Menu Options

The **Schema** menu, which Figure 5-17 shows, has options to install, deinstall, and verify the schema-definition information.

Figure 5-17
The Schema Menu

You must provide the *alias_RDB_name* and the Gateway server name (*gwservername*) so that the **gwdba** utility can connect to the application server to access and modify schema-definition information.

The **gwdba** utility creates the schema information objects with the special qualifier (authorization ID) **INFORMIX**. The Gateway changes owner-name **INFORMATION_SCHEMA** to **INFORMIX** in SQL statements it sends to application servers. The following example shows a statement before the Gateway changes it:

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
```

The following example shows the statement after the Gateway changes it:

```
SELECT TABLE_NAME FROM INFORMIX.TABLES
```

The Install Option

The **Install** option creates schema-information objects at the application server.

The De-Install Option

The **De-install** option drops all of the schema-information objects.

The Verify Option

The **Verify** option checks for the *existence* of the schema-information objects. It does *not* check for the integrity of data on these objects.

The Test-connect Option

Use the **Test-connect** menus and screens to verify that the connection to the application server is working. This feature eliminates the need to test the connection with a client SQL API. You do not have to be user **informix** to use this option.

Choose the **Test-connect** option from the **gwdba** main menu. The **Test-connect** option calls up the Connection screen, shown in Figure 5-18 on page 5-43, which prompts you to supply the *alias_RDB_name* and the Gateway server name to use for the connection. The Gateway then connects to the application server.

Once you are connected to the application server, the **Test-connect** menu appears, as shown in Figure 5-19.
The **Test-connect** menu has two options: **Select-count-test** and **Exit**. To test the connection, choose the **Select-count-test** option. The **Select-count-test** screen appears, as shown in Figure 5-20 on page 5-44.

Figure 5-18
The Connection Screen

```
Press ESC to continue processing, CTRL-C to abort

 --------------------   Press CTRL-W for HELP   ----------------------------

    Alias RDB Name                  [                        ]
    Gateway Server Name             [                        ]


Remote Alias RDB Name
```

Figure 5-19
The Test-connect Menu

```
Test-connect: Select-count-test     Exit
Test-connect to Application Server using SQL
 --------------------   Press CTRL-W for HELP   ----------------------------









Connected to Application Server. Choose Select-count-test for further test.
```

The Select-count-test screen prompts you for the owner and name of an existing table or view and issues a SELECT COUNT(*) FROM *owner.objectname* statement. You can specify a delimited owner and / or object name with the double quote (") delimiter.

Important: *The* ***Select-count-test*** *option does not work if the packages are not already bound at the application server with the* ***gwdba*** *utility.*

Figure 5-20
The Select-count-test Screen

```
Press ESC to continue processing, CTRL-C to abort

 --------------------   Press CTRL-W for HELP   ---------------------------

You are connected to the Application Server.

Please complete the following SQL statement by specifying the
owner and object name of a table or view on the application server.
This SQL statement will be executed to further test the connection.


SELECT COUNT(*) FROM [               ] . [                   ]
                           Owner               Object Name



Optional owner name (including quotes) will be used unmodified.
```

Executing Catalog Options in Batch Mode

You can use command-line options to invoke the **Catalog** and **Test-connect** options of the **gwdba** utility in batch mode. The format of the **gwdba** batch commands is as follows:

- *Keys* you enter interactively follow the hyphen (-).

  Keys represent the keystrokes that you need to access the relevant menu option. For instance, `cr` represents the **Catalog Refresh-tables** option.

- *Parameters* or *fields* you enter interactively follow the keys.

For example, to execute the **Add-tables** option of the **Catalog** menu in batch mode, issue the following command:

```
$INFORMIXDIR/bin/gwdba -ca db2_1 gw_1 /work/table_list
```

The preceding command executes the **Catalog Add-tables** (**-ca**) option at *alias_RDB_name* **db2_1**, using the Gateway server **gw_1**, and adds to the Informix catalog the tables and views that are listed in the file **/work/table_list**.

To install the Informix catalog from a **.sql** script file in batch mode, issue a command such as the following example:

```
$INFORMIXDIR/bin/gwdba -ci db2_1 gw_2 sqlds.sql
```

The name of the SQL script file for installing or deinstalling the Informix catalog must be an absolute pathname.

To execute the **Select-count-test** option, issue the following sample command:

```
$INFORMIXDIR/bin/gwdba -ts db2_1 gw_1 tab1
```

The screen displays the number of rows in the specified table or view or the resulting error. When you execute the **Test-connect** option, the screen displays either `Connection successful` or `Connection failed`.

You can use command-line options to invoke the following **Catalog** and **Test-connect** options in batch mode.

Option	Command-Line Keys and Parameters
Install	-ci <*alias_RDB_name*> <*Gateway_server_name*> <*SQL_filename*>
Add-tables	-ca <*alias_RDB_name*> <*Gateway_server_name*> <*filename*>
Refresh-tables	-cr <*alias_RDB_name*> <*Gateway_server_name*>
Purge-tables	-cp *alias_RDB_name*> <*Gateway_server_name*> <*filename*>
De-install	-cd <*alias_RDB_name*> <*Gateway_server_name*> <*SQL_filename*>
Test-connect	-t <*alias_RDB_name*> <*Gateway_server_name*>
Select-count-test	-ts <*alias_RDB_name*> <*Gateway_server_name*> [<*owner*>] <*tabname*>

Chapter

6

Programming with the Gateway

In This Chapter

The information in this chapter gives guidelines for how to prepare Informix client applications for use with an application server you access through Informix Enterprise Gateway with DRDA.

This chapter assumes that you are using the *Informix Guide to SQL: Reference* as your primary reference for Informix tools from Version 5.x. For Version 6.0 and higher Informix tools, the command syntax is in a separate volume: the *Informix Guide to SQL: Syntax*. The discussions in this chapter are intended to supplement the information from those books. In addition to the Informix syntax guides, you also need SQL reference material for the application server that you are accessing. For additional information on Global Language Support (GLS), see the *Informix Guide to GLS Functionality*.

In general, you can expect an SQL statement that conforms to ANSI SQL to function correctly with a DRDA-compliant application server when used in the Gateway environment. In this context, when we refer to an *application server*, we mean a remote database server that implements the DRDA protocols.

Informix has added some extensions to ANSI SQL statements as well as some new statements that are themselves extensions to ANSI SQL. The application server might not accept SQL statements that use those Informix extensions. SQL statements that include Informix extensions to standard ANSI SQL are marked in the *Informix Guide to SQL: Syntax* with the following special symbol: 

Because SQL statements are ultimately processed by the application server, the only way to be certain that a statement will work correctly is to refer to the documentation for that specific application server. This chapter provides general guidelines about how to use SQL statements to address an application server and specific guidelines for how to use IBM DB2, IBM SQL/DS, and IBM OS/400 application servers.

Handling SQL Statements in the Gateway Environment

An Informix client application relates to Informix Enterprise Gateway with DRDA as if the Gateway were an Informix database server. However, the DRDA database servers that the Gateway accesses are different from Informix database servers. When you write the application code, you must use syntax that is acceptable to the application server.

Although you do not need to know how SQL statements are processed to write an application, it might be helpful to understand how SQL statements are processed in the Gateway environment. The three steps for processing a statement are:

1. Processing by the client
2. Processing by the Gateway
3. Processing by the application server

Figure 6-1 shows the client, the Gateway, and the application server.

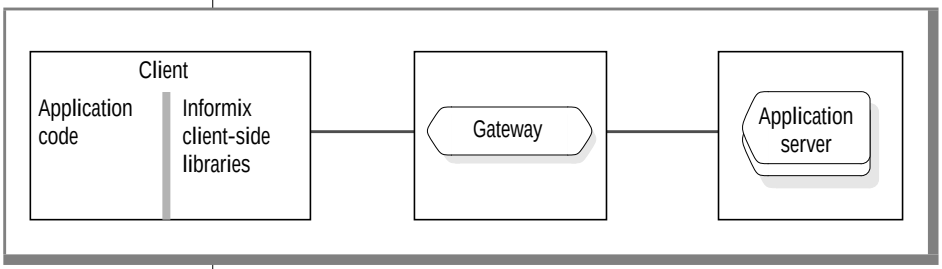


Figure 6-1 *The Three Pieces of a Gateway Environment*

Processing by the Client

The behavior of the client application is not influenced by the fact that it is attached to a Gateway. At compile time, the ESQL client checks static SQL statements for correct Informix SQL syntax. If a statement does not comply with Informix SQL syntax, the compile fails.

The ESQL client software does not check dynamic SQL statements for correct Informix SQL syntax either at compile time or at runtime. Dynamic SQL statements are sent unchanged to the Gateway, which sends them to the application server. (The Gateway can modify the statements, as described in "Processing by the Gateway.")

Because of this client processing, you must apply the following rules when you code applications that send SQL statements to an application server:

- If SQL statements conform to Informix SQL syntax, code them as either static or dynamic statements and send them to the application server.
- If SQL statements do not conform to Informix SQL syntax, code them as dynamic statements and send them to the application server.

You can write dynamic SQL statements in two different ways. The first way to write dynamic SQL statements is:

```
EXEC SQL PREPARE s1 FROM 'SELECT * FROM mytable';
```

The second way to write dynamic SQL statements is:

```
sprintf (stmt, "SELECT * FROM mytable");
EXEC SQL PREPARE s1 FROM :stmt;
```

Informix recommends the second method for IBM-specific syntax. When you use the first method, ESQL might check the syntax and return an error for the non-Informix syntax.

Similarly, you can write the EXECUTE IMMEDIATE statement in two ways. The first way to write the EXECUTE IMMEDIATE statement is:

```
EXEC SQL EXECUTE IMMEDIATE 'DROP INDEX my_temp_index';
```

The second way to write the EXECUTE IMMEDIATE statement is:

```
sprintf (stmt, "DROP INDEX my_temp_index");
EXEC SQL EXECUTE IMMEDIATE :stmt;
```

Again, Informix recommends the second method for IBM-specific syntax.

Processing by the Gateway

Statements that the Gateway receives for processing are examined according to Informix SQL syntax. If the statement does not comply with Informix SQL syntax, then the statement is sent unchanged to the application server. (This allows an application to use application server-specific SQL syntax.)

If the statement complies with Informix SQL syntax, it is subject to further processing by the Gateway. This processing occurs for both static and dynamic statements that are received from the client.

Tip: *Although this processing generally applies only to data-manipulation language (DML) statements, it also applies to data-definition language (DDL) statements for OS/400 prior to Version 3.1.*

The following list describes the additional process:

- The Gateway recognizes the limits of the target application server and enforces limits on the lengths of authorization, collection, table, view, index, and column names. If a name is too long, the Gateway truncates the name before passing it to the application server.
- The Gateway changes four-part table names to conform to the two-part table-name syntax of the application server.
- The Gateway processes some statements in cooperation with either the client application or the application server. For example, the Gateway provides the temporary storage and management for scrolling cursors.
- String literals in SQL statements delimited with double quotation marks are translated and delimited with single quotation marks.
- Where applicable, the MATCHES keyword is mapped to the LIKE keyword.
- Application servers that follow the DRDA protocol can handle only one SQL statement in each dynamic-management statement (EXECUTE IMMEDIATE or PREPARE and EXECUTE). If the Gateway detects that more than one SQL statement is included in a dynamic-management statement, the statement is rejected.
- The Gateway performs code-set conversions. For information about code-set conversions, refer to the *Informix Guide to GLS Functionality*.

Processing by the Application Server

All SQL statements are sent to the application server as dynamic SQL statements. The application server processes each statement and sends the results back to the Gateway. To the application server, the Gateway appears as a DRDA application requestor.

Summary of SQL Statement Behavior with the Gateway

The table in Figure 6-2 on page 6-12 summarizes the behavior of Informix SQL statements when you use the Gateway. This table is intended to give you a quick idea about which Informix SQL statements work the same with the Gateway as they do with Informix servers and which Informix SQL statements might or might not work the same, depending on the support for the SQL statement.

The table in Figure 6-2 has five columns, which the following list describes:

1. **SQL Statement**

   The SQL statements are listed in alphabetical order.

2. **Works**

   Statements checked in this column work the same with the Gateway as with Informix database servers.

3. **Works with Restrictions**

   Statements checked in this column work almost the same with the Gateway as they do with Informix database servers.

   The cases where the statement does not work are usually one of the following situations:

   - The statement includes non-ANSI Informix extension(s) that are not acceptable to the application server.
   - The statement syntax is acceptable, but the results that an application server returns are slightly different from the results that an Informix server returns.

4. **Application-Server Dependent**

   Statements checked in this column are SQL statements that function according to the documentation of the application server.

5. **Refer To**

   This column references discussions of specific SQL statements.

Important: *Treat any SQL statement other than those listed in Figure 6.2 as application-server dependent when you use them with the Gateway.*

In general, statements checked in the **Works** column or the **Works with Restrictions** column are SQL statements of the following types:

- The statement is processed completely by the client application or the Gateway or both. The application server is not involved. Examples of such statements include ALLOCATE DESCRIPTOR and SET ISOLATION.
- The statement is processed in cooperation with the application server. The DRDA commands that are sent to the application server to process the statement are constructed by the Gateway and normally should not fail. Examples of such statements include COMMIT, CLOSE *cursor id*, and the statements that are used for insert-cursor and scroll-cursor support.

Statements marked with an asterisk (*) have two columns checked intentionally.

Figure 6-2
Summary of SQL Statement Syntax and Semantics for the Gateway

SQL Statement	Works	Works with Restrictions	Application-Server Dependent	Refer To
ALLOCATE DESCRIPTOR	✓			
ALTER INDEX			✓	
ALTER FRAGMENT			✓	
ALTER TABLE			✓	

(1 of 6)

SQL Statement	Works	Works with Restrictions	Application-Server Dependent	Refer To
BEGIN WORK	✓			
CLOSE *cursor id*	✓			
CLOSE DATABASE	✓			
COMMIT WORK	✓			
CONNECT	✓			page 6-40
CREATE DATABASE			✓	
CREATE INDEX			✓	page 6-25
CREATE PROCEDURE			✓	
CREATE PROCEDURE FROM			✓	
CREATE ROLE			✓	
CREATE SCHEMA			✓	
CREATE SYNONYM			✓	
CREATE TABLE			✓	page 6-25
CREATE TRIGGER			✓	
CREATE VIEW			✓	
DATABASE		✓		page 6-41
DEALLOCATE DESCRIPTOR	✓			
DECLARE *		✓	✓	page 6-26
DELETE			✓	

(2 of 6)

SQL Statement	Works	Works with Restrictions	Application-Server Dependent	Refer To
DESCRIBE	✓			page 6-27
DISCONNECT	✓			
DROP DATABASE			✓	
DROP INDEX			✓	
DROP PROCEDURE			✓	
DROP ROLE			✓	
DROP SYNONYM			✓	
DROP TABLE			✓	
DROP TRIGGER			✓	
DROP VIEW			✓	
EXECUTE	✓			
EXECUTE IMMEDIATE		✓		page 6-27
EXECUTE PROCEDURE		✓		page 6-32
FETCH	✓			
FLUSH	✓			
FREE	✓			
GET DESCRIPTOR	✓			
GET DIAGNOSTICS	✓			
GRANT			✓	

(3 of 6)

SQL Statement	Works	Works with Restrictions	Application-Server Dependent	Refer To
GRANT FRAGMENT			✓	
INFO			✓	
INSERT			✓	page 6-28
INSERT cursor	✓			page 6-42
LOAD	✓			page 6-28
LOCK TABLE			✓	
OPEN *	✓		✓	page 6-26
OUTPUT	✓			
PREPARE		✓		page 6-27
PUT	✓			
RENAME COLUMN			✓	
RENAME DATABASE			✓	
RENAME TABLE			✓	
REVOKE			✓	
REVOKE FRAGMENT			✓	
ROLLBACK WORK	✓			
scroll cursor	✓			
SELECT			✓	page 6-60
SET AUTOFREE			✓	

SQL Statement	Works	Works with Restrictions	Application-Server Dependent	Refer To
SET CONNECTION	✓			
SET Database Object Mode			✓	
SET DATASKIP			✓	
SET DEBUG FILE TO			✓	
SET DEFERRED_PREPARE			✓	
SET DESCRIPTOR	✓			
SET EXPLAIN			✓	
SET ISOLATION		✓		page 6-23
SET LOCK MODE			✓	
SET LOG			✓	
SET OPTIMIZATION			✓	
SET PDQPRIORITY			✓	
SET PLOAD FILE			✓	
SET Residency			✓	
SET ROLE			✓	
SET SCHEDULE LEVEL			✓	
SET SESSION AUTHORIZATION			✓	
SET TRANSACTION				page 6-25
START VIOLATIONS TABLE			✓	

(5 of 6)

SQL Statement	Works	Works with Restrictions	Application-Server Dependent	Refer To
STOP VIOLATIONS TABLE			✓	
UNLOAD	✓			
UNLOCK TABLE			✓	
UPDATE			✓	
UPDATE STATISTICS			✓	
WHENEVER	✓			

(6 of 6)

General Considerations When Writing Application Programs

Although some Informix statements and extensions to SQL are not available on application servers, often other options that serve the same function are available.

This section includes notes about some SQL statements that are of particular interest when you write applications to run on the Gateway. It does not include comprehensive documentation. For more complete information about the statements that are discussed here, refer to the *Informix Guide to SQL: Reference* (for versions prior to 6.0) or to the *Informix Guide to SQL: Syntax*.

Using ANSI SQL

Informix recommends that you use ANSI-compliant SQL for all Gateway applications. DRDA-compliant application servers understand ANSI SQL, but they might not understand non-ANSI SQL statements.

Using Decimal Points

Decimal points must be represented as periods (**.**) in both static and dynamic SQL statements. The use of commas (**,**) as decimal points is not supported.

Using Quotation Marks

In Informix applications, you can use single and double quotation marks interchangeably. ANSI SQL does not permit you to interchange quotation marks. ANSI SQL uses single quotes around quoted strings and double quotes around delimited identifiers. Follow the ANSI standards, as shown in the following example:

```
INSERT INTO "user1".table1 VALUES ('abc', 2, 'zip')
```

Although the use of ANSI-style double quotes and single quotes is recommended, an SQL statement that conforms to Informix SQL syntax and uses double quotes around a string literal can be used. The Gateway converts the double quotes to single quotes before sending the SQL statement to the application server. The SQL statement must conform to Informix SQL syntax in order for the conversion to succeed.

For information on the meaning of single quotes and double quotes when the **DELIMIDENT** environment variable is set, see "Using Delimited Identifiers" on page 6-18.

Using Delimited Identifiers

The Gateway supports delimited identifiers. It uses ANSI-standard syntax rules to identify delimited identifiers and character literal strings. Delimited identifiers can include alphanumeric characters and any nonalphanumeric characters including spaces, whereas nondelimited identifiers can only include alphanumeric characters and underscores. Support for delimited identifiers means that the Gateway can access any column of any table at a target application server, no matter how the table or column names are named.

DELIMIDENT Environment Variable

You must set the **DELIMIDENT** environment variable if you plan to use delimited identifiers. The Gateway recognizes **DELIMIDENT** as being set if it is either in the Gateway environment or in the environment of a Version 7.0 or later Informix client. The client must be connected to the Gateway or to an Informix Dynamic Server which is in turn connected to the Gateway.

If **DELIMIDENT** is set, the Gateway parses SQL statements by using ANSI rules for delimited identifiers and string constants. When **DELIMIDENT** is set, you must use single quotation marks (‘ ‘) to delimit string constants. Any string delimited by double quotation marks (“ “) is considered an identifier.

Processing Nondelimited and Delimited Identifiers

Identifiers specify the names of database objects such as tables and columns in SQL statements. The two types of identifiers are nondelimited identifiers and delimited identifiers. You specify a nondelimited identifier as a string without any delimiters. For example, you specify a table as `customers` without any quotation marks. You specify a delimited identifier as a string delimited by double quotation marks. For example, you specify a table as `"customers"`.

Informix products are not case sensitive to nondelimited identifiers in SQL statements. When **DELIMIDENT** is not set, the database server shifts nondelimited identifiers to lowercase before they are used. Thus, the following sample statements produce the same result:

```
SELECT col1 FROM database@servername:owner.tabname

SELECT Col1 FROM database@servername:owner.tabname
```

Informix products are case sensitive to delimited identifiers in SQL statements. When you enable delimited-identifier support by setting the **DELIMIDENT** environment variable, the Gateway does not change the value of the delimited identifier before it is used.

Delimiting All Identifiers in a Query

When the **DELIMIDENT** environment variable is set in distributed mode, you should delimit all identifiers in the query. The following statement might result in an error because **col2** is not delimited:

```
SELECT "Col1", col2 FROM database@server:owner."Tabname"
```

You should delimit column **col2** and specify it exactly as it is specified in the application-server table definition. For example, you can rewrite the preceding SELECT statement as follows:

```
SELECT "Col1", "COL2" FROM database@server:owner."Tabname"
```

Delimited Identifiers in Distributed Mode

When you use delimited identifiers in distributed mode, you cannot use the Informix-style catalogs for catalog information unless all of the referenced table and column names are in lower case.

Using Conditions in SQL Statements

You can use conditional clauses, as defined in the Condition segment of the syntax chapter of the *Informix Guide to SQL: Reference* or the *Informix Guide to SQL: Syntax*, as long as you observe the following rules:

- You can use all the ANSI-standard forms.
- Non-ANSI forms might or might not work, as the following example describes:
  - You cannot use the keyword UNIQUE (use DISTINCT instead).
  - You cannot use the keyword ROWID.
- You can use only objects that the application server recognizes. In other words, all table names must use application-server syntax, any variables used must be data types that can be mapped to application server data types, and so forth.

Translating the MATCHES Condition to the LIKE Condition

Informix tools such as 4GL and ViewPoint recognize only *, ?, [, and] as wildcard characters and generate SQL statements that use the MATCHES condition. The DRDA application servers (DB2, OS/400, and SQL/DS) support only the LIKE condition (and the wildcard characters % and _), which conforms to the ANSI SQL standard.

In general, the MATCHES condition can be translated to the LIKE condition along with the mapping of the associated special characters. This permits the existing applications to work unmodified with the DRDA Gateway, although with a few restrictions.

Translating Wildcard Characters

The MATCHES condition uses four wildcard characters: *, ?, [, and]. The LIKE condition uses only two: % and _ (which correspond to * and ?, respectively No direct equivalents exist for [and] in the LIKE condition.

You can suppress the special meaning of the wildcard characters with an ESCAPE clause in both the MATCHES and the LIKE conditions. Consider the following example:

```
SELECT * FROM tab1 WHERE col1 MATCHES <pattern> ESCAPE '\'
```

The following table lists the rules for how to translate MATCHES to LIKE.

MATCHES Pattern	Translation
1. Character string literal with no wildcard characters: `MATCHES 'abcdef'`	Translate MATCHES to LIKE. No change to pattern: `LIKE 'abcdef'`
2. Pattern has * and ?, no escapes, and no [and]: `MATCHES 'ab*cd?'`	Translate MATCHES to LIKE. Map * and ? to % and _, respectively: `LIKE 'ab%cd_'`

(1 of 2)

MATCHES Pattern	Translation
3. Pattern has [and] (and optionally * and ?): `MATCHES 'a[bc]'`	Cannot translate. Return an error to the client with the pattern as a token: **sqlca.sqlcode** = -29040 **sqlca.sqlerrm** = *pattern*
4. Pattern has *, ?, [, or], which are escaped: `MATCHES 'a*b\*\['`	Translate MATCHES to LIKE. Strip the ESCAPE characters from pattern: `LIKE 'a%b*['`
5. Pattern contains the LIKE wildcard characters %, or _: `MATCHES 'a*%c_'`	Translate MATCHES to LIKE. Insert ESCAPE character before % and _. Add ESCAPE clause to the LIKE condition: `LIKE 'a%\%c\_' ESCAPE '\'`

(2 of 2)

In all the cases in the table, an ESCAPE clause is generated if the translated pattern contains an escape sequence. The escape character that is used (as in rule 5) is the escape character that is specified in the ESCAPE clause.

Translation Limitations

Translation is performed only if the pattern is specified as a literal string and not as a host variable. This is not as great a restriction for 4GL applications that use the CONSTRUCT statement to implement a query by forms (QBF) feature because the CONSTRUCT statement generates the pattern for MATCHES as a character-string literal.

The following examples can be translated to use the LIKE condition:

```
SELECT col2 FROM tab1 WHERE col1 MATCHES 'a*c'

PREPARE s1 FROM "SELECT col2 FROM tab1
        WHERE col1 MATCHES 'a*c'"
EXECUTE s1
```

Translation fails for dynamic SQL statements that pass the pattern as a parameter during the execution phase of a prepared statement. The following example *cannot* be translated to use the LIKE condition. The PREPARE statement causes a DRDA application-server error because the application server will not parse a statement that contains the MATCHES keyword.

```
strcpy(pattern, "a*c");
PREPARE s1 FROM 'SELECT * FROM tab1 WHERE col1 MATCHES ?';
EXECUTE s1 USING :pattern;
```

Transaction Handling

The application server behaves the same as an ANSI-compliant Informix database where transactions are concerned. That is, BEGIN WORK is performed implicitly on the first SQL statement that an application executes or on the first statement that follows a COMMIT WORK or a ROLLBACK WORK.

The SET ISOLATION Statement

You can use the full syntax of the SET ISOLATION statement, but application servers that use the DRDA protocol support only two isolation levels. The Gateway maps the Informix isolation levels to the application-server isolation levels as the following table shows.

Informix Isolation Level	Maps to Application Server Isolation Level
Dirty Read	Cursor Stability
Committed Read	
Cursor Stability	
Repeatable Read	Repeatable Read

Important: *The default isolation level is Repeatable Read. Informix recommends that the isolation mode for all applications be set to Cursor Stability unless Repeatable Read is needed. Repeatable Read is the default isolation level in compliance with ANSI standards.*

The SET ISOLATION statement does not cause any statements to be sent to the application server. Instead, the SET ISOLATION level causes the Gateway to use sections from a Cursor Stability package instead of a Repeatable Read package, or vice versa. (For information about how isolation levels are implemented, refer to your application-server manuals.)

Dynamic statements use the isolation level that is effective when the statement is prepared. Static cursors use the isolation level that is effective when the cursor is opened. Other static statements use the isolation level that is in force when the statement is executed.

Using the GWISOLEVEL Environment Variable

Use the **GWISOLEVEL** environment variable to specify the initial isolation level explicitly. This allows the Gateway administrator to optionally configure an initial isolation level for all users by specifying a value for **GWISOLEVEL** in the common environment-configuration file **$INFORMIXDIR/etc/informix.rc**. Individual users can override this file with the private environment-configuration file **.informix** in their home directory.

The **GWISOLEVEL** environment variable has the following values:

Value	Meaning
1	Dirty Read
2	Committed Read
3	Cursor Stability
4	Repeatable Read

These values correspond to the isolation levels that Informix Dynamic Server supports. The Gateway maps these values to the appropriate isolation levels that the DRDA application server supports. When you specify a **GWISOLEVEL** value of 1, 2, or 3, an initial isolation level of Cursor Stability is chosen at the DRDA application server. Any other value (including 4) results in the initial isolation level of Repeatable Read being chosen at the application server. If **GWISOLEVEL** is not set, the Gateway uses Repeatable Read as the default initial isolation level at connect time.

Setting **GWISOLEVEL** is effective only in the following two cases:

- Direct-mode connection
- Distributed-mode connection from an unlogged Informix database

  You can set **GWISOLEVEL** to 3 (that is, CURSOR STABILITY). If you bound the package with **GWMAPCS** set to 2, you can update the nonjournaled files n OS/400 from an unlogged Informix database.

Tip: *You might encounter error -7008 if you access AS400 nonjournaled files in libraries with the isolation level set to Cursor Stability. To avoid this error, use Repeatable Read or set the* ***GWMAPCS*** *environment variable and rebind the package to start journaling on the physical file. See* *"Access to Nonjournaled OS/400 Files" on page 5-29*.

The SET TRANSACTION Statement

When you execute the SET TRANSACTION statement in direct mode, the statement functions as the documentation for the application server describes.

When you execute this statement in distributed mode, the behavior of SET TRANSACTION with the ISOLATION LEVEL clause is similar to the behavior of the SET ISOLATION statement described on page 6-23. Other SET TRANSACTION statements have no effect.

Creating Database Objects

In general, DDL statements such as CREATE TABLE or CREATE INDEX might not behave the same way in an application-server environment as in an Informix environment. The two environments might have vastly different concepts of how a database is defined. It is important that you understand how databases and tables work in an application-server environment. In general, you should use dynamic statements and the syntax of the application server when you use any DDL statement. For information, refer to your application-server manual.

CREATE VIEW statements must explicitly qualify selected tables with owner names, as the following example shows:

```
CREATE VIEW someview AS SELECT * FROM "USER1".tab1
```

If you omit the owner name, the distributed queries that reference the view from Dynamic Server databases might fail. A Dynamic Server coordinator accesses a view by selecting data from the underlying base table(s) on which the view is defined. If a base-table name is unqualified in the view definition, the Dynamic Server coordinator qualifies the base table with an owner name according to the rules specified in "Fully Qualified Object Names" on page 6-45. This owner name could be different from the base-table owner name that the application server uses when the view is created, and the view access could consequently provide incorrect results or fail.

Using a Cursor

The default fetch mode of a cursor is *block fetch*, that is, FETCH for SELECT only. To declare a cursor that can modify rows, you must use the FOR UPDATE clause, in which case *single-row fetch* is used. If the FOR UPDATE clause is not included, block fetch is used.

All forms of the Informix DECLARE cursor and OPEN cursor statements are supported in the DRDA protocols. The DECLARE CURSOR statement executes successfully if the SQL text that is associated with the cursor is static and conforms to Informix SQL syntax or if the SQL text is dynamic. However, if the SQL text does not conform to the syntax of the application server, the subsequent OPEN of the cursor fails. When the OPEN statement is executed, the SQL text is sent for the first time to the Gateway and is passed on to the application server for processing. The application server checks the SQL text and flags any syntax errors only at OPEN time.

Important: *SQL/DS does not currently support hold cursors.*

Using Dynamic SQL Statements

By definition, any statement enclosed in an EXECUTE IMMEDIATE statement or a PREPARE and EXECUTE statement pair is a dynamic statement. Typically, an application uses dynamic statements where some input from the user is required in constructing the statement.

The EXECUTE IMMEDIATE, PREPARE, and EXECUTE Statements

To issue an application server-specific statement, make the statement dynamic with EXECUTE IMMEDIATE or PREPARE and EXECUTE, as the following examples show:

```
EXECUTE IMMEDIATE :stmt
```

or

```
PREPARE s1 FROM :stmt
EXECUTE :stmt
```

In both of these examples `:stmt` is an ESQL/C host variable that contains the statement string.

Application servers that follow the DRDA protocol can handle only one SQL statement in each EXECUTE IMMEDIATE or PREPARE and EXECUTE statement. If the Gateway detects that multiple SQL statements (that is, SQL statements separated by a semicolon) are included in a dynamic statement, the statement is rejected.

The DESCRIBE Statement

If the DESCRIBE statement is used on a statement that matches Informix syntax, the constant name for the statement type is defined in the Informix file **sqlstype.h**. For all other statements that do not conform to Informix syntax, the constant name **SQ_UNKNOWN** is returned as the statement type.

You can use the DESCRIBE statement to describe column data that the SELECT statement returns. You can also use it to describe the input values (data types and lengths) of an INSERT statement.

Column names that the DESCRIBE statement returns are lowercased. In this respect, the Gateway functions the same as Dynamic Server.

Inserting into VARCHAR Columns Using LOAD and INSERT Statements

The LOAD and INSERT cursor statements remove trailing blanks from values that are inserted into application-server columns of type VARCHAR(*n*) where *n* is greater than 254 characters.

Using Scroll Cursor Statements

Scroll cursors function in the same way with the Gateway as they do with Dynamic Servers. For more details, refer to the *Informix Guide to SQL: Syntax*.

The Gateway uses temporary files to implement scroll cursors. The Gateway creates the temporary files in the directory that the **DBTEMP** environment variable specifies. If the **DBTEMP** environment variable is set in the client environment, this value overrides the **DBTEMP** setting in the Gateway environment. If **DBTEMP** is not set, the Gateway creates the temporary files in the **/tmp** directory.

Using Multibyte Character Data

IBM database products that are configured to handle multibyte character data typically use three distinct code sets to store data, as the following list describes:

- The *Single Byte Character Set* (SBCS), used for characters that are represented internally with single-byte code points
- The *Double Byte Character Set* (DBCS), used for characters that are represented internally with double-byte code points
- The *Mixed Character Set* (MCS), used for character strings that contain characters from both the SBCS and DBCS code sets. In the Mixed Character Set, double-byte characters are delimited by the reserved single-byte code points 0x0e (called *shift-out*) and 0x0f (called *shift-in*).

Multibyte Data Types

Multibyte data types that IBM application servers support fall into two groups:

- Columns that are declared as GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC contain only DBCS characters.
- Columns that are declared as CHAR, VARCHAR, or LONG VARCHAR contain either SBCS characters or MCS characters, depending on how the column declaration was qualified when the table was created. For example, the column declaration might include a qualifier such as FOR SBCS DATA or FOR MIXED DATA.

Character-Data Length Expansion

When a client application uses a multibyte code set (typically this applies only to Asian languages), character data can expand in length when it is converted from the code set of the application server into the code set of the client application. Because the extent of such expansion depends on the character data that is returned, the expansion cannot be predicted before the data is fetched.

Tip: *For clients with single-byte code sets, inbound character-data expansion cannot occur, and the reported output SQLDA lengths for character-data fields are the same as the field lengths at the application server.*

How the Gateway Calculates Data-Length Expansion

To ensure that the Gateway can return character data even when the character string expands in length, the Gateway calculates an expansion factor for each code set. The Gateway applies this expansion factor to the byte length for every output field on the application server that maps to a CHAR or VARCHAR field in an output SQLDA returned for a PREPARE statement or a DESCRIBE statement.

The expanded lengths reported for character-type output fields depend on the specific code-set conversion that will be performed on the fields. For example, if the largest character code-point conversion defined is from a 2-byte code point to a 3-byte code point, then the reported length of the output field is 1.5 times the length that the application server reports. The upper limit for any reported expanded output character field length is 32,767 bytes (the maximum length for a CHAR data type).

Data Expansion "Too Long" Error

If a character output field value expands beyond its reported output SQLDA field length, error -29034 with a conversion failure type of `too long` is returned to the user. This error occurs only when the data expands past 32,767 bytes in length, because the worst-case expansion is always reported in the output SQLDA, unless it exceeds 32,767 bytes.

Accessing Application-Server Multibyte Character Data

You can use the Gateway to access both SBCS and MCS data on the application server. If you use host variables, they must be of a character data type (CHAR, CHAR *, STRING, FIXCHAR, or VARCHAR). You can also use literal strings in SQL statements to send multibyte characters to the application server. The Gateway can convert multibyte character data to and from the code sets of the application server.

Null-terminated character strings (that is, STRING or CHAR *) or VARCHAR input host variables are preferable whenever you send multibyte-character data with input host variables to the application server.

You can select GRAPHIC data into character data-type host variables (CHAR, CHAR *, STRING, FIXCHAR, or VARCHAR), but you cannot send character data in host variables to the application server for use as GRAPHIC data. For details, see "Sending GRAPHIC Data to the Application Server" on page 6-30.

Do not send trailing blanks to the application server unless the blanks are significant. The expanded length of the trailing blanks might be enough to cause the length to exceed the length of the target column.

Sending GRAPHIC Data to the Application Server

To send character data to an application server for use as GRAPHIC-type data, you must use GRAPHIC string literals in an SQL statement. You cannot use input host variables to send GRAPHIC-type character data to the application server. Informix does not have a GRAPHIC data-type host variable, and application servers do not convert from CHAR types to GRAPHIC types.

On DB2, SQL/DS, and OS/400 you supply the GRAPHIC type data with literal strings within the SQL statement. When you use the Gateway, the syntax for GRAPHIC literals is one of the following equivalent forms:

```
G'******'
N'******'
```

For example, the following ESQL/C statement inserts a value into a GRAPHIC data type:

```
sprintf (stmt, "INSERT INTO graphtab VALUES (G'****')"
EXEC SQL EXECUTE IMMEDIATE :stmt;
```

Each asterisk (*) represents a character that is mapped to a double-byte code point in the GRAPHIC code set of the application server. This syntax is slightly different from the DB2, SQL/DS, and OS/400 syntax because the Gateway automatically adds the shift-out and shift-in code points, which are required to delimit GRAPHIC characters. The Gateway supports this syntax only for dynamic SQL statements (statements that are processed with EXECUTE IMMEDIATE or PREPARE and EXECUTE).

Supplying character data for use as GRAPHIC-type character data at the application server can be difficult because it is sometimes difficult to predict in advance that all the characters supplied will convert into double-byte code points in the code set of the application server. If you have a choice, it is easier to load data into CHAR-type columns that are declared to hold mixed (SBCS and/or DBCS) character data.

Using Multibyte SQL Identifiers

To the same extent that each application server supports the use of multibyte SQL identifiers (for tables, columns, and so on), the Gateway also supports such usage. For example, in DB2 and SQL/DS, ordinary SQL identifiers can contain either SBCS or DBCS characters, but not a mixture of SBCS and DBCS characters. However, in OS/400, ordinary SQL identifiers can contain only SBCS characters.

Database names (*alias_RDB_name*) and server names (*gwservername*) in DATABASE statements or CONNECT statements cannot contain multibyte characters. The Gateway does not recognize multibyte characters for these identifiers.

SQL Statements That You Cannot Use

Normally, you execute an application-server-specific SQL statement that is not an Informix statement with a dynamic statement (EXECUTE IMMEDIATE or PREPARE). However, some SQL statements are static-only statements. That is, you cannot use those SQL statements in dynamic form. If the statement is not a valid Informix statement *and* the application-server syntax forbids using it as a dynamic statement, you cannot use that statement in an application that uses the Gateway.

For example, you cannot use the following DB2 SQL statement:

```
SET host-variable=special register
```

Stored Procedures

Informix Enterprise Gateway with DRDA lets you execute stored procedures on supported application servers in direct and distributed mode. Use the EXECUTE PROCEDURE statement to invoke and receive output parameters from stored procedures. The syntax and use of the EXECUTE PROCEDURE statement are described in the *Informix Guide to SQL: Syntax*.

General Restrictions and Limitations

In a DRDA environment, stored procedures are generally created locally with a procedural language such as C or COBOL. The stored procedures must be registered (usually manually) in the system catalog for the application server. Thus, you cannot use other Informix SQL statements relating to stored procedures, including CREATE PROCEDURE, DROP PROCEDURE, and GRANT EXECUTE ON *procedure-name*.

Not all application servers that support stored procedures systematically anchor the procedure execution information in their system catalog tables. Also, different application servers operate differently. For example, a DB2 for MVS 4.1 application server stores procedure information in the system catalog table **SYSIBM.SYSPROCEDURES**, but an equivalent system catalog table does not exist on a DB2 for OS/400 3.1 application server.

To provide support for executing DB2 for OS/400 stored procedures, the DB2 for OS/400 administrator must create a new table, populate it appropriately, and specify this table by setting the **GWPROCINFOTABLE** environment variable. For more information about setting **GWPROCINFOTABLE**, see "The GWPROCINFOTABLE Environment Variable" on page 6-33.

Application Server-Specific Limitations

When you invoke stored procedures on a DB2/MVS or DB2 for OS/390, system, if the procedure name given to the EXECUTE PROCEDURE statement is qualified by the owner name (according to Informix syntax), the qualifier must be **SYSPROC**. If any other qualifier is supplied, the Gateway returns error -674. If the qualifier is not specified, the Gateway assumes **SYSPROC** as the COLLECTION ID. This occurs because, for DRDA, if the server is DB2/MVS, or DB2 for OS/390, the qualifier (such as COLLECTION ID) for a two-part procedure name must be **SYSPROC**.

For a DB2 for OS/400 system, if the procedure name is not qualified by the COLLECTION ID, the Gateway uses the *RDB_user_ID* as the COLLECTION ID for the stored procedure.

The GWPROCINFOTABLE Environment Variable

The **GWPROCINFOTABLE** environment variable lets you obtain stored-procedure information on all supported application servers (such as DB2 for OS/400 3.1) that do not store procedure information in the system catalog. Set **GWPROCINFOTABLE** to any valid table name for the user table that has information about stored procedures.

If **GWPROCINFOTABLE** is not set, it defaults to the catalog table **INFORMIX.GWPROCINFO**. You do not have to set this environment variable for the DB2 for the MVS 4.1 or DB2 for OS/390 application server because the Gateway can get necessary information from the **SYSIBM.SYSPROCEDURES** system catalog table.

The following example specifies the schema for the stored procedures catalog table.

```
CREATE TABLE userx.proctaby
    (procname CHAR(18) NOT NULL, collid CHAR(18) NOT NULL,
    linkage CHAR(1) NOT NULL,
    parmlist VARCHAR(3000) NOT NULL);
```

The columns of the example table have the same meaning as those of **SYSIBM.SYSPROCEDURES** in DB2/MVS 4.1 and DB2 for OS/390.

procname	Name of the procedure.
collid	The collection ID associated with the procedure.
linkage	N if the procedure accepts nulls; blank otherwise.
parmlist	String that describes the arguments of the procedures. Contains parameters separated by a comma. Each parameter has an optional parameter name followed by a data-type specifier. Valid data types are INT, INTEGER, SMALLINT, REAL, FLOAT, DOUBLE PRECISION, DECIMAL(*prec*, *scale*), DEC(*prec*, *scale*), CHAR(*size*), VARCHAR(*size*), GRAPHIC(*size*), VARGRAPHIC(*size*) and an argument type which can be one of the keywords IN, OUT, or INOUT.

Tip: *The Gateway does not handle GRAPHIC and VARGRAPHIC types. If the procedure does not take any arguments, enter a blank character. For the syntax of the PARMLIST parameter and other details, refer to the IBM documentation "DB2 for MVS Version 4 Release Guide" (SC26-3394-01).*

The following example SQL statement populates the stored-procedure system catalog table specified in the **GWPROCINFOTABLE** setting.

```
INSERT INTO userx.proctaby
    VALUES ('PROCX', 'QUSERSYS', 'N',
    'parm1 INT IN, CHAR(20) INOUT');
```

The example adds information about a procedure **procx** in the collection QUSERSYS. QUSERSYS takes two arguments, IN and INOUT, which are INTEGER and CHAR(20) types, respectively.

The EXECUTE PROCEDURE Statement

Issue the EXECUTE PROCEDURE statement from the application either statically or dynamically. The latter method uses the PREPARE and EXECUTE statements or the EXECUTE IMMEDIATE statement.

A DRDA stored procedure might return output parameters and result sets. Use the EXECUTE PROCEDURE statement to retrieve result values of output parameters from the target stored procedure. The Gateway returns the output parameters from a DRDA stored procedure to the Informix application as a result set (one row). The Gateway does not support the invocation of stored procedures that return result sets.

The following restrictions apply when you issue the EXECUTE PROCEDURE statement to execute a stored procedure on an application server:

- The arguments can be only host variables, NULLS, character-string literals, or numeric literals. Arguments cannot be an expression of any kind nor an EXECUTE PROCEDURE statement to another (Informix or DRDA) stored procedure.
- You cannot supply arguments with the `argument_name =` syntax. Named arguments are not allowed. All arguments must be supplied as positional arguments.

Even with these restrictions, you can invoke a DRDA stored procedure with the EXECUTE PROCEDURE statement from an Informix application in a way that is comparable to issuing a CALL statement to invoke a stored procedure from an IBM DRDA implementation.

If you need to provide the result of an expression as the input value to a DRDA stored-procedure parameter, evaluate the expression, assign the result to a host variable, and provide the host variable as an argument to the EXECUTE PROCEDURE statement.

Passing Parameters

You must provide input values for all IN and INOUT parameters of the target stored procedure as input arguments to the EXECUTE PROCEDURE statement. The parameters must be in the same sequence as defined in the prototype definition of the stored procedure at the application server. No OUT parameters are required as input arguments to the EXECUTE PROCEDURE statement.

If you use the INTO clause of the EXECUTE PROCEDURE statement, when the stored procedure executes, the resulting values of the OUT and INOUT parameters of the stored procedure are returned to the INTO clause (output) host variables. If you use the cursor mechanism instead, the resulting values of all OUT and INOUT parameters of the stored procedure are returned to the application as a single-row result set.

The following table illustrates the use of DRDA IN, OUT, and INOUT parameters as input and output arguments for stored procedures. Assume that the prototype for the stored procedure **procx** is `procx(a,b,c,d,e,f)` and that the parameters are defined at the application server as follows.

Ordinal Position	Parameter Name	Data Type	Parameter Role
1	a	INT	IN
2	b	INT	INOUT
3	c	CHAR()	IN
4	d	CHAR()	OUT
5	e	INT	OUT
6	f	CHAR()	INOUT

In direct mode, the following example EXECUTE PROCEDURE statement invokes the procedure **procx** from an Informix client through the Gateway:

```
EXECUTE PROCEDURE procx(:ha, 123, NULL, :hf)
    INTO :hb, :hd, :he, :hf;
```

For DB2/MVS or DB2 for OS/390, Informix recommends that you use the correct value of the collection identifier to qualify the procedure name in the EXECUTE PROCEDURE statement. If you do not specify a collection ID qualifier for a procedure, the Gateway assumes the following default values.

Target Application Server	Default Collection ID Used
DB2/MVS or DB2 for OS/390	**SYSPROC**
DB2 for OS/400	User ID of the currently connected user at the RDB

In distributed mode, you can use an EXECUTE PROCEDURE statement that has a fully qualified (four-part) object name, such as the following example shows:

```
EXECUTE PROCEDURE
    db1@gw1:"COLLECTION1".procx(:ha, 123, NULL, :hf)
    INTO :hb, :hd, :he, :hf;
```

For DB2/MVS or DB2 FOR OS/390, if "COLLECTION1" is not **SYSPROC**, the Gateway rejects the statement with error -674.

The following table illustrates the positional relationship between the stored procedure parameters and the EXECUTE PROCEDURE statement arguments.

Parameter Name	Parameter Role	Input Argument	Output Argument
a	IN	:ha	
b	INOUT	123	:hb
c	IN	NULL	
d	OUT		:hd
e	OUT		:he
f	INOUT	:hf	:hf

In the distributed mode example, the alias of the target database name is **db1**, the Gateway name is **gw1**, and the collection name of the stored procedure is **COLLECTION1**. (The fully qualified DRDA stored procedure name is **RDB_NAME.COLLECTION.PROCEDURE**.)

Tip: *Informix recommends that you always enclose the collection name in double quotes. For details, see* *"Object Names in Distributed Queries" on page 6-44*.

Remember that you must provide *all* IN and INOUT parameters to the EXECUTE PROCEDURE statement as input arguments in the same sequence as their ordinal position. This means that you must supply input values for **a**, **b**, **c**, and **f**. When the procedure **procx** executes, the Gateway returns the output values for the parameters **b**, **d**, **e**, and **f**. The resulting values of these parameters are returned because they are marked as either OUT or INOUT. Their output order is due to the sequence of their ordinal position.

Alternatively, in direct mode, you can also invoke the stored procedure by with the following segments of SQL statements, which use the cursor mechanism:

```
DECLARE cur CURSOR FOR
    EXECUTE PROCEDURE procx(?, 123, ?, 'abc');
OPEN cur USING :ha, :hc;
FETCH cur INTO :hb, :hd, :he, :hf;
```

In distributed mode, you can use the following DECLARE CURSOR statement instead:

```
DECLARE cur CURSOR FOR
    EXECUTE PROCEDURE
        db1@gw1:"COLLECTION1".procx(?, 123, ?, 'abc');
```

Once again, all IN and INOUT parameters are provided as input arguments to the EXECUTE PROCEDURE statement. The OUT and INOUT parameters are returned as a result set in the FETCH statement according to the ordinal sequence that was defined at the application server.

Remember that, although the output parameters from the stored procedure are returned as a single-row result set to the Gateway client, these output parameters are not a result set. They resemble a single result set, but the DRDA stored procedure returns only output parameters.The Gateway does not support the invocation of DRDA stored procedures that return result sets.

Gateway Access Modes

A client application can attach directly to Gateway, as Figure 6-3 illustrates, or the Gateway can be a participant in distributed queries that involve both Informix databases and DRDA-compliant database servers.

The configuration that Figure 6-3 illustrates is referred to as *direct-access mode*.

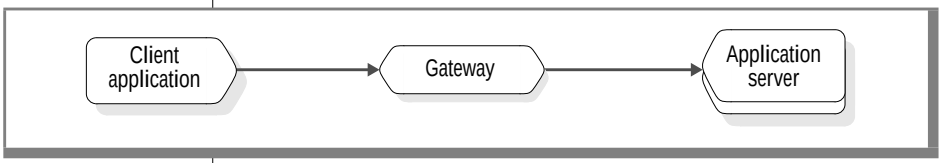


Figure 6-3
Direct-Access Configuration

In distributed queries, the Gateway acts as a subordinate server to an Informix Dynamic Server. This configuration is referred to as *distributed-access mode*. Figure 6-4 illustrates the use of distributed-access mode.

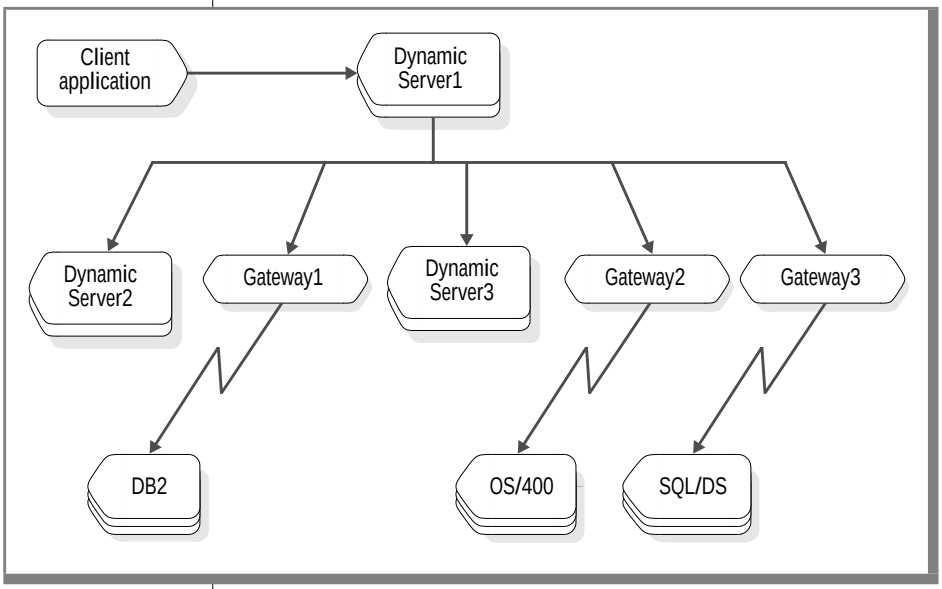


Figure 6-4
Distributed-Query Configuration Using Distributed-Access Mode

Considerations for Direct Access to the Gateway

When you program an application to access the Gateway in a direct-access mode, you must consider the following issues:

- Managing connections
- Non-Informix SQL statements
- ANSI status of the application server
- Performance

Managing Connections

The client and the Gateway process the statements that are used to establish (or terminate) a connection to a specific database server. The application server never sees the statements.

The CONNECT, DISCONNECT, and SET CONNECTION Statements

You can use the CONNECT statement and the associated DISCONNECT and SET CONNECTION statement only with Version 6.0 and higher Informix SQL API products. The full syntax of the CONNECT, DISCONNECT, and SET CONNECTION statements is given in the *Informix Guide to SQL: Syntax*.

The CONNECT statement is the recommended SQL statement for making client / server connections. The following example shows a CONNECT statement for use with the Gateway:

```
CONNECT TO alias_RDB_name@gwservername
```

The names *alias_RDB_name* and *gwservername* used in the preceding example are different from the descriptive names used in the *Informix Guide to SQL: Syntax*, but the interpretation is analogous. The Informix application regards the Gateway as the database server, so *gwservername* is analogous to the server name (*dbservername*) of an Informix database server. For more information, refer to "Step 4: Start the Gateway Daemon" on page 3-24.

The Gateway regards the entire application server as one big database. Hence the *alias_RDB_name* corresponds to the database name of an Informix application. It uses a designated communication mode to specify a connection to an application server. For more information about *alias_RDB_name* and communication modes, refer to "Information About the Advanced Program-to-Program Communication" on page 3-10.

You can also include a user ID and password in the CONNECT statement, as the following example shows:

```
CONNECT TO alias_RDB_name@gwservername USER user_ID
        USING password
```

The *user ID* and *password* provide the user login that should be associated with this connection to the Gateway and possibly to the application server. For more information about the user ID and password, refer to "Enforcing Security" on page 5-12.

The DATABASE Statement

In Version 4.1 and Version 5.0 client applications, you must use the DATABASE statement to open a connection to a database. You can also use the DATABASE statement with Version 6.0 and higher products. The following example shows a sample DATABASE statement for use with the Gateway:

```
DATABASE alias_RDB_name@gwservername
```

The names *alias_RDB_name* and *gwservername* that are used in the preceding example are different from the descriptive names that you would use in a DATABASE statement definition for accessing an Informix database server. The *gwservername* is analogous to the server name (*dbservername*) of an Informix database server. The *gwservername* is established when the Gateway daemon process starts. For more information, refer to "Step 4: Start the Gateway Daemon" on page 3-24.

The Gateway regards the application server as a database. Hence the *alias_RDB_name* is analogous to the database name of an Informix application. The *alias_RDB_name* uses a designated communication mode to specify a connection to an application server. For more information about *alias_RDB_name* and communication modes, refer to "Multiple Connection Modes" on page 3-21.

You cannot use the EXCLUSIVE keyword with the DATABASE statement.

Non-Informix SQL Statements

If the client application connects directly to the Gateway, use the EXECUTE IMMEDIATE or PREPARE and EXECUTE statements to send SQL statements that use syntax that an Informix database server does not recognize to the application server. You cannot issue static non-Informix SQL statements that are specific to the application server because the (Informix) preprocessor rejects them with syntax errors. Refer to "Processing by the Client" on page 6-8.

ANSI Status of the Application Server

When the client application connects directly to the Gateway, as in Figure 6-3 on page 6-38, the Gateway represents the application server to the client application as an ANSI-compliant database server with transaction logging.

General Performance Considerations

The same general performance considerations that apply to writing applications that access data on a local database also apply to applications that access data on remote databases. The effect of the following guidelines can be much more dramatic in distributed applications than in local applications:

- If you execute an SQL statement more than once, use a dynamic statement. PREPARE the statement once and then EXECUTE it many times. In a distributed application, the number of messages sent to the server is usually the most critical performance determinant. When you use dynamic statements to execute multiple statements, you save one message per statement execution. That is, each statement requires a single EXECUTE message versus a PREPARE and EXECUTE message pair.
- Use an insert cursor when you insert a large number of rows into a table. When you use an insert cursor, the client application puts many rows into each message that it sends to the application, reducing the number of messages required to insert the data.

- When you use a SELECT statement to retrieve data from an application server, do not use an update cursor unless you plan to update the rows. If the cursor is not an update cursor, the application server sends the rows of data back in block-fetch mode. In block-fetch mode as many rows of data as can fit into a single buffer are sent in a single message. Thus, the number of messages required to send the data is greatly reduced. (The buffer size is set by the Gateway administrator. Refer to "The Buffer Size" on page 3-11.)
- Whenever possible, make a few SQL statements do the work of many SQL statements. For example, you might be able to update many rows with a single SQL statement through careful use of the WHERE clause. A single statement that updates many rows is more efficient than an individual UPDATE statement for each row.

Considerations for Distributed Access to the Gateway

When you program an application to access the Gateway in distributed-access mode, consider the following issues:

- Non-Informix SQL statements
- Mixed-mode database access
- Fully qualified object names
- Synonyms used to refer to application server objects
- Owner or collection IDs required for OS/400 objects
- SQL text in a cursor
- Separate Gateway daemons are required for each application server
- Two-phase commit
- Accessing views
- The GWVTOT environment variable
- Performance

Non-Informix SQL Statements

If the Gateway is part of a distributed query, the client application connects to a Dynamic Server that acts as the coordinator for the transaction. Each statement from the client application is passed first to Dynamic Server. Dynamic Server parses the statement. If the statement is not acceptable to Dynamic Server, Dynamic Server rejects the statement. Thus, you cannot pass a non-Informix statement to the application server when the Gateway is part of a distributed query.

Application-server syntax allows three-part table names. However, a client application cannot use a three-part table name in a distributed transaction because three-part table names are rejected by the coordinating Dynamic Server. The statement fails in the same way that non-Informix statements fail. (See "Non-Informix SQL Statements" on page 6-41.)

Mixed-Mode Database Access

The Gateway allows access from nonlogged Informix databases to DB2 databases. Observe the following considerations when you access DB2 databases from nonlogged Informix databases.

- All of the statements passed to DB2 are followed by a commit, so each statement becomes a transaction.
- If you open an insert cursor to insert rows into a DB2 database, each flush of the application buffer commits the updates in the cursor. A flush occurs when the insert buffer becomes full or an explicit FLUSH or CLOSE statement is executed.

  Statements of the following kind use an implicit insert cursor:

  ```
  INSERT INTO r42sales@gw_sales:"JOHN".cust_tab
      SELECT * FROM my_customers
  ```

  In this example **r42sales** is the alias name, **gw_sales** is the Gateway name, **cust_tab** is a table in a DB2 database, and **my_customers** is a table in an Informix database. If an error occurs while a row is inserted into the DB2 database, all the previous inserts caused by this statement will not be rolled back.

Object Names in Distributed Queries

You can use one of the following methods to reference objects in distributed queries:

- Fully or partially qualified object names
- Synonyms

In a distributed query, the Dynamic Server coordinator and the Gateway work together to replace Dynamic Server-style object names with application server-style two-part object names before the query is sent to the application server.

The following sections explain in detail the derivation of two-part object names from synonyms and fully or partially qualified object names. To avoid ambiguity, observe the following recommendations when you reference objects in a distributed query:

- Use fully qualified object names or synonyms that are defined with fully qualified object names.
- In addition, delimit the owner-collection name with double quotes. The following example statements illustrate the use of double quotes:

```
SELECT * FROM db2@my_gw:"TERRYH".table1
CREATE SYNONYM as_table1 FOR db2@my_gw:"TERRYH".table1
```

If you choose not to follow these recommendations, read the next sections carefully to understand what objects are referenced when you use either partially qualified object names or object names with nondelimited owner-collection names.

Synonyms have many advantages when used for distributed objects. Refer to "Synonyms" on page 6-48.

Fully Qualified Object Names

A fully qualified reference to an Informix database object has the following format:

```
database@servername:owner.objname
```

The interpretation of the pieces of a fully qualified object name is different on an Informix database server and on a Gateway. The individual pieces of the object name are defined in the following table.

Symbolic Name	Definition for an Informix Database Server	Definition for Gateway
database	Database name	*alias_RDB_name*
servername	Name of the database server that manages the database (*dbservername*)	Name of the Gateway that connects to the DRDA application server (*gwservername*)
owner	Owner of the object	High-level qualifier of the object at the application server
objname	Name of the object that is referenced (such as table or view)	Name of the object that is referenced

Partially Qualified Object Names

A partially qualified object name has one of the following forms:

- *tablename*
- *owner* . *tablename*
- *database* : *tablename*
- *database* : *owner.tablename*
- *database@servername* : *tablename*

For a partially qualified object name, if the database value is not specified, it defaults to the current database. If the server-name value is not specified, it defaults to the current server.

The Dynamic Server coordinator uses the server-name value to connect to the server (in this case to a Gateway). The Gateway then uses the database value to connect to the application server.

Finally, if the owner name is not present, the Gateway replaces the object name in the SQL statement with the following two-part object name:

```
owner.objname
```

The values used for *owner* and *objname* are the values that the Dynamic Server coordinator sends. The Dynamic Server coordinator does not change the *objname* value from the value that the user specifies. However, the Dynamic Server coordinator might change the *owner* value that the user specifies. The Dynamic Server coordinator uses the following rules to construct the owner value that is sent to the Gateway (and subsequently sent to the application server):

1. If *owner* is specified in quotes in the original statement, then the owner is sent as specified.
2. If *owner* is specified, but not specified in quotes, the following conditions apply:
   - If the Dynamic Server database is an ANSI database, the original owner value is uppercased and sent as a quoted value.
   - If the Dynamic Server database is a non-ANSI database, the original owner value is lowercased and sent as a quoted value.
3. If the owner is not specified, the following conditions apply:
   - If the Dynamic Server database is an ANSI database, the user's UNIX user ID is sent as a quoted value.
   - If the Dynamic Server database is a non-ANSI database and the application server is a DB2 or SQL/DS, no owner is sent.
   - If the Dynamic Server database is a non-ANSI database and the application server is an OS/400, the Gateway returns a -29053 error because an owner is required. For more information, see "Collection IDs That OS/400 Objects Require" on page 6-48.

Figure 6-5 on page 6-47 illustrates these rules as well as the rules for a direct-access Gateway connection.

Figure 6-5
Owner Value That the Application Server Uses

Owner value as specified in SQL statement	Owner value that the application server uses		
	Client connects to the Gateway through a Dynamic Server database		Client connects to the Gateway directly
	ANSI Dynamic Server database	Non-ANSI Dynamic Server database	No Dynamic Server database
"John"	"John"	"John"	"John"
John	"JOHN"	"john"	JOHN*
not specified	"bob"	DB2→ SALES** OS/400→ error -29053	SALES**

In this table, **bob** is the UNIX user ID that is used to connect to the Gateway. **SALES** is the user ID that **bob** uses at the application server. For example, the **gwdba** utility might have been used to map **bob** to an *RDB_user_ID* of **SALES**.

*The application server uppercases **John** to **JOHN**.

The application server uses the user ID that **bob uses at the application server. In this case, it is **SALES**.

Synonyms

If you do not want to repeatedly type a long four-part name, you can create a synonym in the Dynamic Server database to reference your application-server table, as the following example shows:

```
CREATE SYNONYM as_table1 FOR db2@my_gw:"TERRYH".table1
```

Once created, you can use the synonym as if it refers to a local table.

After you have created the synonym, you can use it in any SQL statement, as the following example shows:

```
SELECT * FROM as_table1
```

The two-part name sent to the application server is derived from the synonym definition, based on the derivation rules explained in "Partially Qualified Object Names" on page 6-45. In the preceding example, the two-part name sent to the application server is "**TERRYH**"**.table1**.

If the location of the table changes, you can drop the synonym and create a new synonym that refers to the table at the new location. You do not need to change the SQL statements in your application.

Collection IDs That OS/400 Objects Require

References to OS/400 objects in distributed queries must be qualified with the name of the collection in which the object exists, as the following example shows:

```
SELECT * FROM as400@my_gw:"USER1".table1
```

In this case, the qualifier **table1** is an object in the collection **USER1**. This qualifier is also referred to as the *owner of the object*.

Warning: *In distributed queries, the COLLECTION ID cannot exceed eight characters. This limitation stems from the length limit for Dynamic Server owner names.*

If a DB2 object reference is not qualified, the Gateway issues an **sqlcode** of -29053.

Cursor Text

In distributed access mode, the OPEN *cursor id* statement fails if the SQL text associated with a cursor does not conform to the SQL syntax of the application server as well as the Informix SQL syntax. When the OPEN *cursor id* statement is executed, the Dynamic Server must first successfully parse the SQL text before it is sent to the Gateway and application server for further processing.

Separate Gateway Daemon Required for Each Application Server

Each application server must be accessed through a separate Gateway daemon with a unique server name (*gwservername*). Violation of this rule results in a -29052 **sqlcode**.

Without this restriction, a Dynamic Server coordinator could conclude that tables on two distinct application servers were on a single server, and generate a request for one of the application servers to perform a distributed join of the tables. This would fail because currently no DRDA application server is capable of performing a distributed join.

Single-Site Updates

Single-site updates in a distributed transaction that involves the Gateway can always be committed. The update site can be either a data source or an Dynamic Server database.

Multi-Site Updates and Heterogeneous Commit

The heterogeneous multi-site update capabilities of Dynamic Server coordinators differ, depending on whether you enable the Dynamic Server heterogeneous commit feature. To enable this feature, you must set the **HETERO_COMMIT** configuration parameter in the **ONCONFIG** file for your Dynamic Server. For details, refer to the description of the heterogeneous commit feature in your *Administrator's Guide*.

If Heterogeneous Commit Is Enabled

If you enable the heterogeneous commit feature in the Dynamic Server that is used with the Gateway, you can update any number of Dynamic Server databases and update at most one data source that is accessed by the Gateway in the same transaction.

If Heterogeneous Commit Is Not Supported or Not Enabled

If the heterogeneous commit feature is not supported or is not enabled in the Dynamic Server coordinator, a multi-site update that contains an updated data source cannot be committed. Only Version 7.2 or later Dynamic Servers support the heterogeneous commit feature.

Identification of Update Sites

Within a distributed transaction, the Dynamic Server coordinator considers a site to be an update site in the following cases:

- An SQL statement that modifies data (such as UPDATE, INSERT, and DELETE) was executed on the site during the transaction.
- A stored procedure was executed on the site during the transaction, even if the stored procedure did not perform an update. This assumption applies even to stored procedures executed on the Dynamic Server coordinator. The Dynamic Server coordinator does not try to determine whether an update actually occurred.

Single-Site and Multi-Site Update Examples

Consider the configuration that Figure 6-7 on page 6-52 shows. The table in Figure 6-6 characterizes the types of transactions that can occur in this configuration and identifies whether each type of transaction can be committed.

Figure 6-6
Transaction Types

Transaction Type	Number of Updated Application Servers (dbgw1 and/or dbgw2)	Is online1 and/or dbonline2 Updated?	Can Transaction Be Committed?
1	0	Yes or No	Yes
2	1	No	Yes

(1 of 2)

Transaction Type	Number of Updated Application Servers (dbgw1 and/or dbgw2)	Is online1 and/or dbonline2 Updated?	Can Transaction Be Committed?
3	1	Yes	Yes, if **online1** supports heterogeneous commit and it is enabled. No, if **online1** does not support heterogeneous commit or it is not enabled.
4	2	Yes or No	No

(2 of 2)

The next sections show examples of the type-1, type-2, type-3, and type-4 distributed transactions as defined above.

Updating Two Informix Servers with an SQL Query

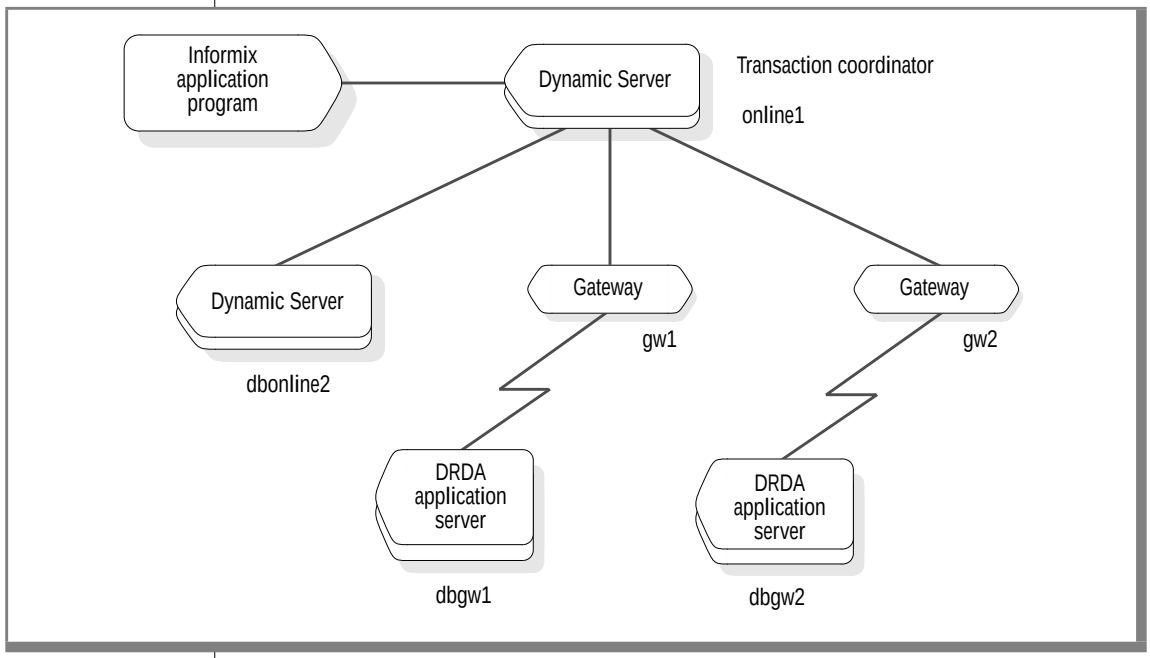
The following transaction involves two Dynamic Server update sites and two DRDA application-server read-only sites:

```
DATABASE dbonline1@online1
BEGIN WORK
INSERT INTO dbonline1@online1:ol1customers
    SELECT * FROM dbgw1@gw1:"USR1".dr1customers
INSERT INTO dbonline2@online2:ol2products
    SELECT * FROM dbgw2@gw2:"USR1".dr2products
COMMIT WORK
```

Data from table **dr1customers** at the Gateway application-server site **gw1** (*alias_RDB_name* **dbgw1**) is inserted into table **ol1customers** at Dynamic Server site **online1** (database **dbonline1**). Also, in the same transaction, data from table **dr2products** at Gateway application-server site **gw2** (*alias_RDB_name* **dbgw2**) is inserted into table **ol2products** at Dynamic Server site **online2** (database **dbonline2**).

This is a type-1 transaction and can be committed whether or not **online1** supports or enables heterogeneous commit.

Figure 6-7
Configuration That the Distributed-Query Examples Use



An SQL Query Block That Updates an Application Server

The following transaction involves one Dynamic Server read-only site and a DRDA application-server update site:

```
DATABASE dbonline1@online1
BEGIN WORK
    SELECT name FROM dbonline1@online1:ol1orders
    INTO :name WHERE empno=12
    EXECUTE PROCEDURE dbgw1@gw1:"USR1".dr1orders_proc()
COMMIT WORK
```

When you use the EXECUTE PROCEDURE statement, the remote procedure **dr1orders_proc** inserts the data into table **dr1orders** at the application server site **gw1** (*alias_RDB_name* **dbgw1**). Also, data from the table at the Dynamic Server site **online1** (database **dbonline1**) is fetched into the host variable *:name*.

This is a type-2 transaction and can be committed whether or not **online1** supports or enables heterogeneous commit.

An SQL Query Block That Updates a Dynamic Server and an Application Server

The following transaction involves one Dynamic Server update site and one DRDA application-server update site:

```
DATABASE dbonline1@online1
BEGIN WORK
INSERT INTO dbgw1@gw1:"USR1".dr1orders
    SELECT * FROM dbonline1@online1:ol1orders
INSERT INTO dbonline2@online2:ol2employees
    SELECT * FROM dbgw1@gw1:"USR1".dr1employees
COMMIT WORK
```

Data from table **ol1orders** at Dynamic Server site **online1** (database **dbonline1**) is inserted into table **dr1orders** at Gateway application-server site **gw1** (*alias_RDB_name* **dbgw1**). Also, data from table **dr1employees** at Gateway application-server site **gw1** (*alias_RDB_name* **dbgw1**) is inserted into table **ol2employees** at Dynamic Server site **online2** (database **dbonline2**).

This is a type-3 transaction and can be committed only if **online1** supports heterogeneous commit, and heterogeneous commit is enabled for **online1.** Otherwise, the Gateway returns error code -29051 when it attempts to commit the transaction.

An SQL Query Block That Updates Two Application Servers

The following transaction involves two Gateway application-server update sites and two Dynamic Server read-only sites:

```
DATABASE dbonline1@online1
BEGIN WORK
INSERT INTO dbgw1@gw1:"USR1".dr1orders
    SELECT * FROM dbonline1@online1:ol1orders
INSERT INTO dbgw2@gw2:"USR1".dr2employees
    SELECT * FROM dbonline2@online2:ol2employees
COMMIT WORK
```

Data from table **ol1orders** at Dynamic Server site **online1** (database **dbonline1**) is inserted into table **dr1orders** at Gateway application server site **gw1** (*alias_RDB_name* **dbgw1**). Also, data from table **ol2employees** at Dynamic Server site **online2** (database **dbonline2**) is inserted into table **employees** at Gateway application-server site **gw2** (*alias_RDB_name* **dbgw2**).

This is a a type-4 transaction and the Dynamic Server returns error code -440 when the second INSERT statement is executed.

Accessing Views

To ensure that views are correctly identified, use base-table names that are explicitly qualified with owner names when you create views that are accessed in a distributed query. (Refer to "Creating Database Objects" on page 6-25.) If you do not use explicitly qualified table names, the user ID that the coordinating Dynamic Server uses to qualify the base-table names in the view text might be incorrect.

Using ANSI Outer Joins Involving Informix and DB2 Tables

Informix Enterprise Gateway with DRDA supports ANSI outer joins that involve Informix tables and DB2 tables. The following subsections define outer joins and describe various scenarios for the outer joins.

Outer Joins Versus Simple Joins

A join occurs when two or more tables are connected by one or more columns in common, creating a new table of results. A simple join treats two or more tables that are involved in a join operation equally. An outer join, however, treats two or more joined tables asymmetrically. An outer join makes one of the tables dominant over other tables, which are subservient.

In a simple join, the result contains only the combinations of rows from the tables that satisfy the join conditions. Rows that do not satisfy the join conditions are discarded. In an outer join, the result contains the combinations of rows from the tables that satisfy the join conditions. In addition, the result preserves rows from the dominant table that would otherwise be discarded because no matching row was found in the subservient table. The dominant-table rows that do not have a matching subservient-table row receive nulls for the columns of the subservient table.

Syntax for Outer Joins

Two forms of syntax are available for creating outer joins:

- Informix Extension Syntax
- ANSI join syntax

The ANSI-SQL Standard syntax implementation provides more flexibility for creating queries. Informix recommends that you create new queries with ANSI-SQL syntax. You must use the same form of syntax for all outer joins in a single query block. That is, in a single query block, you cannot begin a new outer join using a different form of syntax.

Examples of ANSI Join Syntax

This section show different scenarios for using ANSI outer joins and gives examples of ANSI join syntax.

The following example illustrates a join of an Informix table and a DB2 table where the Informix table is the dominant table. Consider table **t1** that has two integer columns **c1** and **c2** in Informix database **d1** in the instance LSERV. Also consider table **t2** that has two integer columns **c1** and **c2** in the DB2 database **d2**. Let the collection ID for the remote server be COL1, and let the Gateway used to connect to the remote DB2 database be **gw1**. The Gateway supports a left outer join between the columns in tables **t1** and **t2** as the following example shows:

```
SELECT * FROM d1@LSERV:"informix".t1 a1
    LEFT OUTER JOIN d2@gw1:"COL1".t2 a2
    ON a1.c1 = a2.c1;
```

The following example illustrates a join of an Informix table and a DB2 table where the DB2 table is the dominant table. Consider table **tab1** that has two integer columns **cola** and **colb** in informix database **d1** in the instance LSERV. Also consider table **tab2** that has two integer columns **cola** and **colb** in the DB2 database **d2**. Let the collection ID for the remote server be COL1, and let the Gateway used to connect to the remote DB2 database be **gw1**. The Gateway supports a left outer join between the columns in tables **tab1** and **tab2** as the following example shows:

```
SELECT * FROM d2@gw1:"COL1".tab2 a1
    LEFT OUTER JOIN d1@LSERV:"informix".tab1 a2
    ON a1.cola = a2.cola;
```

The following example illustrates a join of two DB2 tables. Consider table **mytab1** that has two integer columns **colx** and **coly** in the DB2 database **d1**. Let the collection ID for this database be COL1 , and let the Gateway instance used to connect to this collection be **gw1**. Also consider table **mytab2** that has two integer columns **colx** and **coly** in the DB2 database **d2**. Let the collection ID for this database be COL2, and let the Gateway instance used to connect to this collection be **gw2**. The Gateway supports a left outer join between the columns in tables **mytab1** and **mytab2** as the following example shows:

```
SELECT * FROM d1@gw1:"COL1".mytab1 a1
    LEFT OUTER JOIN d2@gw2:"COL2".mytab2 a2
    ON a1.colx = a2.colx;
```

The following example illustrates the use of function mapping in ANSI outer joins. Consider this variant of the first example in this section. This variant includes a WHERE clause.

```
SELECT * FROM d1@LSERV:"informix".t1 a1
    LEFT OUTER JOIN d2@gw1:"COL1".t2 a2
    ON a1.c1 = a2.c1
    WHERE POW(a1.c1, 2) = POW(a2.c1, 2);
```

This statement incorporates the SQL function mapping feature within the ANSI outer join syntax. In this case the Gateway maps the Informix POW function to the IBM POWER function. For further information on SQL function mapping, see "Mapping Informix Built-in Functions" on page 6-76.

Using the GWVTOT Environment Variable with Views

Set the environment variable **GWVTOT** to any value to indicate to the Gateway that all remote objects at the DRDA application server that are views should be represented as tables to the coordinating Dynamic Server during distributed accesses that involve remote views.

One scenario when the Gateway administrator might want to set **GWVTOT** follows:

User1 at the application server has a table **tab1** that does not have any privileges granted to **public**. **User1** has a view **view1** based on **tab1** and the view **view1** has the Select privilege granted to **public**. In this scenario, when users other than **user1** access the view **user1.view1**, they should be able to select from it, but their access to the table **user1.tab1** should fail. This works as expected with the Gateway in direct mode.

In distributed mode, however, the coordinating Dynamic Server unfolds any remote view references to base-table references, and, if permissions on tables and views are set up as discussed earlier, the query fails with an `insufficient privileges` error at the application server because the user has required privileges only at the view level and not on the dependent objects themselves.

As a workaround, if you set the **GWVTOT** environment variable, the Gateway represents the view as a table to Dynamic Server so that Dynamic Server does not translate view references to table references. Setting **GWVTOT** hides the view information (the tables the view is based on and the statistics on the dependent objects) from Dynamic Server. This might result in inefficient query optimization by Dynamic Server. However, if query performance is very important, granting table-level privileges in addition to view-level privileges will obviate the need for setting **GWVTOT**, at the same time assisting the Dynamic Server optimizer with important statistical data on the dependent objects for better query optimization.

Using GWVTOT with the gwdba Utility

The **GWVTOT** environment variable affects the behavior of the **gwdba** utility during addition of views.

When GWVTOT Is Not Set in the gwdba Environment

By default **GWVTOT** is not set in the **gwdba** environment. Consider what happens when **GWVTOT** is not set.

When you add views into the Informix-style catalogs, the dependent tables that are associated with the view are also added into the Informix-style catalog on the application server.

When you refresh views in the Informix-style catalogs, the dependent tables that are associated with the view are also refreshed in the Informix-style catalog on the application server.

When GWVTOT Is Set in the gwdba Environment

Consider what happens when **GWVTOT** is set in the **gwdba** environment.

During addition of views into the Informix-style catalogs, the dependent tables that are associated with the view are *not* added into the Informix style catalog on the application server.

When you refresh views in the Informix-style catalogs, the dependent tables that are associated with the view are *not* refreshed in the Informix-style catalog on the application server.

Considerations for Setting and Unsetting GWVTOT

You should perform all operations on Informix-style catalogs with **GWVTOT** either set or not set.

If multiple Gateway installations access the same DRDA application server, the **GWVTOT** setting on all these installations should be identical.

If you perform some operations on Informix-style catalogs with **GWVTOT** set and some operations with **GWVTOT** not set, you might leave the Informix-style catalogs in an inconsistent state.

If you were not setting the **GWVTOT** environment variable previously and now decide to set **GWVTOT** for all the catalog operations, then take the following steps.

Setting GWVTOT

1. Set **GWVTOT** in the **gwdba** environment.
2. Purge all the user tables and views from the Informix-style catalogs with the **Purge-tables** option.
3. Add all the necessary tables and views to the Informix-style catalogs with the **Add-tables** option.

If you were previously setting the **GWVTOT** environment variable and now decide not to set **GWVTOT** for all the catalog operations, then take the following steps.

Unsetting GWVTOT

1. Unset **GWVTOT** in the **gwdba** environment.
2. Purge all the user tables and views from the Informix-style catalogs by using the **Purge-tables** option.
3. Add all the necessary tables and views to the Informix-style catalogs by using the **Add-tables** option.

If you were not setting **GWVTOT** before and decide not to set **GWVTOT**, no special action is necessary.

Performance Considerations

When you use the Gateway for distributed queries, unique performance considerations exist:

- If your Gateway administrator has not installed an Informix-style catalog on the application server, ask the administrator to install the catalog. The improvement in distributed query response time can be substantial (as much as 10 times faster), depending on the number of tables and indexes on your application server. The more tables and indexes on your application server, the faster your queries run when you switch to an Informix-style catalog.
- Make sure that the table statistics for tables you use in your queries are current. If you have added any indexes since the last time you updated the statistics, or if the sizes of your tables have changed substantially, then the Dynamic Server optimizer might make bad decisions based on out-of-date table statistics. Ask your Dynamic Server database administrator and your Gateway administrator to make sure that the statistics for your Dynamic Server tables and application-server tables are current.
- If your Gateway administrator has installed an Informix system catalog on your application server, make sure that the catalog includes the application-server tables that you are using.
- Do not switch among databases unnecessarily. Each time you close a database, the Gateway servers that the database uses to access application servers stop. Initial costs are associated both with starting a Gateway and with connecting to an application server. You do not want to pay these costs any more times than is absolutely necessary. Also, the Gateway caches statistical table information that it obtains from the application server. Consequently, subsequent requests from a Dynamic Server coordinator for information about a particular table are very fast. You can take advantage of the cached information if you do not unnecessarily close your database between queries that reference the same table.

Using the GWMAXROWS Environment Variable to Limit Rows Selected in a Query

Set the **GWMAXROWS** environment variable to limit the number of rows that users can select from remote tables. This feature, which discourages users from issuing `SELECT * from table` statements, is especially useful if the tables that are queried have an extremely large number of rows and you need to reduce network traffic.

The Gateway administrator can set **GWMAXROWS** to a reasonable value, such as `100`, to indicate the maximum number of rows that an application can fetch through the Gateway. When this limit is reached, the next FETCH statement returns error -29029 to the application. (See "Errors That the Gateway Returns" on page 6-69.) The Gateway implicitly closes the current cursor, and subsequent FETCH and CLOSE operations on this cursor fail unless the cursor is reopened.

Set **GWMAXROWS** to any positive integer. To unset the environment variable, specify zero or a negative value.

The Gateway administrator can specify the **GWMAXROWS** value in the Informix environment configuration files, thus providing the flexibility to change the value on a per-user basis when there is a valid need to overcome the specified limit.

Tip: *The number of rows that the Gateway actually receives from the application server might be greater than the* ***GWMAXROWS*** *value, because the operation depends on the relative position of the row in the receiving buffer as well as the size of the communications buffer.*

Using the GWCATALOG Environment Variable to Access Catalog Information

Use the **GWCATALOG** environment variable to control the source and amount of catalog information that the Gateway collects about objects in distributed queries.

Normally, you do not need to set **GWCATALOG** because the default behavior of the Gateway is usually acceptable. However, if distributed queries take too long to complete, the Gateway administrator can set **GWCATALOG** to see if this helps.

Prior to executing a distributed query, a Dynamic Server requests catalog information about objects in the distributed query from the Gateway. The Gateway, in turn, collects this information from the target application server. The Dynamic Server optimizer uses the collected information to select the most efficient way to process the query.

The two types of catalog information are column information and statistics information.

Column Information

Column information is collected for both tables and views. Column information is required by the Dynamic Server coordinator, so it is always collected by the Gateway.

Statistics Information

Statistics information is collected only for table and view objects. The Dynamic Server coordinator does not require this information, but if the information is available, the optimizer uses it.

If the native system catalog is used, the Gateway provides the following statistics information:

- Number of rows in the object
- Index statistics
  - Description of the indexes
  - Number of levels
  - Number of leaves
  - Number of unique keys in the first column (except in OS/400)
  - Degree of clustering (except in OS/400)

If an Informix catalog is used, the Gateway provides all of the preceding information as well as additional information. This information consists of the second maximum and second minimum values in the first column of indexes provided that the data type of the column is one of the following items: SMALLINT, INTEGER, SMALLFLOAT, FLOAT, DECIMAL, or DATE.

You can turn off collection of statistics information by the Gateway.

When an Informix-style catalog is not installed, the cost of gathering statistics information about objects in a distributed query might be more than the cost of executing the query with an imperfect access plan.

In addition, if the target application server contains a large number of objects in its system catalog, the time required to find information about a single object can be considerable (depending on how the catalog tables are indexed and accessed by the target application server). If the time that is required to gather statistics information is not cost-effective, you can turn off the collection of statistics information with the **GWCATALOG** environment variable.

The accuracy of the query optimization depends on how current the statistics information in the catalog is. For further information on keeping statistics information up to date, see "Keeping Statistical Information Current" on page 3-40.

Parameters and Values for GWCATALOG

The **GWCATALOG** environment variable can take two parameters separated by a colon:

- The first parameter refers to the catalog type with the assignments that the following table shows.

Value	Meaning
0	Try the Informix catalog; on failure, use the native catalog.
1	Use the Informix catalog.
2	Use the native catalog.

- The second parameter is optional and indicates whether to skip gathering index and statistics information. For example, setting **GWCATALOG** to `2:1` uses the native catalog and skips gathering index and statistics information.

If the **GWCATALOG** environment variable is not set, the Gateway avoids querying **systables** and **syscolumns** and uses a PREPARE or DESCRIBE of a `SELECT * FROM` *table* statement to get the basic column descriptions. This process assumes that the tables do not have any indexes and does not differentiate between views and tables.

The following table shows the valid **GWCATALOG** values and their meaning for Informix Enterprise Gateway with DRDA.

Value	Meaning
`0:0`	Try the Informix catalog. On failure, try the native catalog. Gather index and statistics information.
`1:0`	Use the Informix catalog. Gather index and statistics information.
`2:0`	Use the native catalog. Gather index and statistics information.
`0:1`	Try the Informix catalog. On failure, try the native catalog. Skip index and statistics information.
`1:1`	Use the Informix catalog. Skip index and statistics information.
`2:1`	Use the native catalog. Skip index and statistics information.

If the overall format of the **GWCATALOG** value is incorrect, the Gateway behaves as if **GWCATALOG** were not set and uses the default source for every type of catalog information.

For information on how to use the **Informix Catalog** menu of the **gwdba** utility to work with the Informix-style catalog, see Chapter 5.

Using GWREUSECACHE for the Dynamic Server Catalog Cache

Dynamic Server requests data-dictionary information on all objects that the Gateway accesses. Both Dynamic Server and the Gateway cache this information. To do this, the Gateway queries the system catalog on the target DRDA application server, then sends the information to the Dynamic Server.

Ordinarily, each Gateway process queries the system catalog only once for an object, the first time it accesses the object. This is the default behavior of the Gateway when **GWREUSECACHE** is not set. The reason for this default is that in a production environment objects do not undergo schema changes very often.

However, data-dictionary information for a given object could potentially change between two queries sent to the Gateway for the same object. For example, columns could be added, new indexes created, and so on. Multiple Gateway processes can also exist that access a given object. In these cases, you might want to update the cache information more than once.

Set the **GWREUSECACHE** environment variable to let Dynamic Server reuse the previously cached information**.** In this way, you can significantly improve the performance of distributed queries with multiple Gateway users.

The different settings for **GWREUSECACHE** are:

1. **GWREUSECACHE** is not set
2. **GWREUSECACHE** = 1
3. **GWREUSECACHE** = *dd hh:mm:ss*

Setting 1: GWREUSECACHE Is Not Set

When **GWREUSECACHE** is not set, the Gateway issues catalog queries for any object the first time it accesses the object. For subsequent accesses, information from the Gateway cache is returned to Dynamic Server.

Setting 2: GWREUSECACHE = 1

When **GWREUSECACHE** is set to `1` (or set, but not to any specific value), the Gateway lets Dynamic Server reuse the Dynamic Server cache entry for the object, if a cache entry exists in the Dynamic Server. Information from the Gateway cache is not returned to the Dynamic Server.

When this option value is specified, the database administrator must be aware that any schema changes that happen to a given object at the application server after the first Gateway sends the data-dictionary information to Dynamic Server are not updated in the Dynamic Server cache unless **GWREUSECACHE** is again reset in the Gateway environment.

Setting 3: GWREUSECACHE = dd hh:mm:ss

The `INTERVAL day(2) to second` format specifies a time interval value, in the format: *dd hh:mm:ss*. The current time stamp, or the time when the remote catalog was last queried, is stored along with data-dictionary information in the Gateway cache. At the time of the next request for schema information for the same object from Dynamic Server, the saved time stamp is compared with the current time and the user-configured interval. If the interval has elapsed, the data-dictionary information is obtained from the remote database and put in the cache. Also the current time stamp, as of the time the remote catalog is queried, is stored with the data-dictionary information.

When you specify `'0 00:00:00'` as the time interval value, the cache information on Dynamic Server and Gateway are refreshed every time Gateway receives a request from Dynamic Server for data-dictionary information on the same object. When you specify a value other than `1`, or a value not in the *dd hh:mm:ss* format, the value defaults to **Setting 1.** Also, a warning message is logged in the Gateway system log file and in the Gateway trace.

When **GWREUSECACHE** is set in the format *dd hh:mm:ss*, the database administrator must be aware that any schema changes to a given object at the application server during the *dd hh:mm:s* time interval after the Gateway sent the last data-dictionary information to Dynamic Server are not be updated in the Dynamic Server cache. The Dynamic Server and the Gateway cache are refreshed after the Gateway receives a new request from Dynamic Server for data-dictionary information after the expiration of the time interval.

Important: *If the option is set in the dd hh:mm:ss format, the "dd," "hh," "mm," and "ss" components are mandatory.*

The following table shows an example of how to use the **GWREUSECACHE** environment variable.

Value	Meaning
0 00:00:00	Refresh the cache each time.
2 00:00:00	Refresh after time interval of 2 days.
1 05:06:07	Refresh after time interval of 1 day, 5 hours, 6 minutes and 7 seconds.
0 00:05:00	Refresh after time interval of 5 minutes.

Using the GWTIMEOUT Environment Variable

The **GWTIMEOUT** environment variable is used to reduce resource use in environments where a large number of concurrent Gateway instances exist, but where only a handful are actively servicing client requests. The **GWTIMEOUT** environment variable is set to the number of minutes that the Gateway process remains idle after it services the last request. After waiting for this specified interval the Gateway process terminates.

If subsequent requests are made before the timer expires, the time-out interval is reset after every request to the Gateway.

Important: *You must be careful while you determine the time-out interval. A value that is too small might cause the Gateway to stop before a genuine request can arrive from the client. This occurrence could be caused by network delays.*

Possible Error Conditions and Recovery

The client can expect the following error messages if a request is made after the Gateway terminates.

- Direct mode

  ```
  -25582: Network connection is broken.
  ```

  After this error condition the client must reopen the target database with either the CONNECT or DATABASE statements before issuing any further SQL requests.

- Distributed mode

  ```
  -936: Error on remote connection, <gw_name>, conerr=--
  25582, ...
  ```

  When this error occurs, the application needs to rollback the current transaction, which might result in error -934. Subsequent queries involving the Gateway will result in another connection to the Gateway and another instance of Gateway is spawned transparently.

Using the GWUPPWD Environment Variable

The **GWUPPWD** environment variable allows users to enter passwords in either uppercase or lowercase. This simplifies the password administration tasks in the case where the application server requires uppercase passwords.

The Gateway selects the password from one of several sources, depending on the security configuration of the Gateway, and sends the password to the application server. Before sending this password to the application server:

- If **GWUPPWD** is not set, the selected password is sent unaltered to the application server.
- If **GWUPPWD** is set, the selected password is made uppercase prior to being sent to the application server.

Figure 5-8 on page 5-15 shows how **GWUPPWD** is used.

Tip: *Even with* ***GWUPPWD*** *set, no change occurs in the security enforcement mechanism.*

Error Handling

This section discusses how to handle errors that arise when you use the Gateway. The following topics are discussed:

- Errors that the Gateway returns
- The SQLWARN array
- Errors that the application server returns
- Errors recorded in the **gw.log** file
- Trace-event logging

Errors That the Gateway Returns

When the Gateway process detects an error, it returns an error code to the application. The application treats these errors in the same way errors from any other Informix database server are treated. For more information about error handling, see the *Informix Guide to SQL: Reference* or the *Informix Guide to SQL: Tutorial*. To see the text for all Informix error messages, use the **finderr** or **Find Error** utility or view *Informix Error Messages* in Answers OnLine.

The following section summarizes the error messages that are new in this release.

-29089 RDB password required when client user ('%s') is not authenticated by the Gateway.

This error occurs when the Gateway has not authenticated the client user ID (**gwd** is started with the **-b** option) and the application does not specify a password with the user ID.

Regardless of the security configuration of the remote database server, the Gateway expects to receive a password along with the user ID for the remote server when **gwd** is started with the **-b** option. In this case, the Gateway delegates user ID authentication to the remote server entirely.

Make sure that the password is supplied to the Gateway.

The SQLWARN Array

The Gateway can set the SQLWARN7 flag. When the seventh character of the SQLWARN array is set to W and the operation is other than opening or connecting to a database, then character substitution occurred during the operation. Character substitution occurs during code-set conversion when an input character does not exist in the output character set, and the defined conversion table specifies that a different character be substituted in the output string.

The Gateway sets SQLWARN3 to W to identify itself as a Dynamic Server because it represents a Dynamic Server catalog.

Errors That the Application Server Returns

If a statement that the application server executes fails, the application server returns an SQL Communications Area (**sqlca**) to the Gateway. However, the Gateway cannot pass an **sqlca** structure to the client application because the values in the fields of the **sqlca** do not correspond to the values of the Informix **sqlca**. Instead, the Gateway returns the following values:

- The **sqlcode** field of the (Informix) **sqlca** structure is set to -29000 to indicate that an error was reported by the application server.
- The **sqlerrm** field of the (Informix) **sqlca** structure contains parameters to the application-server message text defined for the error or warning that caused the **sqlcode** field of **sqlca** to be nonzero.

Format of Application-Server Errors

For errors that the application server returns, the **sqlerrm** field is composed of four elements, as the following example and list show:

```
[AS_sqlca.sqlcode AS_sqlca.sqlstate product_id], original_tokens
```

where

AS_sqlca.sqlcode is the original **sqlca.sqlcode** of the application server.

AS_sqlca.sqlstate is the original **sqlca.sqlstate** of the application server.

product_id is the product ID of the application server.

original_tokens are the tokens that the application server generates.

The Product ID

The *product_id* is the **prdid** value that the application server returns. The format of the **prdid** is not defined by the Gateway. However, IBM application servers currently all return a **prdid** with the format *pppvvrrm*, which the following list describes:

- *ppp* is a three-character product identifier with the following values:
  - DSN represents DB2
  - QSQ represents OS/400
  - ARI represents SQL/DS
- *vv* is a two-character version number.
- *rr* is a two-character release number.
- *m* is a one-character modification level.

Example of a Formatted Application-Server Error

Suppose the application server returns the following values:

- *AS_sqlca.sqlcode* = `-463`
- *AS_sqlca.sqlstate* = `52002`
- *product_id* = `DSN02030`
- *original_tokens* = *`name-of-column`*

In this case, the Gateway returns the following values to the client application:

- **sqlcode** field of **sqlca** = `-29000`
- **sqlerrm** field of **sqlca** = `[ -463 52002 DSN02030]`, *`name-of-column`*

The following Informix error message is associated with error code -29000:

```
Application server error (error-message)
```

where *error-message* is the character string in the **sqlerrm** field. Thus, in the previous example, the message would be as follows:

```
Application server error ( [-463 52002 DSN02030],
        name-of-column)
```

To interpret the error, you must look in the manuals for the specific application server that you are using. When you find error number -463, substitute *name-of-column* into the application-server error message.

Interpretation of the sqlerrm Field

The portion of the **sqlerrm** field that is inside the square brackets always uses six characters (right justified) for *AS_sqlca.sqlcode* and five characters (right justified) for *AS_sqlca.sqlstate*, so that an application can access the individual values with relative ease. The *product_id* is eight characters long (left justified). Figure 6-8 shows the fields of **sqlerrm**, where *b* indicates a space character.

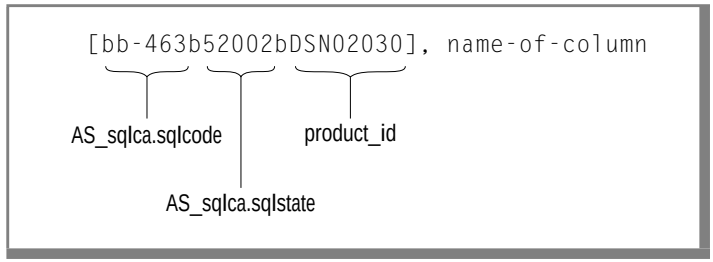


Figure 6-8
Illustration of the Fields of sqlerrm

Errors Recorded in the gw.log File

System error messages are specific to the Gateway system errors, the application-server system errors, and the NETWORK communication errors. These messages are written to the system log file, **$INFORMIXDIR/gw/log/gw.log**.

The Gateway logs system messages that include the following information:

- A user is connected to RDB.
- A user is disconnected from RDB.
- A network communication session cannot be created.
- A network communication conversation failed unexpectedly.
- A network error occurred unexpectedly.
- An unrecoverable error caused the application server to terminate.
- An unrecoverable error caused the Gateway to terminate the connection to the application server.
- A remote, distributed agent process at the application server terminated abnormally (ABENDed).
- An exception occurs in a response from the application server.

Unrecoverable errors that cause the connection to the application server to terminate have the prefix `Hard Error:` in the log-file message.

Sample Conversation Messages

The following example shows the **gw.log** file entries for the conversations that are used for a distributed connection when the **GWTWOCON** environment variable is set. The UNIX user ID **informix** is mapped to the *RDB_user_ID* **informix**. All entries refer to the UNIX user ID of the user running the application (**terryh**), even for the connection that is established for the catalog queries.

```
1996-05-02 14:09:08.077093 PID= 8112 Informix user "terryh"
(Unix user terryh) connected to RDB INFORMIXSOFTWARE as user
TERRYH
1996-05-02 14:09:09.304976 PID= 8112 Informix user "terryh"
(Unix user terryh) connected to RDB INFORMIXSOFTWARE as user
INFORMIX
1996-05-02 14:09:16.585710 PID= 8112 Informix user "terryh"
(Unix user terryh) disconnected from RDB INFORMIXSOFTWARE
1996-05-02 14:09:16.703197 PID= 8112 Informix user "terryh"
(Unix user terryh) disconnected from RDB INFORMIXSOFTWARE
```

The following example shows the **gw.log** entries for the conversation that is used for a direct connection. In contrast with the distributed connection in the previous example, this example has only one connect record and one disconnect record.

```
1996-05-02 14:10:32.221934 PID= 8279 Informix user "terryh"
(Unix user terryh) connected to RDB INFORMIXSOFTWARE as user
TERRYH
1996-05-02 14:10:50.602366 PID= 8279 Informix user "terryh"
(Unix user terryh) disconnected from RDB INFORMIXSOFTWARE
```

For more information about conversations, refer to "Distributed Queries" on page 1-9.

The Trace File

The **GWDEBUG** environment variable controls trace-event logging. You can set **DBTEMP** to specify a directory for the trace file. For more information about environment variables, refer to the *Informix Guide to SQL: Reference*. Trace-file data is written in the code set of the client application.

To turn on trace-event logging, set the **GWDEBUG** environment variable either in the Gateway daemon (**gwd**) environment before you start the daemon, or in the environment-configuration file at any time.

Using Trace Logging

The following table indicates the values and meaning for the **GWDEBUG** environment variable settings for the Gateway.

Value	Meaning
1	Enables the Gateway to trace the **sqlca** that the application server returns.
2	Enables the Gateway to trace the **sqlda** that the application serve returns.
3	Enables the Gateway to trace the distributed data-management (DDM) commands that are sent to the application server, the DDM commands that are received from the application server, and the **sqlda** that the application server returns.

(1 of 2)

Value	Meaning
4	Enables the Gateway to trace the DDM datastream that is sent to the application server and the DDM datastream that is received from the application server.
6	Enables the Gateway to trace the SQL statements that the Gateway receives and the SQL statements that are sent to the application server.

(2 of 2)

When **GWDEBUG** = 1, the trace file records each **sqlca** that the application server returns. To specify multiple trace levels, separate the desired trace numbers with a comma. For example, set **GWDEBUG** to "1,2" to turn on both the 1 and 2 traces. To generate all the trace levels, set **GWDEBUG** to "1,2,3,4,6".

Use trace-event logging *only* for debugging. When the debugging trace is activated, it slows down the performance of the Gateway.

A separate trace file is generated for each Gateway session. If a given user requests multiple trace types, entries for the different types of traces are all written to the same file. The name of the trace file is **gwt.*euidpid***, where ***euid*** is the effective UNIX user ID and ***pid*** is the UNIX process ID of the particular Gateway process instance (**gw**).

If the **DBTEMP** environment variable is set, the trace file is created in the directory specified by **DBTEMP**. If **DBTEMP** is not set or if the directory it points to is not writable or accessible, the Gateway trace file is created in the **/tmp** directory. The value of the **DBTEMP** environment variable in the client environment does not affect the location where the Gateway trace file is created. The location depends only on the value of **DBTEMP** in the Gateway environment.

The Gateway trace outputs the communication facility used (SNA or TCP/IP) as part of the initial connection information. The Gateway environment variable settings are also logged in the trace file.

To stop trace logging, unset the **GWDEBUG** environment variable.

Tip: *Informix recommends that users set* ***GWDEBUG*** *in private environment-configuration files because this allows restriction of the tracing on a per-user basis and does not affect all Gateway users that a Gateway daemon serves.*

The time stamps printed in the system log and trace files use the time zone in effect at the Gateway computer. If the environment variable ***TZ*** *is not set in the Gateway daemon environment, the time zone defaults to GMT.*

Mapping Informix Built-in Functions

Informix Enterprise Gateway with DRDA maps certain Informix built-in functions to their respective DB2 equivalents even though the DB2 equivalents have different names. You can use Informix built-in SQL functions that have corresponding DB2 equivalents with a different name and identical behavior in both direct and distributed mode.

In addition, you can specify Informix behavior for those built-in functions that have the same name as the DB2 function but have different behavior. You achieve this result by setting the **GWIFXSEMANTIC** environment variable. For further information about this environment variable, see "Specifying Informix Behavior for SQL Functions" on page 6-78.

Mapping Functions to DB2 Equivalents with Different Names

The following list shows the Informix built-in functions that are mapped to DB2 equivalents that have different function names.

- ABS
- LOGN
- OCTET_LENGTH
- POW
- ROOT

  This function is mapped in the default case (the single-parameter case) and in the two-parameter case when the second parameter has a value of `2` or `2.0`.

- ROUND

  This function is mapped in the default case (the single-parameter case).
- TODAY
- TRIM
- TRUNC
- WEEKDAY

Examples of Function Mapping

Consider the example of the Informix built-in function POW. The equivalent DB2 built-in function for this function is POWER. The POWER function is supported on DB2 for OS/400 version 3.2 and above.

Direct Mode

Suppose **tab1** is a table in DB2 for OS/400 V3.2 and this table has an integer column **cola**. Then the query

```
select * from tab1 where POW(cola,2) < 100
```

is mapped to

```
select * from tab1 where POWER(cola,2) < 100
```

Distributed Mode

Suppose COL1 is a collection ID that contains a table **tab1** with an integer column **cola** in the DB2 database **mydb1**. Let **gw1** be the gateway that is used to connect to the database. Then the query

```
select * from mydb1@gw1:"COL1".tab1 where POW(cola,2) < 100
```

is mapped to

```
select * from "COL1".tab1 where POWER(cola,2) < 100
```

Limitations of SQL Functions

You need to be aware of certain limitations in the SQL functions. Whether they are mapped or not, the SQL functions support the input arguments that are described in the *Informix Guide to SQL: Reference*. For the arguments that are not supported by Informix built-in functions, the behavior of the function may be unpredictable.

As an example, consider the MOD function. The Informix built-in MOD function supports only integer arguments. However, the application server supports float values as well. The behavior of the MOD function for float values is unpredictable.

Specifying Informix Behavior for SQL Functions

You can specify Informix behavior for built-in functions that have the same name as their DB2 equivalents but different behavior. You specify Informix behavior by setting the **GWIFXSEMANTIC** environment variable. By default this environment variable is not set.

When this environment variable is set, it changes the SQL statements sent to the application server for function invocations that meet the following criteria:

- The name of the Informix function or literal is the same as that of its counterpart in the application server.
- The output of the application server function is different from the output of the Informix function or literal.

If **GWIFXSEMANTIC** is set, the Gateway adjusts input parameters or the returned value of the application server function invocation or both so that the returned value within the SQL statement is the same as what the Informix function or literal would return.

Functions Affected by GWIFXSEMANTIC

When the **GWIFXSEMANTIC** environment variable is set, the following functions and literals reflect Informix behavior:

- CURRENT literal
- LENGTH function

Examples of Specifying Informix Behavior

Consider the Informix built-in LENGTH function. This function returns the number of bytes in a character column not including the trailing spaces. However, the DB2 LENGTH function returns the number of bytes in a character column including trailing spaces. So the Gateway needs to modify DB2 LENGTH(x) to LENGTH(STRIP(x,TRAILING)) in order to reflect Informix behavior.

The following examples of the LENGTH function assume that you have set the **GWIFXSEMANTIC** environment variable.

Direct Mode

Suppose **tab1** is a table in DB2 for OS/400 V3.2 and this table has a character column **cola**. Then the query

```
select * from tab1 where LENGTH(cola) < 5
```

is mapped to

```
select * from tab1 where LENGTH(STRIP(cola,TRAILING)) < 5
```

Distributed Mode

Suppose COL1 is a collection ID that contains a table **tab1** with a character column **cola** in the DB2 database **mydb1**. Let **gw1** be the Gateway instance that you use to connect to the database. Then the query

```
select * from mydb1@gw1:"COL1".tab1
    where LENGTH(cola) < 5
```

is mapped to

```
select * from "COL1".tab1
    where LENGTH(STRIP(cola,TRAILING)) < 5
```

Unmapped Functions

The Gateway passes any functions that are not listed in"Mapping Functions to DB2 Equivalents with Different Names" on page 6-76 or in "Functions Affected by GWIFXSEMANTIC" on page 6-79 to the Application Server without any modifications. However, you should be aware of possible limitations in these cases.

Consider the HEX function as an example. This function in the Informix implementation returns the hexadecimal equivalent of an integer, and so the input for this function is restricted to be an integer. However, for the same function, DB2 accepts other data types like Float and character strings as input. When the user tries to pass a literal other than type integer in a query in distributed mode, an error results since the Informix database server cannot compute the value of the function.

For more information on the limitations of unmapped functions, refer both to the *Informix Guide to SQL: Syntax* and to the DB2 SQL reference manuals.

Mapping Informix and IBM Data Types

Whenever possible, Informix data types are mapped to appropriate data types for the DB2, SQL/DS, and OS/400 application servers. You can, with some limitations, use most Informix data types with the Gateway. You cannot use the following data types with the Gateway:

- BYTE
- INTERVAL
- NCHAR
- NVARCHAR
- TEXT

Figure 6-9 shows the maps between Informix data types and DB2, SQL/DS, and OS/400 data types. The sections that follow the table give details about the maps of different data types.

Figure 6-9
Relationship Between Informix and DB2, SQL/DS, and OS/400 Data Types

Informix Data Type	DB2, SQL/DS, or OS/400 Data Type	Can Use with the Gateway?
BYTE	---	No
CHAR CHARACTER	CHAR	Yes
CHAR	GRAPHIC	Limited
CHAR	LONG VARGRAPHIC	Limited
NCHAR NVARCHAR	---	No
CHAR or VARCHAR	VARCHAR	Yes
TEXT	---	No
CHAR or VARCHAR	VARGRAPHIC	Limited
INT INTEGER	INT INTEGER	Yes
INTERVAL	---	No
MONEY	DECIMAL	Yes
DATE	DATE	Yes
DATETIME other than HOUR TO SECOND	TIMESTAMP	Yes
DATETIME HOUR TO SECOND	TIME	Yes

(1 of 2)

Informix Data Type	DB2, SQL/DS, or OS/400 Data Type	Can Use with the Gateway?
DEC DECIMAL NUMERIC	DEC DECIMAL NUMERIC	Yes
FLOAT DOUBLE PRECISION	FLOAT DOUBLE PRECISION	Yes
REAL SMALLFLOAT	REAL	Yes
SERIAL	INTEGER	Yes
SMALLINT	SMALLINT	Yes

(2 of 2)

Mapping Character Data Types

Informix, DB2, SQL/DS, and OS/400 databases all have CHAR and VARCHAR data types but differ in the lengths of the strings that they can handle. The following table shows the length limits of the data types.

Data Type	Informix Limit	Application Server Limits
CHAR	32,767	254
VARCHAR	254	Varies, but always greater than 254

The Gateway chooses maps for character strings that depend on the declared lengths of the strings. The following two tables show how the Gateway maps data types from Informix data types to application-server data types and vice versa. When single-byte characters are used, the following table applies.

Informix Data Type	Length (*n*, *m*)	Maps to Application-Server Data Type
CHAR(*n*)	*n* <= 254	CHAR(*n*)
	n > 254	VARCHAR(*n*)
VARCHAR(*n*)	All values of *n*	VARCHAR(*n*)
VARCHAR(*n*,*m*)	All values of *n* and *m*	VARCHAR(*n*)

When multibyte characters are used, the data can expand in length when it is converted from the code set of the application server into the code set of the client application. The following table applies when multibyte characters are used.

Application-Server Data Type	Length (*m*) Including the Expansion Factor	Maps to Informix Data Type
CHAR(*n*)	All values of *m*	CHAR(*m*)
VARCHAR(*n*)	*m* <= 254	VARCHAR(*m*)
	m > 254	CHAR(*m*)

In the preceding table, $m = n \times expansion\_factor$, where *expansion_factor* is the maximum inbound character-data expansion factor calculated for code-set conversion from the code set of the application server to the code set of the client application. (Refer to "Character-Data Length Expansion" on page 6-29.)

Mapping Graphic Data Types

You can use the Gateway to access data stored at the application server in GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC columns. When such data is fetched from an application server, the application-server data types are mapped to Informix CHAR and VARCHAR data types as the following table shows.

Application-Server Data Type	Length (*m*) Including the Expansion Factor	Maps to Informix Data Type
GRAPHIC(*n*)	All values of *m*	CHAR(2*m*)
VARGRAPHIC(*n*)	*m*<=127	VARCHAR(2*m*)
VARGRAPHIC(*n*)	127<*m*<=16383	CHAR(2*m*)
LONG VARGRAPHIC(*n*)	*m*<=127	VARCHAR(2*m*)
LONG VARGRAPHIC(*n*)	127<*m*<=16383	CHAR(2*m*)
LONG VARGRAPHIC(*n*)	*m*>16383	Not supported

In the preceding table, $m = n \times expansion\_factor$, where *expansion_factor* is the maximum inbound character-data expansion factor calculated for code-set conversion from the code set of the application server to the code set of the client application.

Mapping Integer Data Types

In Informix databases, the maximum negative values for both INTEGER and SMALLINT data types are reserved values and cannot be used. DRDA-compliant databases do not reserve the maximum negative values, therefore these values could be misinterpreted.

If the Gateway tries to fetch data from an INTEGER or SMALLINT column that contains the maximum negative value, the Gateway returns error -1214 (SMALLINT) or -1215 (INTEGER). This situation can occur only if the database values were entered by some means other than an Informix application or if the database values were manipulated arithmetically.

Mapping Floating-Point Data Types

Both REAL and FLOAT data types, by the nature of their hardware representation, are *approximate* data types. Informix, DB2, SQL/DS, and OS/400 databases all use 32-bit and 64-bit representations to store REAL (single-precision) and FLOAT (double-precision) data types.

Floating-Point Data Types on DB2 and SQL/DS

Most of the UNIX systems on which Informix software runs use the IEEE standard for floating-point representation, whereas DB2 and SQL/DS use the IBM S/370 floating-point representation.

The following table shows the internal formats of the IEEE and IBM S/370 representations.

Data Type	IEEE Format	IBM S/370 Format
REAL (single precision)	1(S)+8(E)+23(F)	1(S)+7(E)+24(F)
FLOAT (double precision)	1(S)+11(E)+52(F)	1(S)+7(E)+56(F)

In the preceding table, the letters S, E, and F in parentheses stand for the number of sine, exponent, and fraction bits in the representation.

Because of the difference in the number of fraction bits in the two representations, you can experience a loss of precision in the final one or two significant digits when floating-point values are transferred between an Informix application and a DB2 database.

Floating-Point Data on OS/400

Both Informix and OS/400 databases use the IEEE standard for floating-point representation, so precision is not lost when floating-point values are transferred between an Informix application and an OS/400 database.

Mapping DECIMAL and MONEY Data Types

Both the Informix DECIMAL(*p*,*s*) and MONEY(*p*,*s*) data types map to the DB2, SQL/DS, and OS/400 data type DECIMAL(*p*,*s*). The precision (*p*) and scale (*s*) specifications are optional. In Informix databases, the maximum value of *p* is 32. In DB2 and OS/400 databases, the maximum value of *p* is 31.

If you do not specify the scale parameter (*s*) in the Informix MONEY(*p*) data type, Informix database servers assume it to be 2. That is, the Informix MONEY(*p*) data type is equivalent to DECIMAL(*p*,2). Use the DECIMAL data type to store Informix MONEY data and explicitly specify the scale parameter as DECIMAL(*p*,2).

Mapping the SERIAL Data Type

DRDA-compliant databases do not have a SERIAL data type that automatically increments when a row is inserted. The Informix SERIAL data type maps to the DB2 INTEGER data type and is treated as an ordinary integer with no special characteristics.

Mapping DATE Data Types

The Gateway does not support conversion of DATE values that are expressed as character strings. The Gateway expects DATE values that are expressed as integers and returns DATE values that are expressed as integers. You must invoke the appropriate ESQL/C functions to convert to or from the integer representation and the character-string representation of DATE values.

Mapping TIMESTAMP Data Type to DATETIME

The TIMESTAMP data type of DB2, SQL/DS, and OS/400 represents date and time to a precision of six fraction digits in the seconds part. Informix DATETIME format can accommodate a maximum of five fractional second digits. Thus the sixth fractional digit in seconds (considered the least significant digit) is truncated when DB2 or OS/400 TIMESTAMP data is converted to Informix DATETIME data.

Expressing DATE and DATETIME Values in DATETIME Literal Formats

In SQL statements, express the DATE and DATETIME (or TIMESTAMP) values as Informix DATETIME literals.

For example, you should express the date August 16, 1999 as:

```
DATETIME (1999-08-16) YEAR TO DAY
```

You should express the time 3:35 P.M. as:

```
DATETIME (15:35:00) HOUR TO SECOND
```

You should express the moment August 16, 1999 at 3:35:34.3451 P.M. as:

```
DATETIME (1999-08-16 15:35:34.3451) YEAR TO FRACTION(4)
```

The Gateway converts Informix DATETIME literal values as follows:

- The DATETIME precisions YEAR TO YEAR, YEAR TO MONTH, YEAR TO DAY, MONTH TO MONTH, MONTH TO DAY and DAY TO DAY are converted to *yyyy-mm-dd* format.
- The DATETIME precisions HOUR TO HOUR, HOUR TO MINUTE, HOUR TO SECOND, MINUTE TO MINUTE, MINUTE TO SECOND and SECOND TO SECOND are converted to ISO format *hh.mm.ss*.
- All the other DATETIME precisions are converted to the format *yyyy-mm-dd-hh:mm:ss.fffff*.

Guidelines for Using a DB2 Database Server

Much of the discussion in this chapter applies to application servers in general, but some features might vary from one application server to another. This section discusses features and guidelines that explicitly apply to DB2 application servers. For more information, refer to the *IBM DB2 Version 2 SQL Reference*, Release 3 (SC26-4380).

Characteristics of a DB2 Database

The concept of a database is quite different on Informix and DB2 database servers. An Informix database server manages several databases that are quite distinct. Each Informix database, for example, has its own group of tables and its own system catalog tables.

A DB2 database server, on the other hand, has only one set of system catalog tables. All the tables within the DB2 space are equivalent. A DB2 database is merely a convenient way to refer to a group of tables. Figure 6-10 illustrates the different structures.

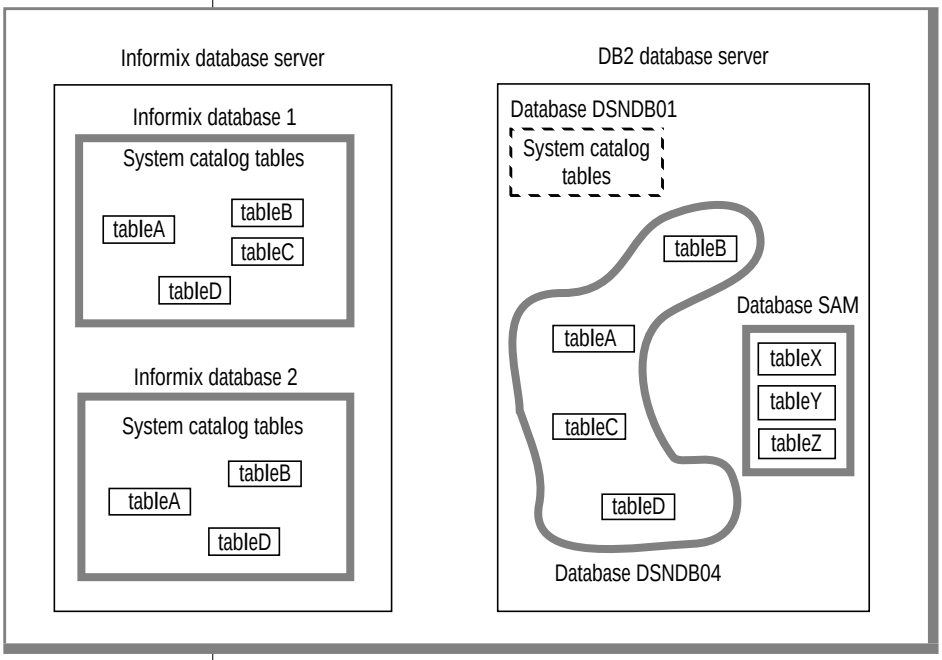


Figure 6-10 *Contrasting an Informix Database Server with a DB2 Database Server*

Some DB2 tables are members of the **SAM** database. The rest of the tables, which are not assigned to a specific database, belong to the default database for tables, **DSNDB04**. The DB2 system catalog tables belong to the default database **DSNDB01**.

DB2 Collection IDs and Packages

For a DB2 application server, collection IDs refer merely to a logical grouping of packages on the DB2 system. When you use DB2 with the Gateway, you usually need only one collection ID. You bind packages in a collection ID with the **gwdba** utility. Refer to "The Bind-Package Screen" on page 5-17.

Table-Naming Conventions

In a DB2 database, tables can have either a two- or a three-part name, as the following example shows:

- *owner.table*
- *location.owner.table*

Two-Part Table Names

Two-part table names follow standard ANSI conventions: the first part of the name is the owner of the table, and the second part is the actual name of the table. If the owner name is omitted, the implicit table name is as the following example shows:

```
currentUserID.table
```

Three-Part Table Names

Three-part table names use a feature known as *DB2 system-directed access*. DB2 supports a distributed unit of work with system-directed access, which is a DB2 private protocol (that is, not the DRDA protocol). You cannot use three-part names in a distributed query that uses the distributed-access mode of the Gateway.

In the DB2 private protocol, you access a remote site with a three-part table name in an SQL statement. The application server specified by *location* in the three-part name executes the SQL statement.

In Version 2, Release 3, DB2 introduced remote unit-of-work support with the DRDA protocol, and also allowed the two protocols to mix.

An Informix client application that uses direct access can connect to a DB2 application server and issue an SQL statement that contains a three-part table name. The SQL statement is then passed to the DB2 location specified in the table name. This process is sometimes referred to as a *two-hop* scenario because a DRDA *hops* to the first DB2, and then a system-directed-access *hops* to the second DB2, as the following example shows:

```
DATABASE 'my_RDB_alias@my_gateway_name'
.
.
.
SELECT * FROM LOCATION2.OWNER2.TABLE2
.
.
.
INSERT INTO LOCATION3.OWNER3.TABLE3 VALUES (3, 7, 'billy')
```

Three DB2 application servers are accessed in this example. The first DB2 is accessed from the Gateway with DRDA protocols when the DATABASE statement is executed. The second and third DB2 application servers (LOCATION2 and LOCATION3) are accessed from the first DB2 with DB2 system-directed access when the SELECT and INSERT statements are executed.

If all the DB2 application servers support multiple-site update, all the DB2 application servers can be updated within a unit of work. Otherwise, only one DB2 application server can be updated within a unit of work. For more information about how to use system-directed access, refer to your DB2 programming manuals.

Guidelines for Using an OS/390 Database Server

This section discusses features and guidelines that apply to OS/390 database servers.

Using the GWKEEPDYNAMIC Environment Variable

You can use the **GWKEEPDYNAMIC** environment variable only when the Gateway is connected to DB2 for OS/390 Version 5.0 or higher. To use this feature you need to rebind the Gateway package with the **bind** option KEEPDYNAMIC=YES. To set the **bind** option KEEPDYNAMIC to YES, use the following DB2 command:

```
REBIND PACKAGE(COLLECTION_ID.PACKAGE_ID) KEEPDYNAMIC(YES)
```

When you bind or rebind a package with **bind** option KEEPDYNAMIC=YES, the DB2 server keeps dynamic statements past the commit points for an application process. The DB2 server holds the dynamic statements until one of the following events occurs:

- The application process ends.
- A rollback operation occurs.
- The application uses an explicit PREPARE statement with the same name.
- The number of dynamic statements to be held reaches the maximum limit set by the DB2 server.

Two advantages result from setting KEEPDYNAMIC to YES:

- When an application uses a package bound with KEEPDYNAMIC=YES, it does not have to PREPARE a previously prepared statement after a commit to execute it again.
- When an application uses a package bound with KEEPDYNAMIC=YES, hold cursors are automatically closed when the end of data is detected. So no separate message to close a hold cursor needs to be sent to DB2.

For both of these reasons, performance is usually enhanced when an application, such as the Gateway, uses a package bound with KEEPDYNAMIC=YES and takes advantage of the resulting performance improvements.

If you have rebound the Gateway packages with the KEEPDYNAMIC=YES option, and you want the Gateway to take advantage of the performance improvements that are now available, set the **GWKEEPDYNAMIC** environment variable in the Gateway environment.

The default behavior of the Gateway assumes that the Gateway packages are *not* bound with KEEPDYNAMIC=YES. You must not set the **GWKEEPDYNAMIC** environment variable in the Gateway environment when the Gateway packages are *not* bound with KEEPDYNAMIC=YES. Errors can result in this case because the Gateway and the DB2 server can become out of sync with respect to what statements are currently prepared, and what cursors are currently open.

Guidelines for Using an OS/400 Database Server

This section discusses features and guidelines that apply to OS/400 database servers. For more information, refer to the *IBM DB2/400 SQL Reference*, Version 3 or above (SC41-3612).

Characteristics of OS/400

Conceptually, an Informix database and an OS/400 collection are quite similar. Both Informix and OS/400 database servers manage logical groupings of data. Informix calls these groupings *databases*, and OS/400 calls the groupings *collections*. Each Informix database and each OS/400 collection, for example, has its own group of tables and its own system catalog tables.

OS/400 Collection IDs

On OS/400, the collection ID uniquely identifies a collection of tables and other database objects (including a system catalog for the collection). When you use OS/400 with the Gateway, you must choose one of these collections into which to bind the Gateway packages. You bind packages in a collection ID with the **gwdba** utility.

The OS/400 SQL statement CREATE COLLECTION is analogous to the Informix SQL statement CREATE DATABASE.

Table-Naming Conventions

OS/400 databases allow two- and three-part table names, as the following example shows:

- *collection.table*
- *server.collection.table*

If you omit the *collection* part of the table name, OS/400 uses the ID of the user running the SQL package at the application server for the collection name. If you omit the *server* part of the table name, OS/400 uses the name of the current application server (the *real_RDB_name*).

Using Nonjournaled Files with Unlogged Informix Databases

If you access a nonjournaled file from a logged Informix database, a ROLLBACK WORK statement will not roll back the updates on the DB2 side.

To update nonjournaled files on DB2, take the following steps:

1. Before binding the Gateway packages with **gwdba**, set the **GWMAPCS** environment variable to 2 in the Gateway environment. This action instructs the Gateway to bind its CURSOR STABILITY package with an OS/400 isolation level of NONE. For more information on the **GWMAPCS** environment variable, see "Access to Nonjournaled OS/400 Files" on page 5-29.
2. Before running your application, set the **GWISOLEVEL** environment variable to 3 in the Gateway environment. This action instructs the Gateway to use its CURSOR STABILITY package to execute statements. You mapped the CURSOR STABILITY package to the OS/400 isolation level of NONE in the preceding step. OS/400 permits updates to nonjournaled files because the updates are executed using the OS/400 isolation level NONE. For more information on the **GWISOLEVEL** environment variable, see "Using the GWISOLEVEL Environment Variable" on page 6-24.

Support for OS/400 3.1 Long Table and Column Names in Direct Mode

In direct mode, Informix Enterprise Gateway with DRDA supports DDL and DML statements that involve OS/400 3.1 tables, columns, indexes, and views that are longer than 18 characters. SQL statements that contain long object names are not parsed by the Gateway and are passed to the application server without any processing. Therefore, you should not expect the same Gateway functionality as described in "Processing by the Gateway" on page 6-10 when you use SQL statements that have long object names.

In distributed mode, Dynamic Server does not permit object names longer than 18 characters. If both direct- and distributed-mode access are required for AS/400 tables longer than 18 characters, Informix recommends that you create a view with names that do not exceed the limits of Dynamic Server object names.

The GWNOBITDATA Environment Variable

Use the **GWNOBITDATA** environment variable only when the Gateway is connected to an OS/400 application server. When this variable is set, and the Gateway is connected to an OS/400 application server, the Gateway associates the AS/400 System CCSID with OS/400 character data that is described by the OS/400 as having CCSID 65535 (0xFFFF). You can set **GWNOBITDATA** to any value; the Gateway only checks whether it is set.

For example, if **GWNOBITDATA** is set, the AS/400 system CCSID is 37, and the OS/400 sends a character string tagged with CCSID 65535 to the Gateway, the Gateway tags the character string with CCSID 37.

When the Gateway converts the character string to the character code set of the client application, it converts from CCSID 37 to the client-application CCSID or code-set number, with a code-set conversion table.

This environment variable can be useful when the Gateway accesses OS/400 tables and files that were created when the AS/400 System CCSID was set to 65535 (the default CCSID when shipped).

Unless the file-level CCSID of each file is changed, character data in these files and tables are tagged with CCSID 65535 when sent to the Gateway. This is an incorrect value for the character data. In the DRDA architecture (which uses the CDRA architecture for character-data description), CCSID 65535 designates bit data. Bit data does not represent characters (visual symbols) and it is passed to the client application unchanged. In contrast, character data with a CCSID other than 65535 *does* represent characters (visual symbols), and this character data is changed so that the client application receives the same visual symbols (A, B, and so on) that were sent from the application server. In this case, the character data represents character data, but because it is tagged with CCSID 65535, the Gateway will not convert it unless **GWNOBITDATA** is set.

When this environment variable is set, real bit data is also converted (changed) when fetched from the OS/400 system. If you use **GWNOBITDATA**, make sure that none of the tables and files accessed contains bit data (that is, columns declared with the FOR BIT DATA qualifier). Otherwise, the Gateway erroneously changes this data along with correctly converting true character data.

A better alternative to setting **GWNOBITDATA** is to change the file-level CCSID to the correct value for each file or table that the Gateway accesses. The AS/400 system administrator can do this. For most U.S. English installations, CCSID 37 is the correct CCSID. However, in cases where large numbers of files and tables are tagged with CCSID 65535, and where no real bit data is accessed, it might be more cost-effective to use the **GWNOBITDATA** environment variable.

The GWUSESYSCOLNAME Environment Variable

You can use the **GWUSESYSCOLNAME** environment variable only when the Gateway is connected to OS/400 Version 3.1 or higher. When **GWUSESYSCOLNAME** is set, and the Gateway is looking up catalog information about columns of an object in **qsys2.syscolumns**, the Gateway looks up the **SYS_CNAME (SYSTEM_COLUMN_NAME)** field instead of the **NAME (COLUMN_NAME)** field to find the column names of the object.

The two methods of creating files on OS/400 are:

- Using SQL

  When you create a file with SQL, the **NAME (COLUMN_NAME)** field in **qsys2.syscolumns** is populated with the field name, and the **SYS_CNAME (SYSTEM_COLUMN_NAME)** field in **qsys2.syscolumns** is populated with one of the following names:

  - The alternate name for the field (if the user supplied an alternate name with the column alias construct)
  - The field name itself
  - A system-generated name (if the field name is longer than 10 characters)

- Using data definition syntax (DDS)

  When you create a file with DDS, the **SYS_CNAME (SYSTEM_COLUMN_NAME)** field in **qsys2.syscolumns** is populated with the field name, and the **NAME (COLUMN_NAME)** field in **qsys2.syscolumns** is populated with one of the following names:

  - The alternate name for the field (if the user supplied one)
  - The field name itself

When the Gateway queries the system catalog for information about an object, it normally looks up the **NAME** field to find the column name of the object. But when the **GWUSESYSCOLNAME** and **GWCATALOG** environment variables are set, and the OS/400 version is 3.1 or higher, the catalog queries made by the Gateway look up the **SYS_CNAME (SYSTEM_COLUMN_NAME)** field in **qsys2.syscolumns** for the column names of the object.

If you create files by using DDS, you need to set the **GWUSESYSCOLNAME** and **GWCATALOG** environment variables to be able to use the actual name of a field (which has an alternate name or alias defined) in a distributed query.

Administering the Informix Catalog

If you are using an Informix catalog and need the column names to be used as they are stored in the **SYS_CNAME** field in **qsys2.syscolumns**, you must set **GWUSESYSCOLNAME** when you add or refresh the table with the **gwdba** tool in the Informix catalog. Setting **GWUSESYSCOLNAME** causes the name of the column to be read from the **SYS_CNAME** field in **qsys2.syscolumns**.

Distributed Query Processing

Use of the **GWUSESYSCOLNAME** environment variable in distributed query processing depends on the setting of the **GWCATALOG** environment variable.

Using the Informix Catalog for Catalog Information

If you use the Informix catalog (that is, if **GWCATALOG** is set to 0 or 1), the valid column names in the query are the names as stored in the Informix catalog. You do not need to set **GWUSESYSCOLNAME** during access of the Informix catalog. For further information on using the Informix catalog, see "Administering the Informix Catalog" on page 6-96.

Using the Native System catalog for catalog information

If you are using the native system catalog (that is, if **GWCATALOG** is set to 0 or 2) and you need the column names to be used as they are stored in **SYS_CNAME** in **qsys2.syscolumns**, you must set **GWUSESYSCOLNAME**.

GWCATALOG not Set

The **GWUSESYSCOLNAME** environment variable has no effect if **GWCATALOG** is not set. This is because when **GWCATALOG** is not set, the Gateway uses a PREPARE and DESCRIBE of a SELECT * FROM <tab> statement to get the basic column descriptions. The column name in the output of DESCRIBE from OS/400 appears as it is stored in the **NAME** field in **qsys2.syscolumns**.

Using SYS_CNAME and NAME Column Names Together

You can use column names as stored in the **SYS_CNAME** field in **qsys2.syscolumns** and column names as stored in the **NAME** field in **qsys2.syscolumns** together in the same application with different objects.

Assume that you create a table named **tab1** by using DDS. The column named **col1** in this table has the alternate name **column_one**. Also assume that you create a table named **tab2** by using SQL. The column named **col1** in this table has the alternate name **column_one.**

To use **tab1.col1** and **tab2.col1** together, either use the Informix catalog for both tables, or use the Informix catalog for one of the tables.

Using the Informix Catalog for Both Tables.

If you want to use the Informix catalog for both tables, take the following steps:

1. Add table **tab1** to the Informix catalog with **GWUSESYSCOLNAME** set.
2. Add table **tab2** to the Informix Catalog with **GWUSESYSCOLNAME** not set.
3. Set the **GWCATALOG** environment variable to look for catalog information only in the Informix catalog. That is, set **GWCATALOG** to the value 1:*n*.

Using the Informix Catalog for One of the Tables.

If you want to use the Informix catalog for one of the tables, take the following steps:

1. Add table **tab1** to the Informix catalog with **GWUSESYSCOLNAME** set. Alternatively, add table **tab2** to the Informix catalog with **GWUSESYSCOLNAME** not set.
2. Set the **GWCATALOG** environment variable so that the Gateway looks for catalog information first in the Informix catalog and then in the native system catalog. That is, set **GWCATALOG** to the value 0:*n*. If you did not add table **tab1** to the Informix catalog, set **GWUSESYSCOLNAME**.

Guidelines for Using an SQL/DS Database Server

This section discusses features and guidelines that apply to SQL/DS database servers. For more information, refer to the *SQL Reference for IBM VM Systems and VSE,* Version 3, Release 4 or above (SH09-8087).

Characteristics of an SQL/DS Database

An SQL/DS database is the entire set of database objects that the SQL/DS system manages. It has one system catalog that includes information about all the tables, views, indexes, and other objects in the database. Databases are further subdivided into dbspaces, which are logical groupings of sets of tables and their associated indexes. The two major types of dbspaces are public and private. Only the owner of the dbspace and users with DBA authority can access private dbspaces. Multiple users can access public dbspaces.

SQL/DS Collection IDs and Packages

In SQL/DS the *collection_ID* is the owner of the package. *Owner-name* and *collection_ID* mean the same thing in SQL/DS. Packages are stored in system dbspaces.

Table-Naming Conventions

SQL/DS identifies tables with two-part table names, as the following example shows:

```
collection_ID.table
```

If you omit the *collection_ID* part of the table name in an SQL statement, SQL/DS uses the user ID of the user who is running the SQL package at the application server as the *collection_ID* of the table.

Sample Programs

You can use a program written for Dynamic Server or SE with the Gateway if the program does not use any unsupported features. The two programs in the next sections illustrate how to move data from an application server without and with distributed processing.

The demo1.ec Program

The sample program that Figure 6-11 shows is a modified version of the program **demo1.ec** from the *INFORMIX-ESQL/C Programmer's Manual*. If the application server (for example, a DB2 database) has tables with the same schema as the Informix **stores_demo** database, you can use this program on the Gateway almost unchanged.

You must make the following changes to make the program usable on the Gateway:

- You must change the CONNECT statement to give both the Gateway name (*gwservername*) and the *alias_RDB_name* of the application server.
- You must examine your use of BEGIN WORK and COMMIT WORK. In this example, the BEGIN WORK statement was removed and the COMMIT WORK statement was included.

Figure 6-11
The demo1.ec Sample Program

```
#include <stdio.h>

/* The following line has been uncommented because      */
/* the database has transactions:                       */

EXEC SQL define TRANS;

EXEC SQL define FNAME_LEN 15;
EXEC SQL define LNAME_LEN 15;

main()
{
    EXEC SQL BEGIN DECLARE SECTION;
    char fname[ FNAME_LEN + 1 ];
    char lname[ LNAME_LEN + 1 ];
    EXEC SQL END DECLARE SECTION;

    printf( "DEMO1 Sample ESQL program running.\n\n");

 /* The database name has been changed for Gateway Use */

    EXEC SQL connect to "db2_stores_demo_alias@my_gw";
    EXEC SQL declare democursor cursor for
    select fname, lname
        into :fname, :lname
        from customer
        where lname > "C";

/* The following lines have been commented out, because */
/* application servers start a unit of work implicitly*/
/* (ANSI style transactions)                           */

/*      EXEC SQL ifdef TRANS; */
/*      EXEC SQL begin work; */
/*      EXEC SQL endif; */

    EXEC SQL open democursor;

    for (;;)
        {
        EXEC SQL fetch democursor;
        if (strncmp(SQLSTATE, "00", 2) != 0)
            break;
         printf("%s %s\n",fname, lname);
        }

    EXEC SQL close democursor;
```

```
    EXEC SQL disconnect current;

    EXEC SQL ifdef TRANS;
    EXEC SQL commit work;
    EXEC SQL endif;

    printf("\nProgram Over.\n");
}
```

An Example Using Distributed Processing

Figure 6-12 shows a program that issues a single distributed query to move data from DB2 to Dynamic Server.

Figure 6-12
Sample Program Using a Distributed Query

```
/*
 * Sample ESQL/C program to transfer data from a DB2
 * (or OS/400) table to an OnLine table via Gateway
 * using a distributed query.
 *
 * Usage:
 *        gwdemo <ol_db> <ol> <as_db> <gw> <rdbuid>
 * where
 * <ol_db> = Informix OnLine database name
 * <ol>    = Dbservername for Informix OnLine SQL Engine
 * <as_db> = Alias rdb name for DB2 (or OS/400) RDB
 * <gw>    = Dbservername for Gateway
 * <rdbuid>= Owner of DB2 table (Collection for OS/400 table)
 *
 *
 * Schema of table ALLTYPES used in this example:
 *    (DB2 or OS/400 table)      (OnLine table)
 *       C CHAR(8)                  C CHAR(8)
 *       S SMALLINT                 S SMALLINT
 *       I INT                      I INT
 *       R REAL                     R REAL
 *       F FLOAT                    F FLOAT
 *     DEC DEC(4,2)               DEC DEC(4,2)
 *     TS TIMESTAMP            TS DATETIME YEAR TO FRACTION(5)
 *      TM TIME                    TM DATETIME HOUR TO SECOND
 *      DT DATE                    DT DATE
 */

EXEC SQL include decimal;
EXEC SQL include datetime;

EXEC SQL BEGIN DECLARE SECTION;

char c[8+1];
```

```
    short int s;
    long int i;
    float r;
    double f;
    dec_t dc;
    datetime year to fraction(5) ts;
    datetime hour to second tm;
    date dt;

    static char conbuf[80], stmt[256];

    EXEC SQL END DECLARE SECTION;

    main(ac,av)
    int ac;
    char *av[];
    {
        if (ac != 6)
        {
        printf("Usage: %s <ol_db> <ol> <as_db> <gw> <rdbuid>\n",
    av[0]);
        exit(1);
        }

        sprintf(conbuf, "%s@%s", av[1], av[2]);

        /* Connect to Informix OnLine */
        EXEC SQL database :conbuf;

        /* Drop OnLine table */
        EXEC SQL execute immediate "drop table alltypes";
        EXEC SQL commit work;

        /* Create OnLine table */
        EXEC SQL execute immediate "create table
            alltypes(c char(8), s smallint,
            i int, r real, f float, dc dec(4,2),
            ts datetime year to fraction(5),
            tm datetime hour to second, dt date)";
        EXEC SQL commit work;

        /* Do the data transfer now */
       sprintf(stmt,"insert into alltypes(c,s,i,r,f,dc,ts,tm,dt)
       select c,s,i,r,f,dc,ts,tm,dt from %s@%s:\"%s\".alltypes",
            av[3], av[4], av[5]);

        EXEC SQL execute immediate :stmt;

        /* Commit changes */
        EXEC SQL commit work;
```

```
    /* Close connection */
    EXEC SQL close database;

    exit(0);
}
```

Using Informix Products with the Gateway

In general, products that work with Informix Enterprise Gateway with DRDA treat the Gateway as if it were an Informix database server. For a list of supported products, see "What Does the Gateway Support?" on page 1-5.

Additional Informix and third-party products might have been certified since this manual was printed. For the latest information on supported products, see the on-line release-notes file for the Gateway.

Products to Use for Ad-Hoc Queries

You can use any version of DB-Access with the Gateway for ad-hoc queries. You cannot use these products for database-management tasks such as database creation or deletion, schema editing, and retrieving table information because the SQL statements generated to implement these task generally do not work on application servers.

When you use DB-Access and INFORMIX-SQL with Informix Enterprise Gateway with DRDA, remember the following issues:

- INFORMIX-SQL versions earlier than 6.0 require that you use the Relay Module.
- Any statement that acquires a lock on any database object (especially system catalog tables) can severely affect other processes that try to access the same object. For example, a statement, such as CREATE TABLE, that obtains locks on the system catalog tables should be immediately followed by a COMMIT WORK statement to release the locks.
- The Gateway uses the Repeatable Read isolation level by default. If you want more concurrency, change the isolation level to Cursor Stability.

Using the DB-Access Utility

Not all DB-Access options work with the Informix Enterprise Gateway with DRDA. You can use only the following DB-Access main-menu suboptions:

- **Query-language** option: all suboptions
- **Connection** option: all suboptions
- **Database** option
  - **Select**
  - **Close**
- **Table option**: no suboptions supported
- **Session option**: has no suboptions

  Ignore the server type.

All other DB-Access suboptions produce unpredictable results because they generate Informix-specific SQL statements that might not be accepted by an application server. Other suboptions also might require that the Informix-style system catalog be dynamic (updated as soon as a table is created, dropped, altered, and so on), which it is not.

Selecting a Server and Database

The DB-Access SELECT DATABASE SERVER screen shows an alphabetical list of database and Gateway servers, taken from the **$INFORMIXDIR/etc/sqlhosts** file. To select a Gateway server from the list, highlight its entry and press RETURN. Use the arrow keys to move around in the list.

Tip: *The list that the SELECT DATABASE SERVER screen displays includes all the entries in the* ***sqlhosts*** *file, including the "alias_RDB_name" entries. However, you cannot select an "alias_RDB_name" from this page. You select the "alias_RDB_name" on the SELECT DATABASE display.*

Once the Gateway server is selected, DB-Access shows the SELECT DATABASE screen. The SELECT DATABASE screen shows an alphabetical list of entries in the *alias_RDB_name@gwservername* format. The *gwservername* value for every entry is the server name of the Gateway to which you are currently connected. The *alias_RDB_name* values represent different application server and connection-mode combinations from the current **$INFORMIXDIR/etc/sqlhosts** file. Choose one of the entries in the list to select an application server and a connection mode.

Once you are connected to the application server, you can use the **Query-language** option on the main screen to enter SQL statements for the application server to process.

Chapter

7

The bcheckgw Utility

In This Chapter

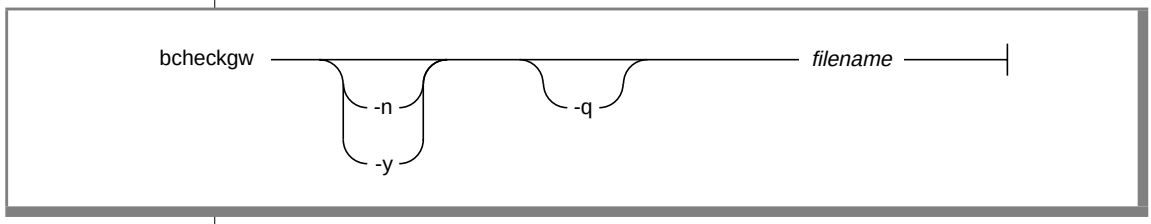
The **bcheckgw** utility is included in the Informix Enterprise Gateway with DRDA installation. Use **bcheckgw** to check the integrity of files created by the **gwdba** utility.

Using the bcheckgw Utility

The **bcheckgw** utility takes a table name as input and compares the data (**.dat**) and index (**.idx**) files to see if the two are consistent. Run **bcheckgw** whenever a system failure occurs or whenever you suspect the integrity of the **.idx** files. If the **.dat** files are corrupted, the data must be re-entered or the **.dat** files must be recovered from backups. The **bcheckgw** utility does not fix **.dat** files.

You can use **bcheckgw** to check the integrity of the **gwuser** and **gwbind** files that the **gwdba** utility creates. If bad entries are found, the prompts from **bcheckgw** request confirmation that you want to re-create the index. All data from the **.dat** file is read and the index is re-created and repaired.

Always run the **bcheckgw** utility from the directory **$INFORMIXDIR/gw/sysinfo**. The syntax is as follows:



Element	Purpose	Key Considerations
-n	Answers no to all questions.	None
-q	Suppresses printing of the program banner.	None
-y	Answers yes to all questions.	None
filename	Specifies the name of the table (**gwuser**).	None

If you do not specify either the **-n** or **-y** option, **bcheckgw** is interactive and waits for you to respond to each error that it finds.

Warning: *Use the* ***-y*** *option with caution. Do not run* ***bcheckgw*** *with the* ***-y*** *option if you are checking the files for the first time.*

Examples That Use the bcheckgw Utility

In Figure 7-1, **bcheckgw** checks all indexes for the **gwuser** table and finds no errors. For each index, **bcheckgw** prints a group of up to eight numbers. These numbers indicate the position of the key in each record.

Figure 7-1
Using bcheckgw to Check All Indexes for gwuser

```
bcheckgw -n gwuser

BCHECK C-ISAM B-tree Checker version 7.31.UC1
Copyright (C) 1981-1996 Informix Software, Inc.
Software Serial Number IFX#R000000

C-ISAM File: gwuser

Checking dictionary and file sizes.
Index file node size = 1024
Current C-ISAM index file node size = 1024
Checking data file records.
Checking indexes and key descriptions.
Index 1 = unique key (0,8,0) (8,18,0)
 1 index node(s) used -- 1 index b-tree level(s) used
Index 2 = duplicates (8,18,0)
 1 index node(s) used -- 1 index b-tree level(s) used
Checking data record and index node free lists.
5 index node(s) used, 0 free -- 5 data record(s) used, 16 free.
```

Figure 7-2 shows a sample run in which **bcheckgw** finds errors. The **-n** option is selected so that each question that **bcheckgw** asks is automatically answered `no`.

Figure 7-2
Sample bcheckgw Run That Shows Errors

```
bcheckgw -n gwuser

BCHECK C-ISAM B-tree Checker version 7.31.UC1
Copyright (C) 1981-1996 Informix Software, Inc.
Software Serial Number IFX#R000000


C-ISAM File: gwuser

Checking dictionary and file sizes.
Index file node size = 1024
Current C-ISAM index file node size = 1024
Checking data file records.
Checking indexes and key descriptions.
Index 1 = unique key (0,8,0) (8,18,0)
 1 index node(s) used -- 1 index b-tree level(s) used

ERROR: 1 missing data record pointer(s)
Delete index ? no

Index 2 = duplicates (8,18,0)
 1 index node(s) used -- 1 index b-tree level(s) used

ERROR: 1 missing data record pointer(s)
Delete index ? no

Checking data record and index node free lists.

ERROR: 1 duplicate data record pointer(s)
Fix data record free list ? no

5 index node(s) used, 0 free -- 5 data record(s) used, 16 free.
```

Because **bcheckgw** finds errors, you must delete and rebuild the corrupted indexes. In Figure 7-3, the **-y** option is specified so that all questions that **bcheckgw** asks are answered `yes`.

Figure 7-3
Using bcheckgw to Delete and Rebuild Corrupted Indexes

```
bcheckgw -y gwuser

BCHECK C-ISAM B-tree Checker version 7.31.UC1
Copyright (C) 1981-1996 Informix Software, Inc.
Software Serial Number IFX#R000000


C-ISAM File: gwuser

Checking dictionary and file sizes.
Index file node size = 1024
Current C-ISAM index file node size = 1024
Checking data file records.
Checking indexes and key descriptions.
Index 1 = unique key (0,8,0) (8,18,0)
 1 index node(s) used -- 1 index b-tree level(s) used

ERROR: 1 missing data record pointer(s)
Delete index ? yes

Remake index ? yes

Index 2 = duplicates (8,18,0)
 1 index node(s) used -- 1 index b-tree level(s) used

ERROR: 1 missing data record pointer(s)
Delete index ? yes

Remake index ? yes

Checking data record and index node free lists.

ERROR: 1 duplicate data record pointer(s)
Fix data record free list ? yes


ERROR: 1 missing index node pointer(s)
Fix index node free list ? yes

Recreating index node free list.
Recreating data record free list.
Recreating index 2.
Recreating index 1.
4 index node(s) used, 1 free -- 5 data record(s) used, 16 free.
```

Index

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X