Guide to Informix® Enterprise Replication

Informix Dynamic Server
Informix Dynamic Server, Developer Edition
Informix Dynamic Server, Workgroup Edition

Version 7.3
February 1998
Part No. 000-4350

Published by INFORMIX® Press

Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025-1032

Documentation Team: Evelyn Eldridge-Diaz, Geeta Karmarkar, Judith Sherwood

Table of Contents

Section II Preparing for Enterprise Replication

Chapter 4 Designing Enterprise-Replication Systems

Chapter 5 Setting Up Your Replication Environment

Section III Replicating Data

Chapter 6 Preparing Data for Replication

Chapter 8 Using Expert Mode

Chapter 10 Diagnosing Enterprise Replication

Chapter 11 Command-Line Utility

Section IV Appendixes

Introduction

Read this introduction for an overview of the information provided in this manual and for an understanding of the documentation conventions used.

About This Manual

This manual contains information to help you understand the concepts of data replication, design your own Enterprise Replication system, install Enterprise Replication, and administer and manage data replication throughout your enterprise.

The Informix Enterprise Replication feature of Informix Dynamic Server, Version 7.3, provides you with a cost-effective, efficient means to replicate data throughout your open-systems enterprise. Enterprise Replication is a client/server application that operates on UNIX and Windows NT. This application allows you to asynchronously replicate data throughout your enterprise. A graphical user interface (GUI) is provided to help you easily define, monitor, and control your replication system.

Types of Users

This manual is for database server administrators.

This manual assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts
- Some experience with database server administration, operating-system administration, or network administration

If you have limited experience with relational databases, SQL, or your operating system, refer to *Getting Started with Informix Dynamic Server* for a list of supplementary titles.

Software Dependencies

This manual assumes that your database server is one of the following products:

- Informix Dynamic Server, Version 7.3
- Informix Dynamic Server, Developer Edition, Version 7.3
- Informix Dynamic Server, Workgroup Edition, Version 7.3

Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a GLS (Global Language Support) locale.

This manual assumes that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for dates, times, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the *Informix Guide to GLS Functionality*.

Demonstration Database

The DB-Access utility, which is provided with your Informix database server products, includes a demonstration database called **stores7** that contains information about a fictitious wholesale sporting-goods distributor. You can use SQL scripts provided with DB-Access to derive a second database, called **sales_demo**. This database illustrates a dimensional schema for data-warehousing applications. Sample command files are also included for creating and populating these databases.

Many examples in Informix manuals are based on the **stores7** demonstration database. The **stores7** database is described in detail and its contents are listed in the *Informix Guide to SQL: Reference*.

The scripts that you use to install the demonstration databases reside in the **$INFORMIXDIR/bin** directory on UNIX platforms and the **%INFORMIXDIR%\bin** directory on Windows NT platforms. For a complete explanation of how to create and populate the **stores7** demonstration database, refer to the *DB-Access User Manual*. For an explanation of how to create and populate the **sales_demo** database, refer to the *Informix Guide to Database Design and Implementation*.

New Features

Most of the new features for Version 7.3 of Informix Dynamic Server fall into five major areas:

- Reliability, availability, and serviceability
- Performance
- Windows NT-specific features
- Application migration
- Manageability

Several additional features affect connectivity, replication, and the optical subsystem. For a comprehensive list of new features, see the release notes for the database server.

This manual includes information about the following new features:

- Shared connection threads: the ability to specify that certain connections share the threads that service the connection with other connections.
- Replication manager enhancements to support large environments
- Enhanced GLS support: the ability to replicate multiple locales within a single replication environment.

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other Informix manuals.

The following conventions are covered:

- Typographical conventions
- Icon conventions
- Command-line conventions

Typographical Conventions

This manual uses the following standard set of conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All keywords appear in uppercase letters in a serif font.
italics	Within text, new terms and emphasized words appear in italics. Within syntax diagrams, values that you are to specify appear in italics.
boldface	Identifiers (names of classes, objects, constants, events, functions, program variables, forms, labels, and reports), environment variables, database names, filenames, table names, column names, icons, menu items, command names, and other similar terms appear in boldface.
`monospace`	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
♦	This symbol indicates the end of feature-, product-, platform-, or compliance-specific information within a table or section.
→	This symbol indicates a menu item. For example, "Choose **Tools→Options**" means choose the **Options** item from the **Tools** menu.

Tip: *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after you type the indicated information on your keyboard. When you are instructed to "type" the text or to "press" other keys, you do not need to press* RETURN.

Icon Conventions

Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Comment Icons

Comment icons identify warnings, important notes, or tips. This information is always displayed in italics.

Icon	Description
	The *warning* icon identifies vital instructions, cautions, or critical information.
	The *important* icon identifies significant information about the feature or operation that is being described.
	The *tip* icon identifies additional details or shortcuts for the functionality that is being described.

Feature, Product, and Platform Icons

Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

Icon	Description
GLS	Identifies information that relates to the Informix GLS feature.
IDS	Identifies information that is specific to Dynamic Server and its editions. However, in some cases, the identified section applies only to Informix Dynamic Server and not to Informix Dynamic Server, Workgroup and Developer Editions. Such information is clearly identified.

(1 of 2)

Icon	Description
UNIX	Identifies information that is specific to the UNIX platform.
W/D	Identifies information that is specific to Informix Dynamic Server, Workgroup and Developer Editions.
WIN NT	Identifies information that is specific to the Windows NT environment.

(2 of 2)

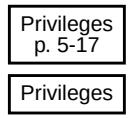
These icons can apply to a row in a table, one or more paragraphs, or an entire section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of the feature-, product-, or platform-specific information that appears within a table or a set of paragraphs within a section.

Command-Line Conventions

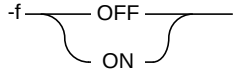
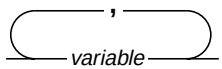
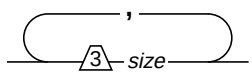
This section defines and illustrates the format of commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner 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 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. 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. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page.
	A shaded option is the default action.
	Syntax within a pair of arrows indicates a subdiagram.
	The vertical line terminates the command.

(1 of 2)

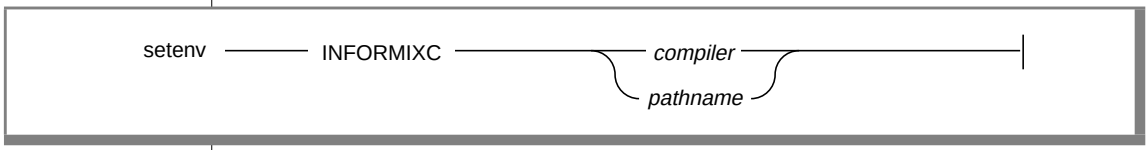
Element	Description
	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
	A gate (3) on a 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.

(2 of 2)

How to Read a Command-Line Diagram

Figure 1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

Figure 1
Example of a Command-Line Diagram



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

Figure 1 diagrams the following steps:

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 RETURN to execute the command.

Additional Documentation

For additional information, you can refer to the following types of documentation:

- On-line manuals
- Printed manuals
- On-line help
- Error message files
- Documentation notes, release notes, and machine notes
- Related reading

On-Line Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see the installation insert that accompanies Answers OnLine.

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order.

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

On-Line Help

The Replication Manager (the graphical user interface to Informix Enterprise Replication) contains context-sensitive help information about objects on the screen as well as general help topics.

Error Message Files

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions. For a detailed description of these error messages, refer to *Informix Error Messages* in Answers OnLine.

UNIX

To read the error messages under UNIX, you can use the following commands.

Command	Description
finderr	Displays error messages on line
rofferr	Formats error messages for printing

♦

WIN NT

To read error messages and corrective actions under Windows NT, use the **Informix Find Error** utility. To display this utility, choose **Start→Programs→Informix** from the Task Bar. ♦

In addition, error messages that are specific to Enterprise Replication are listed in an appendix of this manual.

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

UNIX

On UNIX platforms, the following on-line files appear in the **$INFORMIXDIR/release/en_us/0333** directory.

On-Line File	Purpose
EREPDOC_7.3	The documentation-notes file for your version of this manual describes features that are not covered in this manual or that have been modified since publication.
SERVERS_7.3	The release-notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
IDS_7.3	The machine-notes file describes any special actions that are required to configure and use Informix products on your computer. Machine Notes are named for the product described.

♦

WIN NT

The following items appear in the Informix folder. To display this folder, choose **Start→Programs→Informix** from the Task Bar.

Item	Description
Documentation Notes	This item includes additions or corrections to manuals, along with information about features that may not be covered in the manuals or that have been modified since publication.
Release Notes	This item describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.

Machine notes do not apply to Windows NT platforms. ♦

Related Reading

For additional technical information on database management, consult the following books. The first book is an introductory text for readers who are new to database management, while the second book is a more complex technical work for SQL programmers and database administrators:

- *Database: A Primer* by C. J. Date (Addison-Wesley Publishing, 1983)
- *An Introduction to Database Systems* by C. J. Date (Addison-Wesley Publishing, 1994).

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, 1993)
- *Understanding the New SQL: A Complete Guide* by J. Melton and A. Simon (Morgan Kaufmann Publishers, 1993)
- *Using SQL* by J. Groff and P. Weinberg (Osborne McGraw-Hill, 1990)

The learn more about the UNIX operating system, consider the following books:

- *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 for People* by P. Birns, P. Brown, and J. Muster (Prentice-Hall, 1985)
- *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 X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

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, please tell us about any corrections or clarifications that you would find useful.

Include the following information:

- The name and version of the manual that you are using
- Any comments that 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

Thank you. We appreciate your feedback.

Introducing Informix Enterprise Replication

Section I

Chapter

1

Introducing Data-Replication Concepts and Enterprise-Replication Features

Sharing corporate data is an important aspect of day-to-day business operations. Data replication generates and manages multiple copies of data at one or more sites, which allows an enterprise to share corporate data throughout its organization.

This chapter introduces the following information:

- Data-replication types
- Data-capture mechanisms
- Informix Enterprise Replication features

If you are familiar with data-replication concepts, skip to "Informix Enterprise Replication Features" on page 1-9.

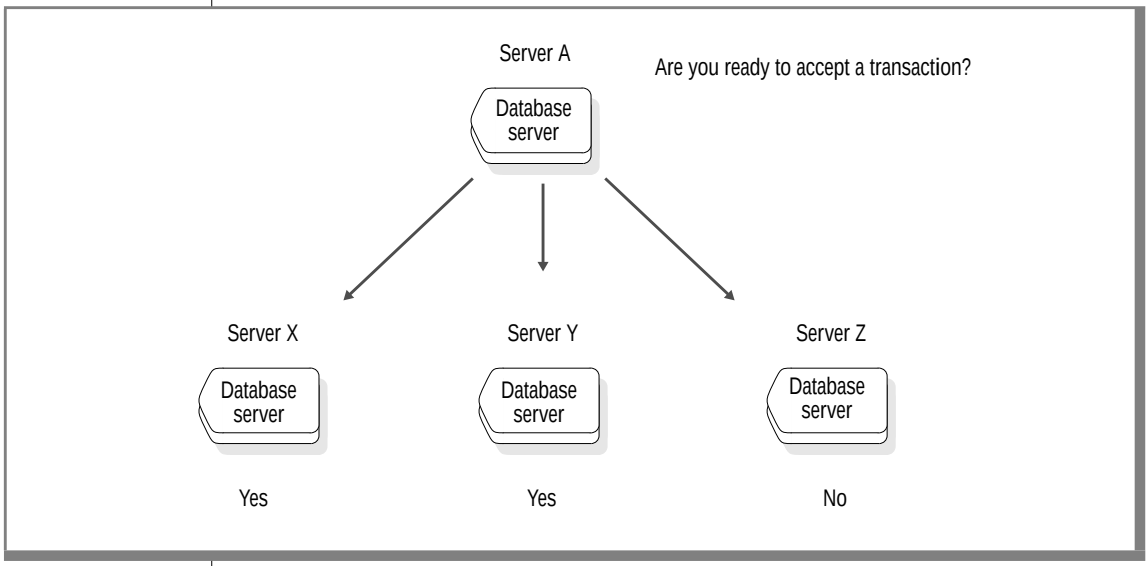
Data-Replication Types

The two types of data replication are synchronous and asynchronous. Each data-replication type provides different capabilities and strengths.

Synchronous Data Replication

In *synchronous* data replication, the replicated data is updated immediately when the source data is updated. Synchronous data replication uses the *two-phase commit* technology to protect data integrity. In a two-phase commit, a transaction is applied only if *all* interconnected distributed sites agree to accept the transaction. Figure 1-1 on page 1-4 illustrates the two-phase commit technology.

Figure 1-1
Two-Phase Commit Technology



In Figure 1-1, the transaction does not commit on *any* database server because Server Z either does not respond or is down. Synchronous data replication might be appropriate for applications that require immediate data synchronization. However, synchronous data replication requires that all hardware components and networks in the replication system be available at all times. It is difficult to guarantee the constant availability of all devices in a replication system, because hardware components and networks fail. An alternative and more reliable type of data replication is asynchronous data replication.

Asynchronous Data Replication

Asynchronous data replication updates the databases that reside at a replicated site after the primary database has committed the change. The delay to update the replicated-site databases can vary depending on the business application and user requirements. However, the data eventually synchronizes to the same value at all sites. The major benefit of this type of data replication is that if a particular database server fails, the replication process can continue and the original transaction is not directly affected by replication.

Asynchronous replication allows the following different ownership models:

- Primary-target

  All database changes originate at the primary database and are replicated to the target databases.

- Workflow

  A database is updated at one location, then the updates are passed to another location for update, then passed to another location for update, and so on.

- Update anywhere

  All databases have read and write capabilities. Updates are applied at all databases.

The update-anywhere model provides the greatest challenge in asynchronous replication. For example, if a replication system contains three replication sites that all have read and write capabilities, update conflicts occur when the sites try to update the same data at the same time. Update conflicts must be detected and resolved so that the data elements eventually have the same value at every site.

Asynchronous replication is the most common type of data replication in open-system environments because it protects against the system and network failures. Asynchronous data replication describes *how* data is replicated. Several mechanisms can be used to *capture* data in asynchronous replication.

Data-Replication Capture Mechanisms

The most frequently used data-replication capture mechanisms are trigger based and log based.

Trigger-Based Data Capture

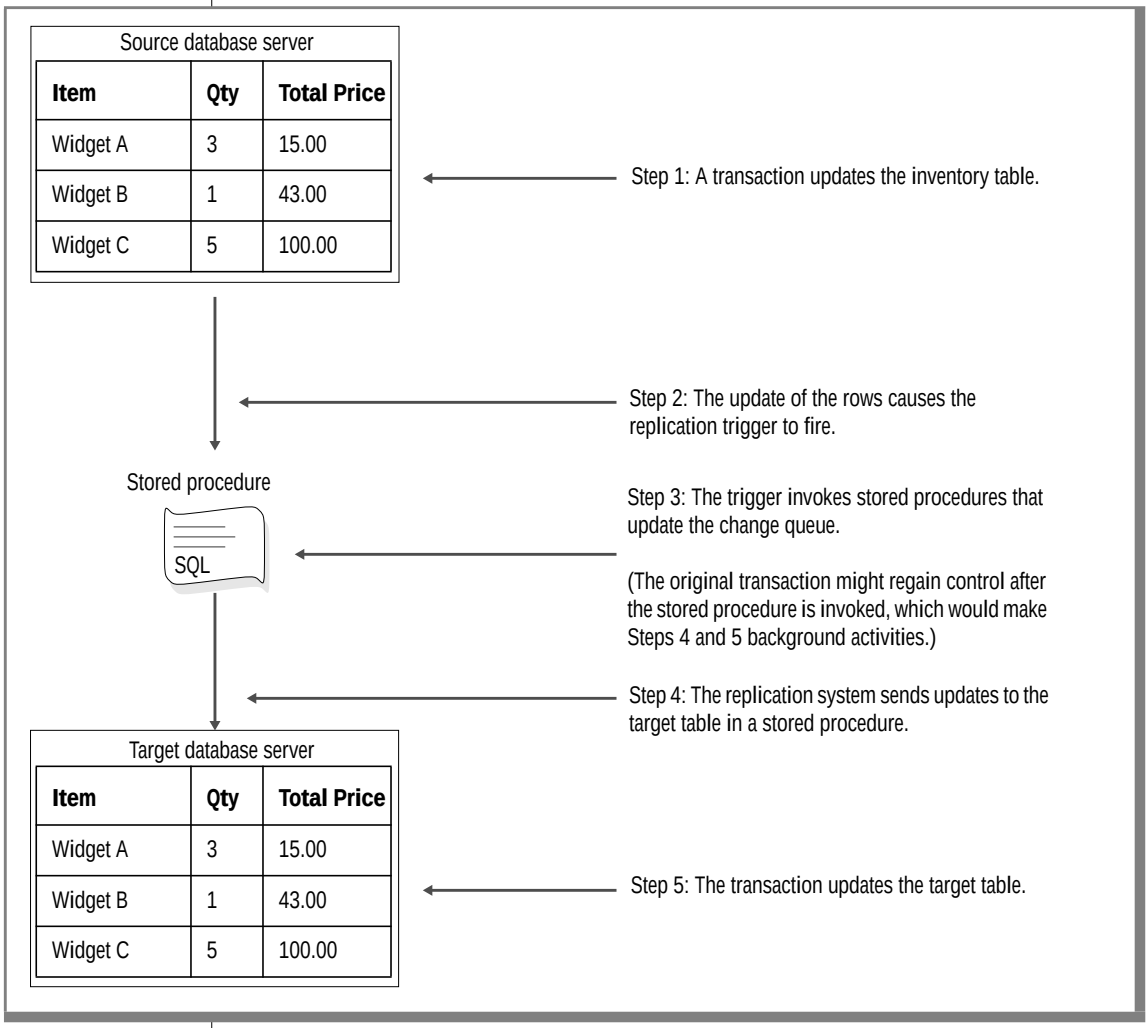
A trigger is like an alarm in the database that is associated with a piece of data. When a change is made to the data item, the trigger activates the replication process. The intent of this data-capture mechanism is to protect the referential integrity of the data. However, when a trigger begins to transfer modified data items, trigger-based data capture does not keep track of transaction information. Because trigger-based capture does not keep track of transaction information, customers and vendors must produce applications to ensure the referential integrity of the data.

Trigger-Based Transaction Capture

Trigger-based transaction capture does not perform replication based on triggered data, instead data changes are grouped into transactions. This data-capture mechanism assures the referential integrity of the data but adds additional work that can affect system performance.

In trigger-based transaction capture, a single transaction might trigger several replications if it modifies several tables, since the triggers are associated with the table. In this case, the trigger receives the whole transaction, but the stored procedure that captures the data runs as a part of the original transaction, thus slowing down the original transaction. Figure 1-2 on page 1-7 illustrates the effect trigger-based transaction-captured data has on performance.

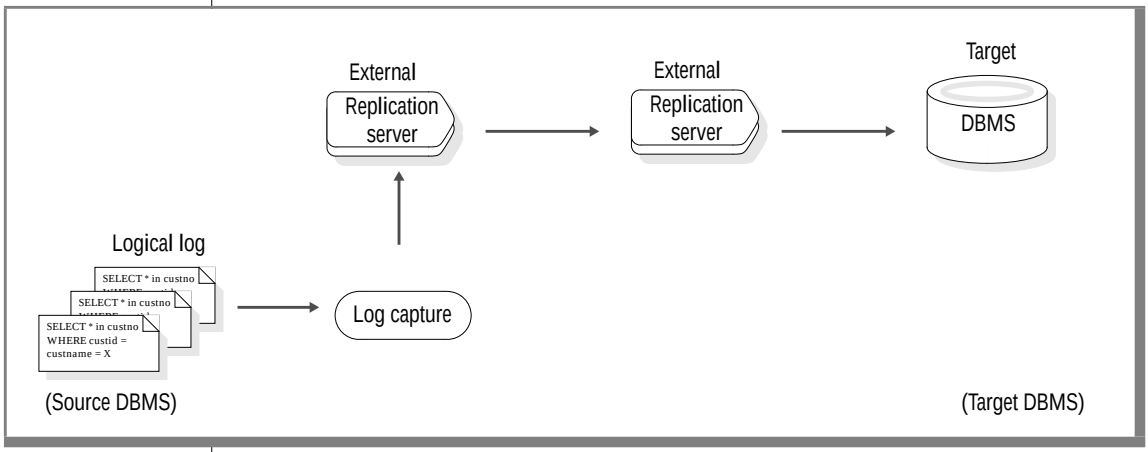
Figure 1-2
Trigger-Based Transaction Capture Affects the Performance of the Original Transaction



Log-Based Data Capture

Log-based data capture takes changes from the log and does not compete with transactions for access to production tables. Log-based data-capture systems operate as part of the normal database-logging process and add minimum overhead to the system. However, not all log-based systems are the same. Log-based systems that use external database servers and processes offset the advantages of log-based transaction capture because of the additional data-transfer overhead. Figure 1-3 shows a log-based data capture that uses external database servers for replication.

Figure 1-3
External Replication Database Servers Require Additional Data Transfer



In addition to whether external database servers are used, another element in a successful log-based replication system is whether every entry in the log needs to be read as part of the transaction-capture process. An efficient log-based transaction-capture mechanism is an integral part of the database system.

To replicate data throughout an open-systems enterprise is much more complex than to copy data from one database server to another. The most efficient and cost-effective data replication is an asynchronous replication that uses a log-based transaction-capture mechanism.

Informix Enterprise Replication Features

Many data replication products use the following key elements to successfully replicate data throughout an open-system enterprise:

- High performance
- High availability
- Consistent information delivery
- Flexible architecture
- Easy, centralized administration

This section associates the key elements of data replication with Enterprise Replication features.

Enterprise Replication Provides High Performance

A replication system must not burden the source of the data, must use networks efficiently, and must use all resources efficiently.

Enterprise Replication uses an efficient, log-based transaction-capture mechanism that is integrated in the database architecture. The database server can implement this capture mechanism efficiently because the capture mechanism is internal to the database. In addition, Dynamic Scalable Architecture (DSA) invokes a *parallel-everything* approach that uses the parallel-processing power of symmetric multiprocessing (SMP). All replication operations are performed in parallel, which further extends the performance of Enterprise Replication.

Enterprise Replication operates in a LAN, WAN, and combined LAN/WAN configuration across a range of network transport protocols. Enterprise Replication uses the standard database server communication facility to establish network connectivity among the Enterprise Replication database servers involved in replication.

Enterprise Replication Provides High Availability

A replication system must be reliable and must not expose business operations to computer system failures.

Enterprise Replication implements asynchronous data replication. Asynchronous replication ensures that network and target database server outages are tolerated. Enterprise Replication also uses a delivery mechanism that stores data locally and then sends the data to the remote database server in two separate transactions. Target database servers acknowledge receipt of a message when the message is *safely stored*. A message is safely stored when it is either in a queue or in the target database.

In the event of a database server or network failure, the local database server can continue to service the local users, which provides a high degree of fault tolerance.

Enterprise Replication Provides Consistent Information Delivery

A replication system must protect data integrity.

All Enterprise Replication transactions are stored in a reliable queue to maintain the consistency of transactions. Additionally, Enterprise Replication provides options to maintain the order of transactions when data is replicated. Maintaining the consistency and order of transactions ensures that if the same record is updated multiple times, the changes are applied to the remote databases in the same order as on the primary database, and applied only once at the target database server.

If update conflicts occur, Enterprise Replication provides built-in automatic conflict detection and resolution.

Enterprise Replication Delivers a Flexible Architecture

A replication system should not impose model or methodology restrictions on the enterprise. A replication system should allow replications based on specific business and application requirements.

Enterprise Replication supports all Informix data types across heterogeneous hardware. Enterprise Replication also supports the Global Language Support (GLS) feature, which allows Informix products to handle different languages, cultural conventions, and code sets. Additionally, Enterprise Replication supports ANSI-compliant databases.

Ownership Models

Enterprise Replication supports the following replication models:

- Primary target

  Unidirectional updates from a primary database server to many target database servers, or from many target database servers to a primary database server.
- Workflow

  Asynchronous updates that move from site to site.
- Update anywhere

  Peer-to-peer update capabilities, which allows users to function autonomously even if systems or networks are not available.

Operating Modes

Enterprise Replication provides two operating modes: standard and expert. The standard operating mode supports primary-target replication only. The expert operating mode supports all ownership models.

Network Topologies

Enterprise Replication supports the following network topologies:

- Fully interconnected

  Continuous connectivity between all participating database servers.
- Hierarchical tree

  A parent-child configuration that supports continuous and intermittent connectivity.
- Forest of trees

  Multiple hierarchical tree that connect at the root database servers.

For more information, see Chapter 3, "Understanding Enterprise Replication Objects, Topologies, and Processes."

Enterprise Replication Provides Easy, Centralized Administration

Administrators must be able to easily manage all the distributed components of the replication system from a single point of control.

From the Replication Manager, a Windows NT graphical user interface (GUI), an administrator can define and monitor the replication system. You do not need to shut down the entire replication system to add database servers to the replication system. The Replication Manager is documented in Chapter 7, "Administering Enterprise Replication," and Chapter 9, "Monitoring Enterprise Replication."

You can also administer the replication system from your system command prompt using the command-line utility (CLU). The CLU is documented in Chapter 11, "Command-Line Utility."

An Enterprise Replication global catalog (a system database table) on each database server keeps track of Enterprise Replication configuration information. The global catalog maintains information such as replication definitions, Enterprise Replication database servers, conflict detection and resolution rules, and their associated stored procedures. Any changes to replication definitions are automatically sent to other database servers in the replication system.

Chapter

2

Overview of Enterprise Replication

This chapter provides you with an overview of how Enterprise Replication replicates data. The first part of this chapter introduces a hypothetical customer, Elite Exporters, Inc., that uses Enterprise Replication. Background information includes a customer profile and the hardware and software configuration. The chapter concludes with a high-level overview of the major processes Enterprise Replication uses to replicate data.

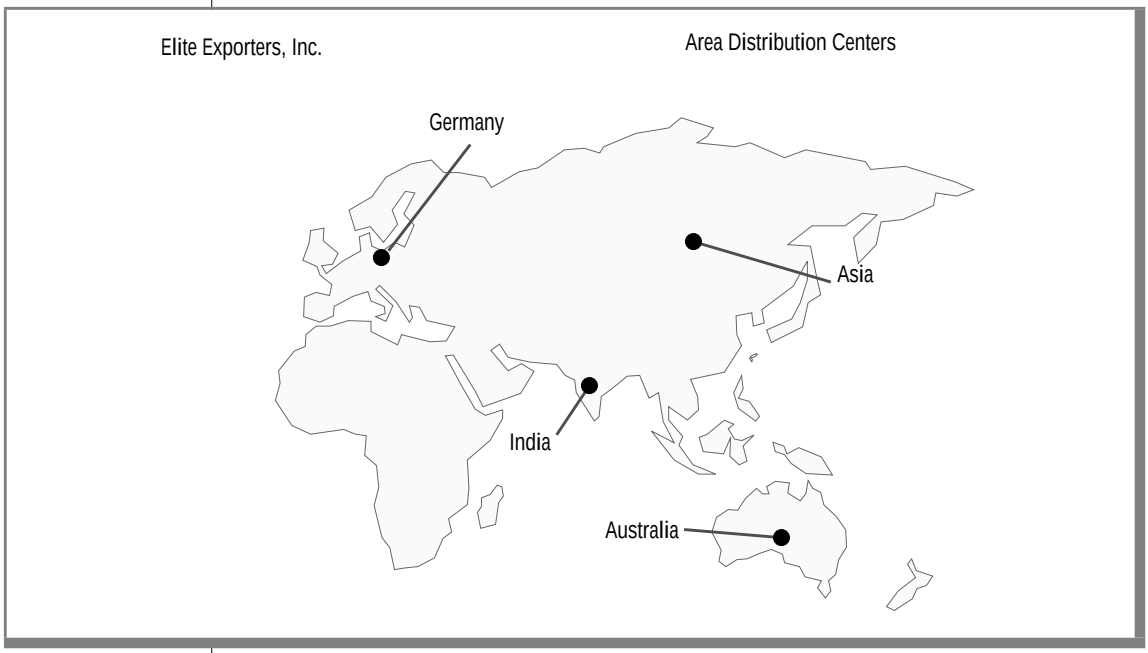
Important: *Elite Exporters, Inc. uses the advanced operating mode of Enterprise Replication. For a complete definition of operating modes, see* *Chapter 4, "Designing Enterprise-Replication Systems."*

Elite Exporters, Inc. Company Profile

Elite Exporters, Inc. is an international export company that specializes in exporting exemplary home merchandise to retailers throughout the world. Their products include the finest chinas, crystals, rugs, home electronics, and so on. Customers can order merchandise from one of four regional centers. These regional centers maintain a complete inventory of the merchandise available from the distributors in their area. A customer can also order merchandise from Elite Exporters headquarters in Germany, which contains a complete inventory of all available merchandise. Elite Exporters uses Enterprise Replication to replicate inventories throughout the enterprise.

Figure 2-1 shows the Elite Exporters regional centers: Asia, Australia, India, and Germany. The main office in Germany maintains an inventory of all available merchandise. The regional centers maintain inventories only for their area.

Figure 2-1
Elite Exporters, Inc.



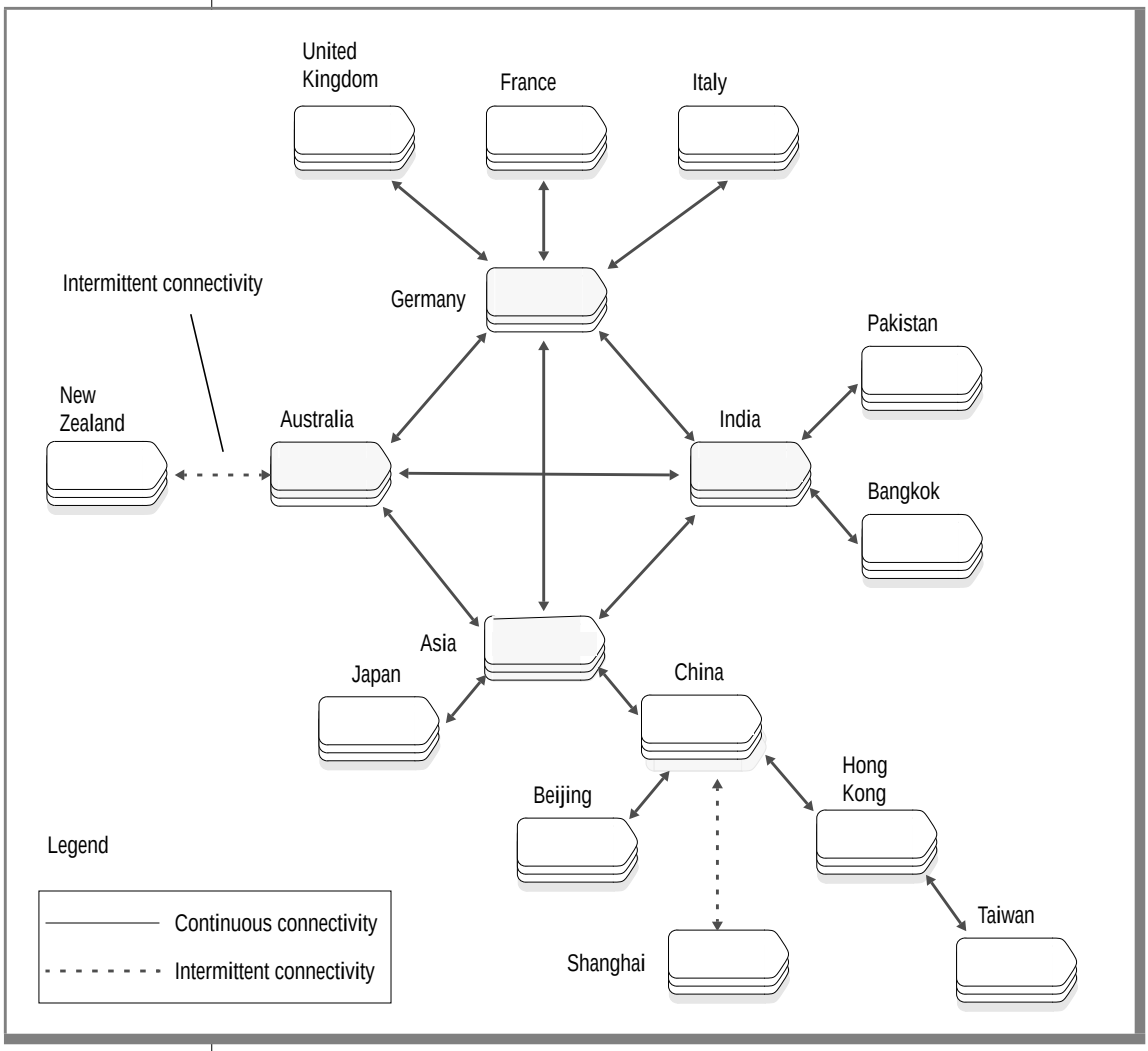
Elite Exporters Hardware and Software Configuration

The Elite Exporters, Inc. hardware configuration is as follows:

- 4 regional centers (Germany, Australia, India, Asia) that run UNIX operating systems.
- 12 distribution centers (United Kingdom, France, Italy, Pakistan, Bangkok, China, Beijing, Shanghai, Hong Kong, Taiwan, Japan, New Zealand) that run Windows NT operating systems.
- Appropriate networking connections.

Figure 2-2 illustrates the Elite Exporters, Inc. network configuration.

Figure 2-2
Elite Exporters, Inc. Hardware Configuration



Each distribution center maintains a database of its inventory. Each regional center maintains a database that contains the inventory of each distribution center in that area.

Figure 2-3 shows an example of the partitioned database residing on database server **India** that contains the India, Pakistan, and Bangkok inventories.

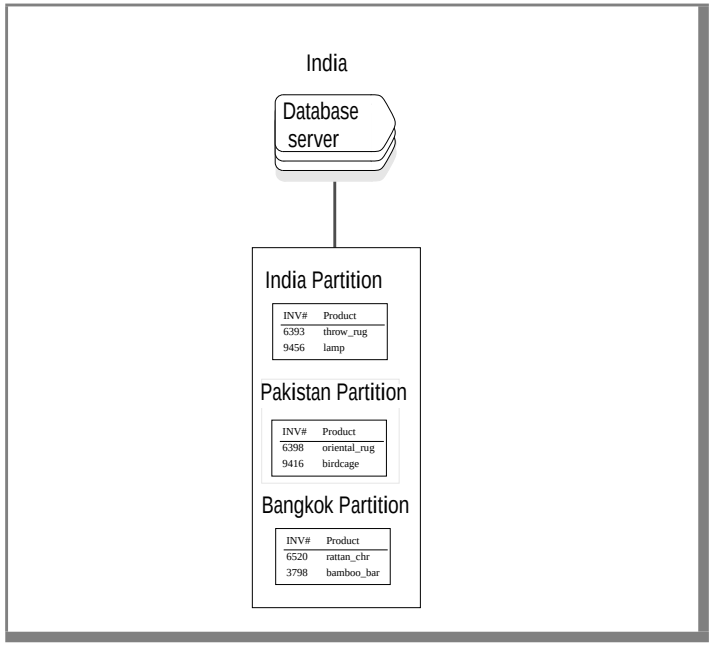
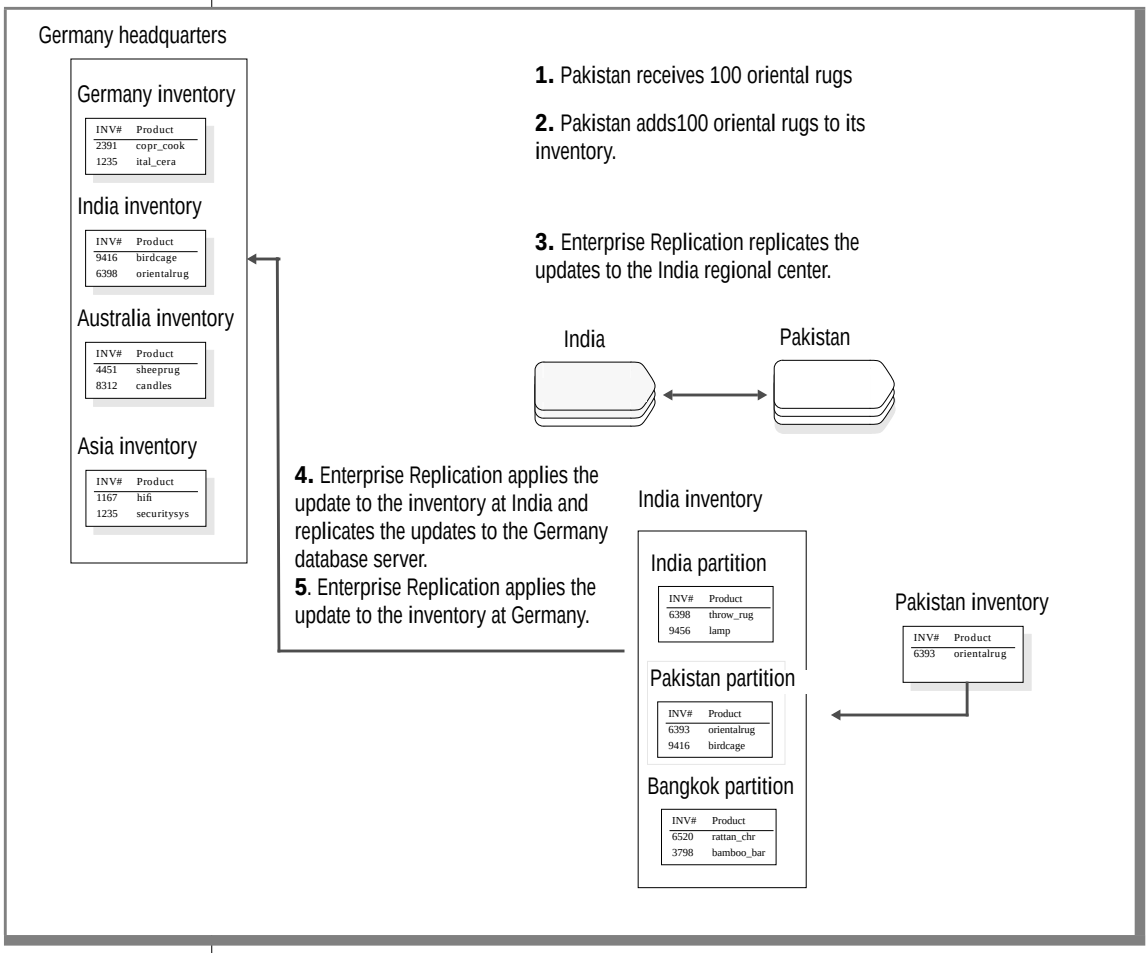


Figure 2-3
Partitioned Database Example

When a distributor in Pakistan or Bangkok debits or credits merchandise to its inventory, the distributor updates the local inventory database. Enterprise Replication sends the updates to the regional center in India. The **India** database server applies the updates to its inventory. The updates automatically replicate to the Germany headquarters.

Figure 2-4 shows an example of how data replicates throughout Elite Exporters, Inc.

Figure 2-4
A Transaction Example in Elite Exporters, Inc.



The next section provides a high-level overview of the processes Enterprise Replication uses to replicate data throughout Elite Exporters, Inc.

How Enterprise Replication Replicates Data

This section introduces the processes that Enterprise Replication uses to replicate data. Detailed information on each process is in Chapter 3, "Understanding Enterprise Replication Objects, Topologies, and Processes." Before you can replicate data, you must define a replicate to Enterprise Replication. For more information on how to define a replicate, see Chapter 7, "Administering Enterprise Replication." Enterprise Replication uses four major processes to replicate data:

- Capture transactions
- Evaluate data for replication
- Distribute data for replication
- Apply replicated data

Capturing Transactions

Enterprise Replication uses a log-based transaction-capture mechanism to capture transactions. A log-based transaction capture minimizes the effect on transaction performance; it captures changes from the log instead of competing with transactions that access production tables.

Enterprise Replication reads the logical log to obtain the row images for tables that participate in a replicate. (Enterprise Replication marks these rows when your transactions are logged.) These rows are passed to Enterprise Replication for further evaluation.

Evaluating Data for Replication

Enterprise Replication evaluates transactions based on a distinct time and a change to a final image of a row.

Evaluating Time

All Enterprise Replication commands are issued at distinct times. Enterprise Replication records the time the command is issued and determines which transactions have committed by that time and are therefore eligible for replication.

Evaluating Images of a Row

After Enterprise Replication evalutes the recorded time to determine whether a transaction should be replicated, Enterprise Replication evaluates the initial and final images of a row and any changes that occur between the two row images to determine which rows to replicate.

Processing in Parallel

Enterprise Replication evaluates the images of rows in parallel to assure high performance. The records that qualify are collected into a transaction list and then passed to the Enterprise Replication reliable-message queue for distribution. Figure 2-5 on page 2-10 illustrates how Enterprise Replication uses parallel processing to evaluate transactions for replication.

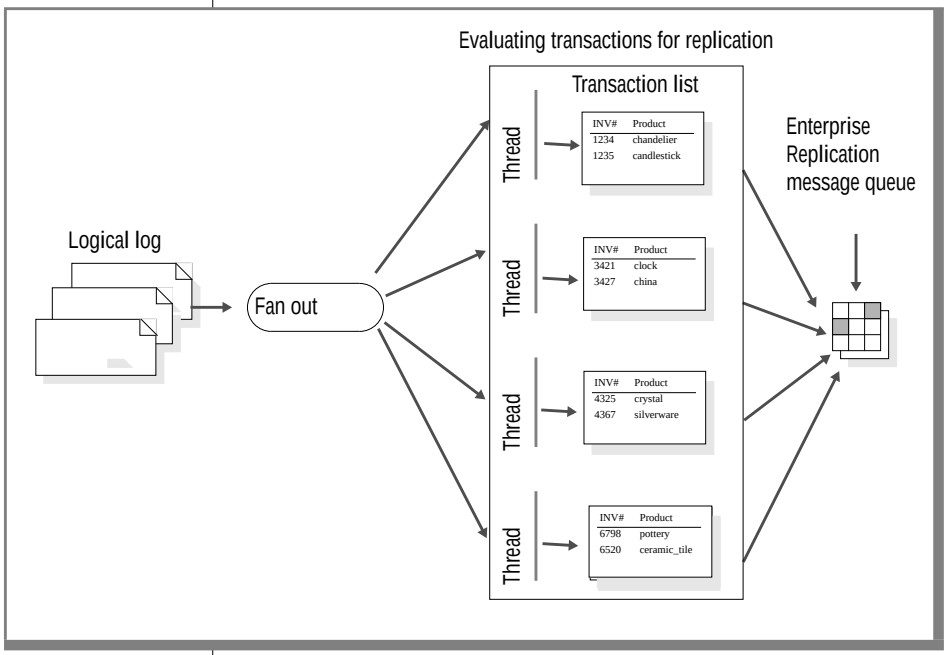


Figure 2-5
Processing in Parallel for High Performance

After Enterprise Replication identifies transactions that qualify for replication, Enterprise Replication transfers the transaction data to a message queue.

Distributing Data

Enterprise Replication uses the message queue to ensure that all data reaches the appropriate server, regardless of a network or system failure. In the event of a failure, the message queue stores data until the network or system is operational. The message queue replicates data efficiently with a minimum of data copying and sending.

Applying Replicated Data

Enterprise Replication uses a data-synchronization process to apply the replicated data to target database servers. The target database servers acknowledge receipt of a message when the message is *safely stored*. A message is safely stored when it is in a stable (recoverable) queue or when it is in the target database. The data synchronization process ensures that transactions are applied at the target in the same order as they were committed on the source database server.

Chapter

3

Understanding Enterprise Replication Objects, Topologies, and Processes

This chapter introduces you to new terms, definitions, and processes to help you use Enterprise Replication. Three sections present the following information:

- Replication objects

  Objects that define the data to replicate.
- Replication topologies

  Network topologies that define the routing of replication data.
- Replication processes

  The procedures to replicate data throughout an Enterprise Replication system.

This chapter references Elite Exporters, Inc., the hypothetical customer introduced in Chapter 2.

Enterprise Replication Objects

Enterprise Replication consists of many objects that define the data in a replication system. The Enterprise Replication objects are:

- Enterprise Replication Server
- Replicate
- Participant
- Replicate group
- Global catalog

Enterprise Replication Server

An Enterprise Replication server is a database server that participates in a replication. Multiple database servers can be on the same physical computer and each database server can participate in Enterprise Replication.

Replicate

A *replicate* defines the data (database, table, and columns) to be replicated and lists the database servers to which the data replicates. A replicate contains the following information

- Replicate name
- Replication type
- Replication options
  - Aborted-transaction spooling (ATS)
  - Row-information spooling (RIS)
  - Canonical message format
  - Database triggers for replicated tables
  - Replication frequency
- Participants
  - Database server
  - Database in which the table resides
  - Table name
  - Table owner
  - SELECT statement

- Conflict-resolution rules
  - Ignore
  - Time stamp
  - Stored procedure
  - Time stamp plus stored procedure
- Conflict-resolution scope
  - Row-by-row
  - Transaction

Each of the components of a replicate is discussed in detail in later sections.

Participant

A *participant* combines a database server, database, table name, owner, and SELECT statement to specify data that should be replicated. A participant is an entity within a replicate. A replicate can contain multiple participants.

The SELECT statement in a participant has the following characteristics:

- The columns selected must include a primary key.
- The statement can reference only a single table.
- The statement cannot include a join or a subquery.
- The statement can include an optional WHERE clause.

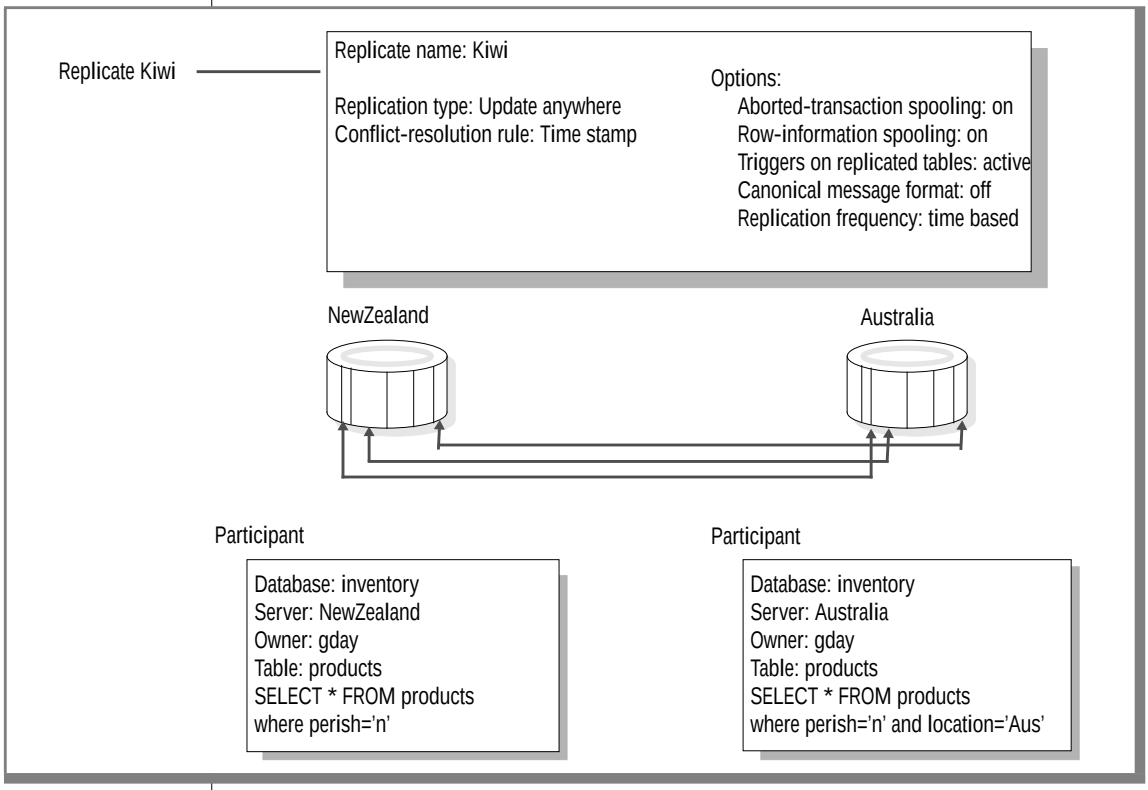
Figure 3-1 on page 3-7 shows the replicate **Kiwi**. A replicate defines the name and type of the replicate, conflict-resolution rule, and options. Each participant within the replicate identifies the data to replicate from one database server to another.

In Figure 3-1 on page 3-7, the **inventory** database on both database servers (**NewZealand** and **Australia**) includes the table **products**, which might be defined as follows:

```
CREATE TABLE products (
    product     CHAR(20),
    prod_id     INTEGER,
.
.
.
    perish      CHAR(1), -- y perishable, n not-perishable
    unit_cost   DECIMAL,
.
.
.
    )
```

Items that are perishable are listed only in the local database. Inserts or updates on perishable items should not be replicated to other database servers.

Figure 3-1
Replicate Kiwi



Replicate Group

A replicate group combines several replicates that have identical participant database servers. Replicate groups provide two major benefits:

- Ease of use through easier administration
- Increased performance with parallel processing

Ease of Use

If your replication system contains many replicates (for example, a hundred replicates) that are controlled identically, when you define a replicate group you can use a single command to start, stop, suspend, or resume all of the replicates.

Increased Performance

You can significantly increase your replication performance when you use replicate groups. Replicate groups can use parallelism to apply replicated transactions at target sites.

In a replicate group:

- Each replicate must replicate to the same set of replication database servers.
- Each replicate in the replicate group must be in the same Enterprise Replication *state*. For more information on states, see "Changing Replicate States" on page 7-26.
- Parallel replicate groups must have disjoint data domains.

Warning: *When you define a replicate group, the default replication process is sequential. You must determine that the data domains between groups are disjoint and that no time dependencies exist before you select parallel (non-sequential) processing. If time dependencies exist or domains are not disjoint and you select non-sequential processing, you can affect data consistency and integrity.*

The following issues are additional factors to consider before you define replicate groups:

- Replicate groups have separate group-processing queues that process in parallel with all other replicate-group queues. If a transaction involves replicates that are not part of a replicate group, the replicates are processed in a single sequential queue.
- Replicate-group names are global within the replication system. Each replicate-group name must be unique. Do not define a replicate group and a replicate with the same name.

Global Catalog

The Enterprise Replication global catalog is a replicated set of tables that maintains a global inventory of replication system options, states, and statistical data. The global catalog contains:

- Replicate definitions
- Enterprise Replication servers
- Enterprise Replication object states
- Participant definitions
- Replicate-group definitions

A global catalog resides on each database server that participates in a replicate. Enterprise Replication manages the meta-data in the global catalog so that the administrator only needs to monitor one global catalog.

A database server that is a leaf can be defined to have a *sparse* global catalog that contains only a portion of the system-wide meta-data.

Enterprise Replication Topologies

Enterprise Replication provides three replication topologies and two connectivity options. The three replication topologies are:

- Fully interconnected
- Hierarchical tree
- Forest of trees

The order in which you declare database servers to Enterprise Replication defines the replication topology. This order is dependent on your physical network topology.

The two connectivity options are:

- Continuous
- Intermittent

You select the database connectivity option when you declare the database server to Enterprise Replication.

For more information on how to declare database servers to Enterprise Replication, see "Opening Replication Manager" on page 7-4.

Before you review the replication topology details, review the replication topology terms in Figure 3-2 and the corresponding illustration in Figure 3-3 on page 3-11.

Figure 3-2
Replication Topology Terms

Term	Definition
Root	An Enterprise Replication database server that is the uppermost level in a hierarchically organized set of information. The root is the point from which database servers branch into a logical sequence. All root database servers within Enterprise Replication must be fully interconnected.
Non root	An Enterprise Replication database server that is not a root database server.
Tree	A data structure that contains database server(s) that are linked in a hierarchical manner. The topmost node is called the root. The root can have zero or more *child* database servers; the root is the *parent* database server to its children.
Parent-Child	A relationship between database servers in a tree data structure in which the parent is one step closer to the root than the child.
Leaf	A database server that has no descendants.

Figure 3-3
Replication Topology Illustration

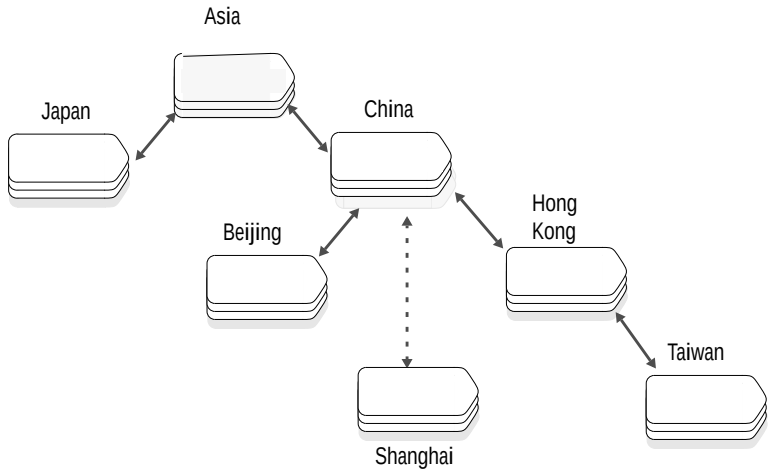


Figure 3-3 illustrates a tree. The root is **Asia**, the uppermost level in a hierarchically organized set of information. **Japan**, **Beijing**, **Shanghai**, and **Taiwan** are all leaves. The parent-child relationship within the tree is as follows:

- **Asia** is the parent of **China** and **Japan**.
- **China** is the child of **Asia** *and* the parent of **Beijing**, **Shanghai**, and **Hong Kong**.
- **Hong Kong** is the child of **China** *and* the parent of **Taiwan**.

All database servers except **Asia** are non-root database servers.

Replication Topology Types

The topology that you choose influences the types of replication that you can use. The next sections describe the topologies that Enterprise Replication supports.

Fully Interconnected

Fully-interconnected replication topology indicates that all database servers connect to each other and that Enterprise Replication establishes and manages the connections. Replication messages are sent directly from one database server to another (no additional routing is necessary to deliver replication messages).

Hierarchical Tree

A *hierarchical tree* consists of a root database server and one or more database servers organized into a tree topology. The tree contains only one root, which has no parent. Each database server within the tree references its parent. A database server that is not a parent is a leaf.

Forest of Trees

A forest of trees is more than one hierarchical tree that connects together with the root database servers. Each hierarchical tree starts with a root database server. The root database servers connect to form the routing to transfer replication messages throughout the network. All root database servers must be able to directly connect to all other root database servers.

Connectivity Options

The two connectivity options you can assign to the database server are continuous and intermittent.

Continuous Connectivity

Continuous connectivity is the default for all database servers. All root database servers use continuous connectivity in the fully-interconnected replication topology. The root database servers of your replication system provide the backbone for your connectivity routing. Enterprise Replication manages the connections between root database servers. If a database server cannot establish contact with another database server, Enterprise Replication tries to establish connectivity until it is successful or a time-out limit expires.

Intermittent Connectivity

Intermittent connectivity indicates that Enterprise Replication does not manage database server connectivity. Connectivity initiates at either end of a connection, at either the child or the parent database server. You must explicitly connect or disconnect to another database server to initiate replication. If a replication system experiences a network or server outage, the connection is lost, and you must reissue the connect or disconnect command when the network resumes. Only leaf nodes can be designated as intermittent.

Enterprise Replication Processes

Enterprise Replication uses various processes to replicate data. This section discusses the processes that replicate data once you define your replication system:

- Capturing transactions
- Evaluating data for replication
- Distributing replication messages

Capturing Transactions

Enterprise Replication uses a log-based transaction-capture mechanism to capture transactions. A log-based transaction capture minimizes the effect that replication has on transaction performance; it captures changes from the log instead of competing with transactions that access production tables.

Enterprise Replication reads the logical log to obtain the row images for tables that participate in a replicate. (Enterprise Replication marks these rows when they are written to the log.) Enterprise Replication reads the logical log in a circular fashion where the most recent logical-log entries write over the oldest entries. If the logical log comes close to overwriting a log that Enterprise Replication has not yet processed (this can happen if the system is severely misconfigured), user transactions are blocked until Enterprise Replication can advance. Reading the log must advance far enough to prevent new log entries from overwriting the logs Enterprise Replication has not yet processed.

The row images that participate in replicates are passed to Enterprise Replication for further evaluation.

Evaluating Data for Replication

Enterprise Replication evaluates transactions based on a distinct time and a change to a final image of a row.

Evaluating Time

Most Enterprise Replication commands are issued at a distinct time. T_{now} refers to a distinct time. The command does not completely take effect until the capture and propagation has passed the T_{now} in the logical log at which the command was originally issued.

In Figure 3-4, a time line illustrates how Enterprise Replication determines the correct data to capture and replicate based on a distinct time.

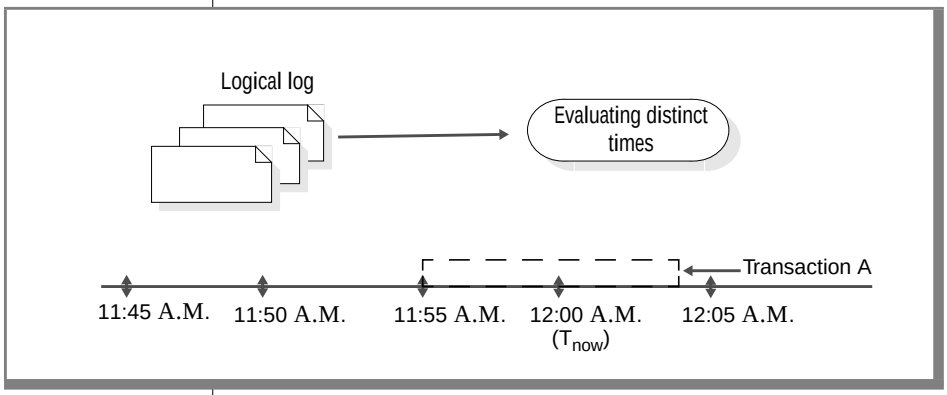


Figure 3-4
Evaluating a Distinct Time

In Figure 3-4, Enterprise Replication begins to read the logical log at 11:45 A.M. (Enterprise Replication only looks at records marked for replication.) At 12:00 P.M., the **Stop Replicate** command is issued. This command stops the replication of data in a replicate. Enterprise Replication refers to the time at which replication stops as T_{now}. All committed transactions from 11:45 A.M. to 12:00 P.M. are passed to Enterprise Replication for evaluation.

Any transactions that are committed after 12:00 P.M. will not be replicated. For example, Transaction A in Figure 3-4 is not passed to Enterprise Replication. Although the transaction starts at 11:55 A.M., it does not complete until 12:04 P.M., and therefore, does not commit by T_{now} (12:00 P.M.)

Evaluating Images of a Row

Enterprise Replication evaluates the initial and final images of a row and any changes that occur between the two row images to determine which rows to replicate. Each row image contains the data in the row as well as the action performed on that row.

Enterprise Replication evaluates the images of a row in parallel to assure high performance. The records that qualify are collected into a transaction list and then passed to the Enterprise Replication *message queue* for distribution. Enterprise Replication uses two queues to send and receive messages: a send queue and a receive queue. The message queue refers to the send and receive queues.

A row might change more than once in a particular transaction. For example, a transaction might insert and update a row prior to committing. Enterprise Replication evaluates the net effect (final state) of a transaction based on the row buffers in the log. Enterprise Replication then determines what should replicate based on the net effect, the initial state of the row, and whether the replicate definition (in particular, the WHERE clause) applies to the initial and final state.

The table in Figure 3-5 shows the logic that determines which rows are candidates for replication.

Figure 3-5
Enterprise Replication Evaluation Logic

Initial Image	Replicate Evaluates	Final Image	Replicate Evaluates	Primary-Key Changed?	Send to Destination Database Server	Comments
INSERT	T *or* F	DELETE	T *or* F	yes *or* no	nothing	Net change of transaction results in no replication
INSERT	T *or* F	UPDATE	T	yes *or* no	INSERT with final row image	Inserts final data of transaction
INSERT	T *or* F	UPDATE	F	yes *or* no	nothing	Final evaluation determines no replication
UPDATE	T	DELETE	T *or* F	yes *or* no	DELETE with initial row image	Net result of transaction is delete
UPDATE	F	DELETE	T *or* F	yes *or* no	nothing	Net change of transaction results in no replication
UPDATE	T	UPDATE	T	yes	DELETE with initial row image and INSERT with final row image	Ensures old primary key does not replicate
UPDATE	T	UPDATE	T	no	UPDATE with final row image	Simple update

(1 of 2)

Initial Image	Replicate Evaluates	Final Image	Replicate Evaluates	Primary-Key Changed?	Send to Destination Database Server	Comments
UPDATE	T	UPDATE	F	yes *or* no	DELETE with initial row image	Row no longer matches replicate definition
UPDATE	F	UPDATE	T	yes *or* no	INSERT with final row image	Row now matches replicate definition
UPDATE	F	UPDATE	F	yes *or* no	nothing	There is no match and therefore, no replication

(2 of 2)

The table in Figure 3-5 on page 3-16 illustrates how Enterprise Replication evaluates the row image type (INSERT, UPDATE, DELETE), the results of evaluating the replicate WHERE clause for both the initial and final image, and whether the primary key changes as a result of the transaction

Tip: *The evaluation logic in Figure 3-5 assumes that the source and the destination tables are synchronized. If the tables are not synchronized, anomalous behavior could result.*

Figure 3-6 through Figure 3-11 show three examples of how Enterprise Replication uses logic to evaluate transactions for potential replication.

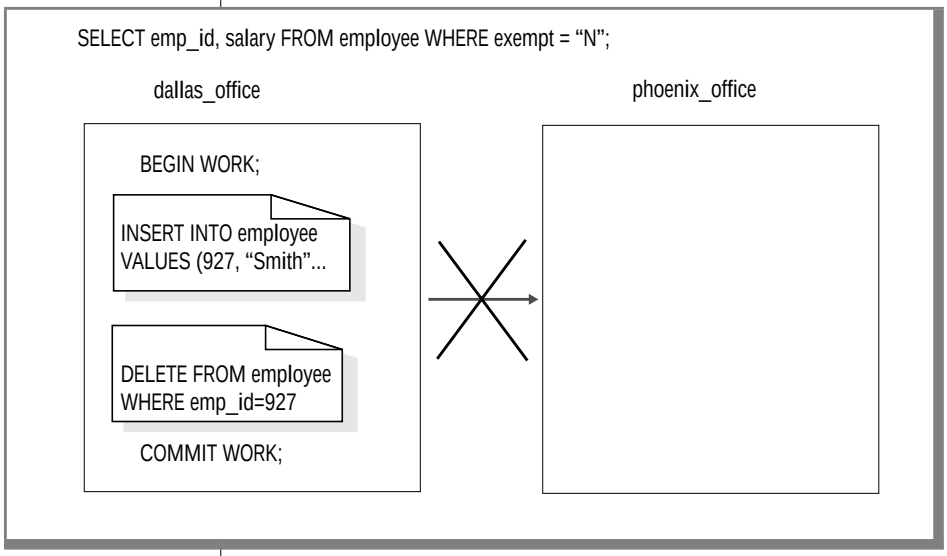


Figure 3-6
Insert Followed by a Delete

In Figure 3-6, Enterprise Replication uses the logic shown in Figure 3-7 to evaluate whether any information is sent to the destination database server.

Figure 3-7
Insert Followed by a Delete Evaluation Logic

Initial Image	Replicate Evaluates	Final Image	Replicate Evaluates	Primary-Key Changed?	Send to Destination Database Server
INSERT	T *or* F	DELETE	T *or* F	yes *or* no	nothing

Enterprise Replication determines that the insert followed by a delete results in no replication operation, therefore, a replication message is not sent.

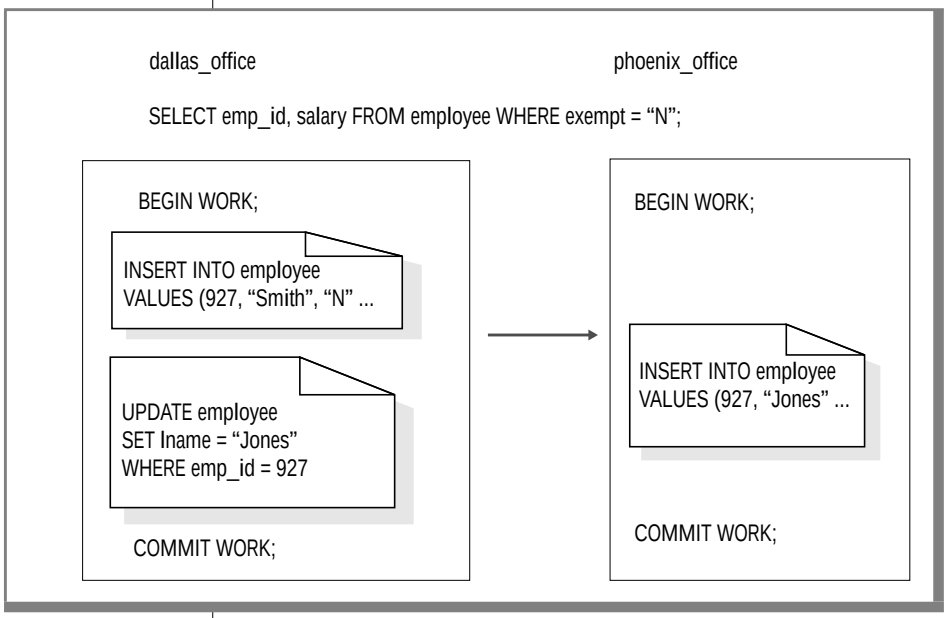


Figure 3-8
Insert Followed by an Update

In Figure 3-8, Enterprise Replication uses the logic shown in Figure 3-9 to evaluate whether any information is sent to the destination database server.

Figure 3-9
Insert Followed by An Update Evaluation Logic

Initial Image	Replicate Evaluates	Final Image	Replicate Evaluates	Primary-Key Changed?	Send to Destination Database Server
INSERT	T *or* F	UPDATE	T	yes or no	INSERT with final row image

The replicate WHERE clause imposes the restriction that only rows with the **exempt** column equal to "N" are replicated. Enterprise Replication evaluates the transaction (an insert followed by an update) and converts it to an insert to propagate the updated image (the final image).

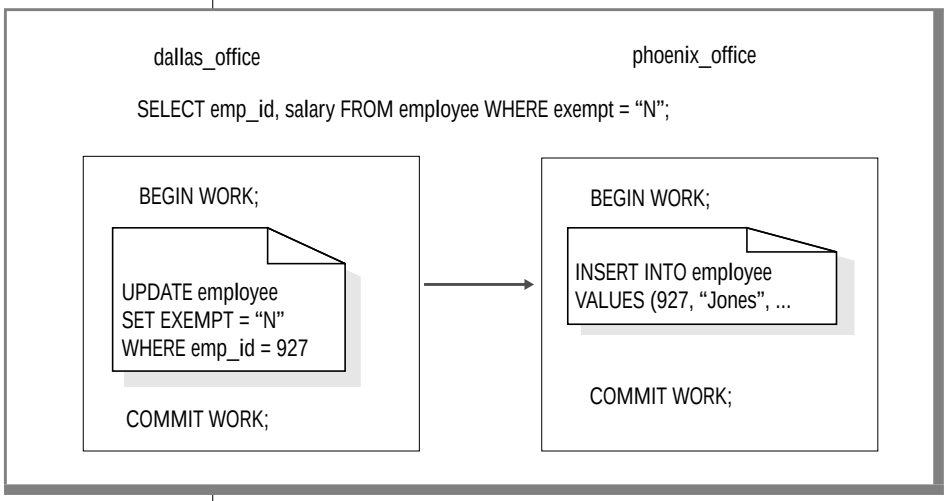


Figure 3-10
Update; Not Selected to Selected

In Figure 3-10, Enterprise Replication uses the logic shown in Figure 3-11 to evaluate whether any information is sent to the destination database server.

Figure 3-11
Update; Not Selected to Selected Evaluation Logic

Initial Image	Replicate Evaluates	Final Image	Replicate Evaluates	Primary-Key Changed?	Send to Destination Database Server
UPDATE	F	UPDATE	T	yes *or* no	INSERT with final row image

The example shows a replicate WHERE clause column update. A row that does not meet the WHERE clause selection criteria is updated to meet the criteria. Enterprise Replication replicates the updated row image and converts the update to an insert.

Enterprise Replication evaluates rows for primary-key updates, for WHERE clause columns updates, and for multiple updates to the same row. The following list describes an occurrence in a transaction and the Enterprise Replication action.

- Primary-key updates

  Enterprise Replication translates an update of a primary key into a delete of the original row and an insert of the row image with the new primary key. If triggers are enabled on the target system, insert triggers are executed.

- Multiple-row images in a transaction

  Enterprise Replication compresses multiple-row images and only sends the net change to the target. Triggers execute only once.

- WHERE clause column updates

  If a replicate includes a WHERE clause in its data selection, the WHERE clause imposes selection criteria for rows in the replicated table.

  - If an update changes a row so that it no longer passes the selection criteria on the source, it is deleted from the target table. Enterprise Replication translates the update into a delete and sends it to the target.
  - If an update changes a row so that it passes the selection criteria on the source, it is inserted into the target table. Enterprise Replication translates the update into an insert and sends it to the target.

For a detailed example that illustrates the evaluation of data in a single transaction where one row is inserted and then updated several times, see "Evaluating Images in a Complex Transaction" on page B-1.

Enterprise Replication supports the replication of BYTE and TEXT data. Enterprise Replication implements the replication of BYTE and TEXT data somewhat differently from the replication of other data types. Read through these sections to understand the processes used to evaluate data for replication, and then read "Replicating BYTE and TEXT Data" on page 3-34.

Receiving and Sending Data from the Message Queue

Enterprise Replication uses receive and send queues (also referred to as the message queues) to receive and deliver replication messages to and from database servers that participate in a replicate.

- Receive queue

  A dbspace that holds replication messages at the target database server until the target database server acknowledges receipt of the messages. You can modify the receive queue dbspace value. For more information, see "Setting Up Your Replication Environment" on page 5-1.

- Send queue

  A dbspace that stores replication messages to be delivered to target database servers that participate in a replicate.

Each message contains the filtered log records for a single transaction. Target sites acknowledge receipt of a message when the message is *safely stored*. A message is safely stored when it is in a stable (recoverable) queue or when it is in the target database.

If a target database server is unreachable, the replication messages remain in a stable queue at the source database server. Temporary failures are common and no immediate action is taken by the source database server; it continues to queue transactions. When the target database server becomes available again, queued transactions are transmitted and applied (see "Applying Data for Distribution" on page 3-23).

If the target database server is unavailable for an extended period, the send queue on the source database server might consume excessive resources. It might not be appropriate to save all transactions for the remote database server in this situation. To prevent unlimited transaction accumulation, the unavailable target database server can be removed from the replicate. The database server that is removed needs to synchronize with the other database servers before it rejoins any replicate. For more information, see Chapter 6, "Preparing Data for Replication."

Applying Data for Distribution

Enterprise Replication applies the replicated data to target database servers.

When Enterprise Replication applies replication messages, it checks to make sure that no *collisions* exist. A collision occurs when potential simultaneous updates of the same data on different database servers are applied. Enterprise Replication reviews data one row at a time to detect a collision. Enterprise Replication scans to see if the same primary key already exists in the target table or in the associated *shadow table*, or if a *replication order error* is detected. A shadow table stores the row images of deleted rows. A replication order error is the result of a replication message that arrives from different database servers with one of the following illogical results:

- A replicated DELETE that finds no row to DELETE on the target
- An UPDATE that finds no row to UPDATE on the target
- An INSERT that finds a row that already exists on the target

If Enterprise Replication finds a collision, it must resolve the conflict before it applies the replication message to the target database server.

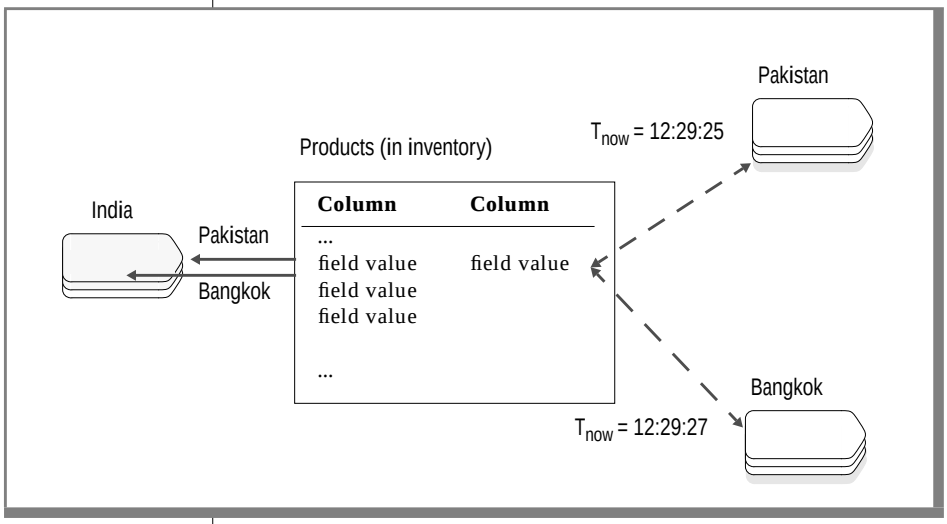


Figure 3-12
Collision Example

Figure 3-12 on page 3-23 shows a situation that yields a conflict. Pakistan updates the row two seconds before Bangkok updates the same row. The Bangkok update arrives at the India site first, and the Pakistan update follows. The Pakistan T_{now} time is earlier than the Bangkok time. Because both updates involve the same data and a time discrepancy exists, Enterprise Replication detects a a collision.

Choosing Conflict-Resolution Rules and Scopes

You need to determine conflict-resolution rules and a conflict-resolution scope for each replicate.

Tip: *The Enterprise Replication Manager supports two operating modes, expert and non-expert (or standard). Expert mode provides more options for control of your Enterprise Replication system than standard mode. The command-line utility (see Chapter 11) and the APIs (see Chapter 12) do not distinguish between these two modes. The CLU and APIs provide all expert-mode options.*

Enterprise Replication supports up to two conflict-resolution rules for each replicate. You can define a primary rule and secondary rule (if desired). Figure 3-13 shows the valid combinations of Enterprise Replication conflict-resolution rules.

Primary Rule	Secondary Rule	Enterprise Replication Operating-Mode Support
Time stamp	None	Expert
Time stamp	Stored procedure	Expert
Stored procedure	None	Expert
Ignore	None	Standard and expert

Figure 3-13 *Valid Conflict-Resolution Rule Combinations*

Each conflict-resolution rule behaves differently depending on the conflict-resolution *scope* you chose. The following list describes each conflict-resolution scope:

- Row-by-row scope

  When you choose a row-by-row conflict-resolution scope, Enterprise Replication evaluates one row at a time. It only applies replicated rows that win the conflict resolution with the target row. If an entire replicated transaction receives row-by-row evaluation, some replicated rows are applied and other replicated rows might not be applied.

- Transaction scope

  When you choose a transaction conflict-resolution scope, Enterprise Replication applies the entire transaction if the replicated transaction wins the conflict resolution. If the target wins the conflict (or other database errors are present), the entire replicated transaction is not applied and rollback occurs.

  A transaction scope for conflict resolution guarantees transactional integrity.

Important: *Enterprise Replication defers some constraint checking on the target tables until the transaction commits. If a unique constraint or foreign-key constraint violation is found on any row of the transaction at commit time, the entire transaction is rejected (regardless of the conflict-resolution scope).*

Time-Stamp Conflict-Resolution Rule

The time-stamp rule evaluates the latest time stamp of the replication against the target and determines a winner. The time-stamp resolution rule behaves differently depending on which scope you choose for conflict resolution (row-by-row or transaction):

- Row-by-row conflict-resolution scope

  Enterprise Replication evaluates one row at a time. The row with the most recent time stamp wins the collision and is applied to the target database servers. If a stored procedure is defined as a secondary conflict-resolution rule, the stored procedure resolves the conflict when the row times are equal.

- Transaction conflict-resolution scope

  Enterprise Replication evaluates the most recent row-update time among all the rows in the replicated transaction. This time is compared to the time stamp of the appropriate target row. If the time stamp of the replicated row is more recent than the target, the entire replicated transaction is applied. If a stored procedure is defined as a secondary conflict-resolution rule, it is used to resolve the conflict when the time stamps are equal.

Tip: *A stored procedure is invoked as a secondary procedure* ***only*** *if Enterprise Replication evaluates rows and discovers equal time stamps.*

If either conflict-resolution scope (row-by-row or transaction) evaluates rows and discovers equal time stamps, and no stored procedure is defined as a secondary conflict-resolution rule, the replicated row or transaction is applied.

Figure 3-14 on page 3-27 shows how a conflict is resolved based on the latest time stamp with row-by-row scope. The time stamp T_{last_update} represents the row on the target database server with the last (most recent) update. The time stamp T_{repl} represents the time stamp on the incoming row.

Enterprise Replication first checks to see if a row with the same primary key exists in either the target table or its corresponding shadow table.

If the row exists, Enterprise Replication uses the latest time stamp to resolve the conflict.

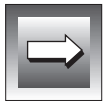
Figure 3-14
Conflict Resolution Based on the Time Stamp

Row exists on target?	Time Stamp	Database Operation		
		Insert	**Update**	**Delete**
No	n/a	Apply row	Apply row (Convert UPDATE to INSERT)	Apply row (INSERT into Enterprise Replication shadow table)
Yes	$T_{last\_update} < T_{repl}$	Apply row (Convert INSERT to UPDATE)	Apply row	Apply row
	$T_{last\_update} > T_{repl}$	Discard row		
	$T_{last\_update} = T_{repl}$ *(how is equality defined?)*	Apply row if no stored procedure is defined as a secondary conflict-resolution rule. Otherwise, invoke the stored procedure.		

In Figure 3-14 the time stamp T_{last_update} represents the row on the target database server with the last (most recent) update. The time stamp T_{repl} represents the time stamp on the incoming row.

Important: *Do not delete the shadow tables created by Enterprise Replication. The shadow table is automatically removed when the last replicate defined with conflict resolution is deleted.*

The time-stamp conflict-resolution rule assumes time synchronization between cooperating Enterprise Replication database servers. All time stamps and internal computations are performed in GMT and have an accuracy of plus or minus one second.



Important: *Enterprise Replication does not manage clock synchronization between database servers that participate in a replicate.*

To synchronize the time on one database server with the time on another database server, use the one of the following commands:

UNIX

```
rdate hostname
```

♦

WiN NT

```
net time \\servername /set
```

or

```
net time /domain:servername /set
```

♦

These commands do not guarantee the times will remain synchronized. Informix recommends that you use a product that supplies a network time protocol to ensure that times remain synchronized.

Stored-Procedure Conflict-Resolution Rule

This rule allows you complete flexibility to determine which row prevails in the database. You can assign stored procedures as the primary conflict-resolution rule.

If you want to use stored procedures as a secondary conflict-resolution rule, the time-stamp conflict-resolution rule must be the primary rule.

Enterprise Replication passes the following information to a stored procedure as arguments.

Argument	Description
Server name [CHAR(18)]	From the local target row (null if local target row does not exist)
Time stamp (DATETIME YEAR TO SECOND)	From the local target row (null if local target row does not exist)
Local delete-table indicator [CHAR(1) or Local key delete-row indicator [CHAR(1)]	■ Y indicates the origin of the row is the shadow table ■ K indicates the origin of the row is the replicate-key delete row (for a key update that is sent as a key delete and a key insert) because the local target row with the old key does not exist.
Server name [CHAR(18)]	Of the replicate source
Time stamp (DATETIME YEAR TO SECOND)	From the replicated row
Replicate action type [CHAR(1)]	I- insert D- delete U- delete update
Local row data returned in regular SQL format	Where the regular SQL format is taken from the SELECT clause of the participant list.
Replicate row data after-image returned in regular SQL format	Where the regular SQL format is taken from the SELECT clause of the participant list.

The stored procedure must set the following arguments before the stored procedure can be applied to the replication message.

Argument	Description
An indicator of the desired database operation to be performed [CHAR(1)]	Same as the replicate action codes with the following additional codes ■ A - accept the replicated row and resolve replication exception ■ S - accept the replicated row as received ■ O - discard the replicated row ■ X - abort the transaction
A non-zero integer value to request logging of the resolution and the integer value in the spooling file(s) (INTEGER)	Logging value takes effect only if logging is configured for this replicate.
The columns of the row to be applied to the target table replicate action type in regular SQL format	This list of column values is not parsed if the replicate action type returned by the stored procedure is S, O, or X.

You can use the arguments to develop application-specific stored procedures. For example, you can create a stored procedure in which a database server always wins a conflict regardless of the time stamp. For an example of a stored procedure, see "A Stored Procedure Example" on page B-3. The following list includes some items to consider when you use a stored procedure.

- Determining when the stored procedure executes:
  - Optimized mode invokes the stored procedure if a collision is detected and the replicate source database server is either different from the server shadow column in the target row, or if the source update time is less than or equal to the target row.

    If you do not choose to optimize the execution of your stored procedures, a stored procedure executes *every* time Enterprise Replication detects a collision.
- Any action that a stored procedure takes as a *result* of replication does not replicate.
- Frequent use of stored procedures might affect performance.

Tip: *Do not assign a stored procedure that is not optimized as a primary conflict-resolution rule for applications that predominantly insert rows successfully.*

Ignore Conflict-Resolution Rule

The ignore conflict-resolution rule does not attempt to detect or resolve conflicts. A row or transaction applies successfully or it fails. A row might fail to replicate because of standard database reasons or replication order errors.

The ignore conflict-resolution rule can only be used as a primary conflict-resolution rule and can have either a transaction or a row-by-row conflict-resolution scope. Figure 3-15 illustrates the ignore conflict-resolution rule.

Figure 3-15
Ignore Conflict-Resolution Rule

	DB Operation		
Row Exists in Target?	**Insert**	**Update**	**Delete**
No	Apply row	Discard row	Discard row
Yes	Discard row	Apply row	Apply row

When a replication message fails to apply on a target, you can spool the information to one or both of the following directories:

- Aborted transaction spooling (ATS)

  If selected, all buffers in a failed replication message that compose a transaction are written to this directory.

- Row-information spooling (RIS)

  If selected, the replication message for a row that could not be applied to a target is written to this directory.

Using Cascading Deletes and Triggers on Replicated Tables

Enterprise Replication supports cascading deletes and triggers that are defined on replicated tables. The following information describes how Enterprise Replication processes cascading deletes and triggers on target database servers.

Using Cascading Deletes

If the source table and the target table define cascading deletes, the children rows that delete as a result of the cascading delete on a parent row on the source database server are also sent to the target database server. When this transaction is applied on the target database server, the cascading delete is invoked when the parent row is processed and the children rows are deleted. Subsequently, when deletes are processed on the children, an error might result since the rows were already deleted. The table in Figure 3-16 indicates how Enterprise Replication resolves cascading deletes with conflict-resolution scopes and rules.

Figure 3-16
Resolving Cascade Deletes

Conflict-Resolution Rule	Conflict-Resolution Scope	Actions on Delete Errors	Result
Time stamp	Row-by-row or transaction	Note as replication exceptions	Process children rows
Ignore	Transaction	Report as errors	Abort entire transaction
Ignore	Row-by-row	Report as errors	Reject row

Using Triggers

If the trigger option is enabled, triggers that are defined on a table that is defined for replication are invoked when transactions are processed on the target database server. Similar to the cascading deletes, if the same triggers are defined on both the source and target tables, any insert, update, or delete operations generated by the triggers are also sent to the target database server. These operations can result in errors depending on the conflict-resolution rule and scope.

In addition, the following trigger operations behave differently when they execute on the target database server:

- All replicated columns are considered to be modified if the database operation is update, including the values for some of the columns that might not have been modified. All update triggers defined on replicate columns are invoked.
- If an update trigger is defined on table *tab1* and the following command is executed

  ```
  UPDATE tab1 SET a = a + 1
  ```

  on the source database server, the command affects all rows in table *tab1*. If any *before* action exists for the update trigger, the *before action* trigger is executed once for this statement, any *for each row* is executed once for each row processed, and any *after* action is executed once for the statement.

  However, if the same update trigger is defined on the target table and the same command is executed on the source database server, the following results occur: all *before*, *for each row*, and *after* actions are executed once for each row that the target database server is processing.

Replicating BYTE and TEXT Data

The following information describes how Enterprise Replication captures, evaluates, and distributes BYTE and TEXT data for replication. Enterprise Replication does not support optical blob data. For general information on the BYTE and TEXT data formats, refer to the *Administrator's Guide*.

Capturing and Evaluating BYTE and TEXT Data

BYTE and TEXT data that is stored in tablespaces (rather than in a blobspace) is placed in the logical log. Enterprise Replication reads the logical log to capture and evaluate BYTE and TEXT data for potential replication.

BYTE and TEXT data that is stored in blobspaces is *not* placed in the logical log. Enterprise Replication retrieves BYTE and TEXT data from blobspaces before sending the data to the target database server. Under the following conditions, Enterprise Replication does not replicate BYTE and TEXT data:

- If the replication operation is an update and the BYTE or TEXT data was not modified, the data is not sent to the target database server for replication.
- If the replication operation is a delete, BYTE or TEXT data is not sent to the target database server for replication.

The following condition only applies to BYTE and TEXT data that is stored in blobspaces.

- If blobspace data is changed or invalidated before it is sent to a database server for replication, the data is flagged as undeliverable when it arrives at the target database server.

  Blobspace data is not entered into the log, and therefore might be modified before it is sent to the target database server. If Enterprise Replication encounters BYTE or TEXT data that was modified, it sends the data that it can successfully retrieve, and flags the non-retrievable BYTE or TEXT data as undeliverable.

Evaluating BYTE and TEXT Data With Time-Stamp Conflict Resolution and Row-by-Row Scope

The following information explains how Enterprise Replication evaluates conflicts in an update-anywhere replicate with a time-stamp conflict-resolution rule and a row-by-row conflict-resolution scope.

When a replicated BYTE or TEXT column that is defined in a dbspace is modified on the source database server, Enterprise Replication attaches **cdrserver** and **cdrtime** to the column. (See Chapter 4, "Designing Enterprise-Replication Systems" for more information on **cdrserver** and **cdrtime**.) If the BYTE or TEXT column on the target database server is also stored in a dbspace, Enterprise Replication evaluates the **cdrserver** and **cdrtime** values in the source and target columns and uses the following logic to determine if the data is to be applied:

1. The column of the replicated data has a time stamp that is greater than the time stamp of the column on the local row.

   If the preceding statement is true, the data for the column is accepted for replication.

2. The server ID and time stamp of the replicated column are equal to the server ID and time stamp on the column on the local row.

   If the preceding statement is true, the data for the column is accepted for replication.

If both conditions are true, the data for the column is accepted for replication even if the rest of the data (that is neither BYTE or TEXT) is rejected.

Evaluating BYTE and TEXT Data With Stored Procedures

If the replicate is defined with a stored-procedure conflict-resolution rule, the stored procedure must return the desired action for each BYTE or TEXT column. When the stored procedure is invoked, each BYTE or TEXT column is passed as an argument for evaluation as five separate fields, as shown in the following table.

Argument	Description
Column size (INTEGER)	Null if the column is null. The size of the column (if data exists for this column).
Blob flag [CHAR(1)]	For the local row, the field is always null. For the replicated row: ■ D - BYTE or TEXT data is sent from the source database server. ■ U - BYTE or TEXT data is unchanged on the source database server.
Column type [CHAR(1)]	P indicates a tablespace data. B indicates a blobspace data.
ID of last update server [CHAR(18)]	The ID of the database server that last updated this column for tablespace data. Null for the blobspace data.
Last update time (DATETIME YEAR TO SECOND)	For the tablespace data: The date and time when the data was last updated. For blobspace data: null.

If the stored procedure returns an action code of A, D, I, or U, the stored procedure parses the return values of the replicated columns. Each BYTE or TEXT column can return a two-character field.

The first character defines the desired option for the BYTE or TEXT column, as shown in the following table.

Value	Function
C	Performs a time-stamp check for this column as used by the time-stamp rule.
N	Sets the replicate column to null.
R	Accepts the replicated data as it is received.
L	Retains the local data.

The second character defines the desired option for blobspace data if the data is found to be undeliverable, as shown in the following table.

Value	Function
N	Sets the replicated column to null.
L	Retains the local data. This is the default.
O	Aborts the row.
X	Aborts the transaction.

Distributing BYTE and TEXT Data

If Enterprise Replication processes a row and discovers undeliverable BYTE or TEXT columns, the following actions can occur:

- Set any undeliverable columns to null if the replication operation is an INSERT and the row does not already exist at the target.
- Retain the old value of the local row if the replication operation is an UPDATE or if the row already exists on the target.

Section II

Preparing for Enterprise Replication

Chapter

4

Designing Enterprise-Replication Systems

This chapter describes Enterprise Replication system types and discusses the trade-offs associated with performance and data availability. Enterprise Replication supports the following three replication types:

- Primary target
- Workflow
- Update anywhere

The three replication systems are presented in order of complexity, where the primary-target replication system is the easiest to define and administer, and update-anywhere replication system is the most complex.

Primary-Target Replication System

In the primary-target replication system, a primary database server can update many target database servers. In this type of replication system, updates to the database(s) are unidirectional, so no conflict detections or rules to resolve conflicts exist. Primary-target replication systems support three business models:

- Data dissemination
- Data consolidation
- Workload partitioning

Data Dissemination

Data dissemination supports a business need where data is updated in a central location and then replicated to read-only sites. This method of distribution can be particularly useful for on-line transaction processing (OLTP) systems where data is required at several sites, but because of the large amounts of data, read / write capabilities at all sites would cripple the performance of the application. Figure 4-1 illustrates data dissemination.

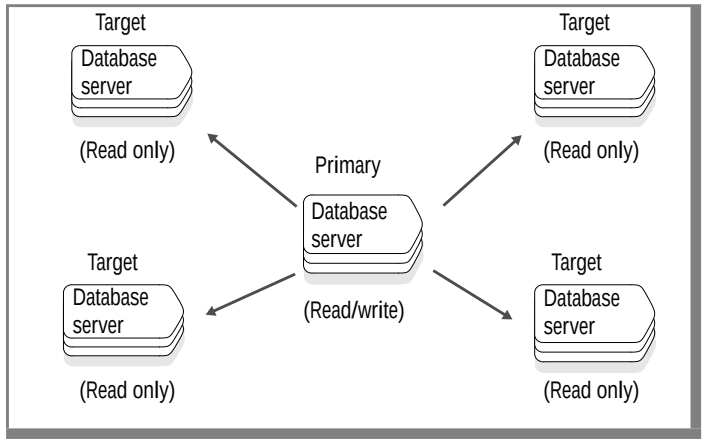


Figure 4-1
Data Dissemination in a Primary-Target Replication System

Data Consolidation

As businesses reorganize to become more competitive, many choose to consolidate data into one central database server. Data consolidation is an excellent vehicle to begin the migration of data from several database servers to one central database server. In Figure 4-2, the remote locations have read / write capabilities while the central database server is read only.

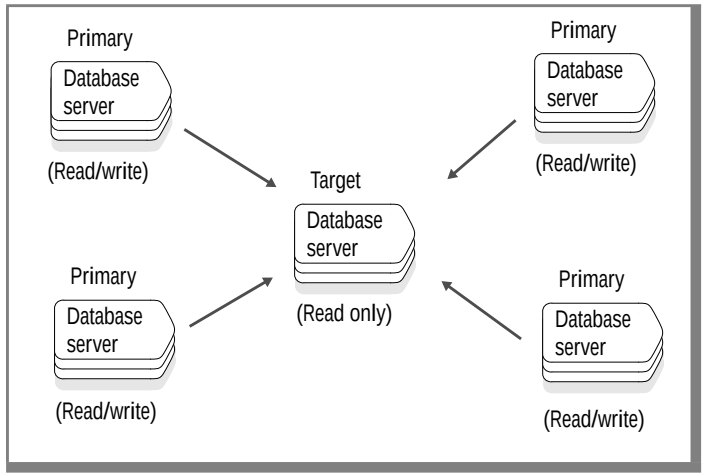


Figure 4-2
Data Consolidation in a Primary-Target Replication System

Businesses can also use data consolidation to off-load OLTP data for decision support (DSS) analysis. For example, data from one or more OLTP systems can be off-loaded to separate local DSS for read-only analysis. Pay close attention to the configuration of the table to which data is replicated to ensure that no collisions occur from the multiple primary database servers. For example, data consolidation can effectively replicate data to a target database server if each primary database server replicates a unique data set.

Tip: *In a replicate defined with multiple primary participants, all of the primaries act as update-anywhere amongst each other.*

Workload Partitioning

Workload partitioning gives businesses the flexibility to assign data ownership at the table-partition level, rather than within an application. Figure 4-3 illustrates workload partitioning.

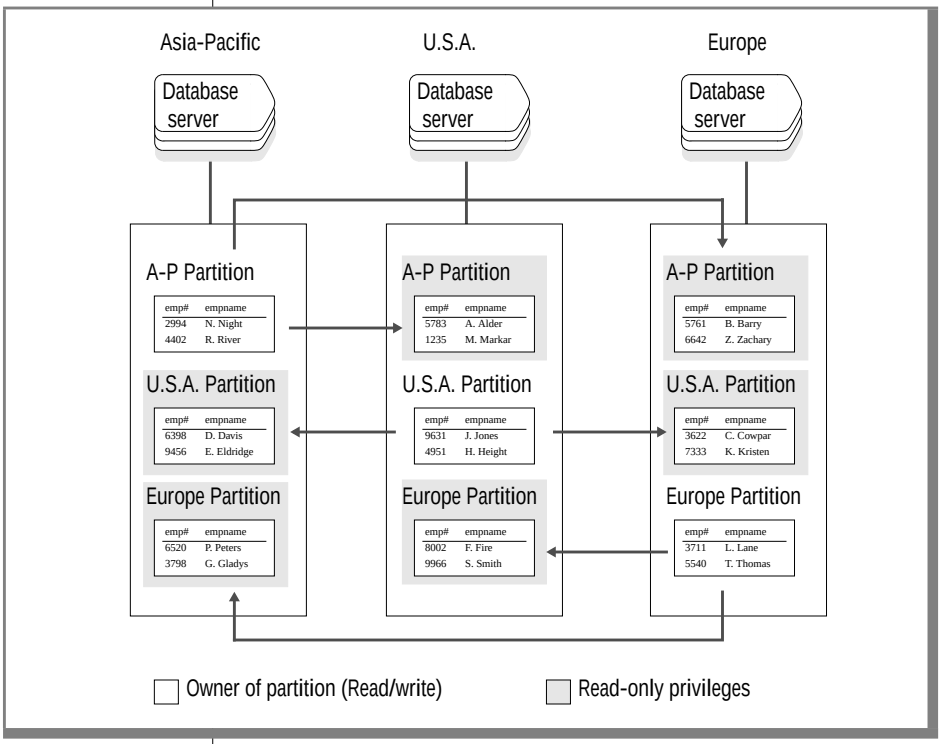


Figure 4-3 *Workload Partitioning in a Primary-Target Replication System*

In Figure 4-3, the replication model matches the partition model for the employee tables. The Asia-Pacific computer owns the partition and can therefore update, insert, and delete employee records for personnel in its region. The changes are then propagated to the U.S. and European regions. Asia-Pacific can query or read the other partitions locally but cannot update those partitions locally. This strategy applies to other regions as well.

Facts to Consider for Primary-Target Replication Systems

Various businesses can use primary-target replication systems to allow users to share data. The following sections describe some of the factors to consider when you select a primary-target replication system for your business application.

Selecting Data for Primary-Target Replication

The requirements for unique primary keys limit the use of serial keys in schema that participate in a replication system. You can define a serial primary key on the primary database server. The replicated primary key column must be defined as an integer on the target.

When Enterprise Replication applies rows that contain a serial-data column at a target database server, the serial-column value is set to the value assigned at the source database server.

Administering the Replication System

Primary-target replication systems are the easiest to administer because all updates are unidirectional and therefore, no conflict detection or resolution rules are required.

Planning for Capacity

All replication systems require you to plan for capacity changes. See Chapter 5, “Setting Up Your Replication Environment” for more information.

Planning for High Availability

In the primary-target replication system, if a target database server goes down or the network goes down, Enterprise Replication continues to log information for the database server until it becomes available. If a database server is unavailable for some time, you may want to remove the database server from the replication system. If the unavailable database server is the read / write database server, you will need to plan a course of action to change read / write capabilities on another database server. If you require a fail-safe replication system, you should design an update-anywhere replication system. For more information, see "Update-Anywhere Replication System" on page 4-9.

Workflow-Replication System

In a workflow replication system, the ability to update asynchronously replicated data moves from site to site and ensures that at any given moment only one site can update the data.

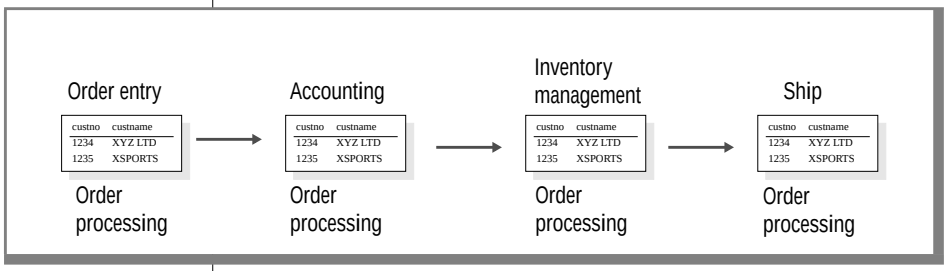


Figure 4-4
A Workflow-Replication System Where Update Authority Moves From Site to Site

Figure 4-4 illustrates an order-processing system. Order processing typically follows a well-ordered series of steps: orders are entered, approved by accounting, inventory is reconciled, and the order is finally shipped.

In a workflow-replication system, application modules can be distributed across multiple sites and databases. Data can also be replicated to sites that need read-only access to the data (for example, if order-entry sites want to monitor the progress of an order).

Facts to Consider

A workflow-replication system, like the primary-target replication system, only allows unidirectional updates. Many of the facts you need to consider for a primary-target replication system should also be considered for the workflow-replication system.

However, unlike the primary-target replication system, availability can become an issue if a database server goes down. The database servers in the workflow-replication system rely on the data updated at a previous site. Consider this fact when you design a workflow-replication system.

Update-Anywhere Replication System

An update-anywhere replication system allows peer-to-peer update capabilities. This capability allows users to function autonomously even when other systems or networks in the replication system are not available.

Figure 4-5
Update-Anywhere Replication system

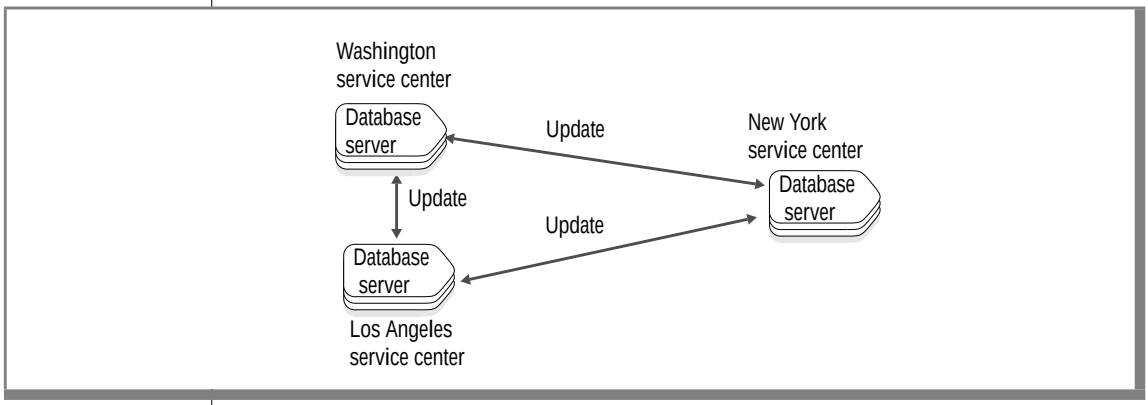


Figure 4-5 illustrates an update-anywhere replication system. Because each service center can update a copy of the data, conflicts can occur when the data is replicated to the other sites. To resolve update conflicts, conflict detection and conflict resolution rules are required.

Facts to Consider

Review the following information before you design your update-anywhere replication system.

Selecting Data for Replication

The requirements for unique primary keys limits the use of serial keys in schema that participate in a replication system. A serial-column key cannot be a primary key by itself. The primary key must include at least one additional column besides the serial column. The additional column must make the combination of the additional column and the serial column globally unique across all database servers that participate in the replication system.

When Enterprise Replication applies rows that contain a serial-data column at a target database server, the serial-column value is set to the value assigned at the source database server.

Administering the Replication System

Update-anywhere replication systems allow peer-to-peer updates, and therefore require conflict-detection or conflict-resolution rules. Administration of this type of replication system requires more administrative steps than the previously mentioned replication systems.

Delivering Consistent Information

Some risk is associated with delivering consistent information in an update-anywhere replication system. You determine the amount of risk based on the type of conflict-resolution rules and procedures you choose with which to resolve conflicts. You can configure an update-anywhere replication system where no data is ever lost, however, you may find that other factors (for example, performance) outweigh your need for a fail-safe mechanism to deliver consistent information.

Planning for Capacity

All replication systems require you to plan for capacity changes. See Chapter 5, "Setting Up Your Replication Environment" for more information.

Planning for High Availability

High availability is a key element in the update-anywhere replication system. Because all database servers have read / write capabilities, you can easily remove or add database servers from the replication system. Design an update-anywhere replication system if your business requires a fail-safe system.

Chapter

5

Setting Up Your Replication Environment

Before you install Enterprise Replication, you need to determine what changes to make to your open-systems environment. This chapter discusses the various operational limitations, capacity requirements, and system changes you need to make based on the Enterprise Replication system you select.

Operational Limitations

Enterprise Replication imposes the following operational limits:

- Enterprise Replication only supports replication between Informix Dynamic Server database servers.
- High-availability data replication (HDR) cannot run concurrently with Enterprise Replication.
- Replication is restricted to base views (logical tables).
- A database server can participate only once in a replicate or group.

SQL Statement Use

Once you have defined a table to Enterprise Replication, you cannot execute any operation that changes the table partition numbers. You cannot use the following SQL statements against the table:

- DROP TABLE
- RENAME TABLE
- ALTER FRAGMENT

The following additional limitations also apply to tables defined for replication:

- Do not remove the primary key constraint.
- Do not rename, add, or drop columns.
- Do not modify a column type.
- Do not add or drop rowids.
- Do not add or drop CRCOLS.
- Do not modify the primary-key column(s).

  For example, do not alter the column to add default values or other constraints.
- Do not create clustered indexes.

The following SQL statements are permitted:

- SET object mode
- START VIOLATIONS TABLE
- STOP VIOLATIONS TABLE
- CREATE TRIGGER
- DROP TRIGGER
- CREATE VIEW
- DROP VIEW
- CREATE SYNONYM
- DROP SYNONYM
- CREATE PROCEDURE
- DROP PROCEDURE
- ADD INDEX
- DROP INDEX
- ALTER INDEX (except TO CLUSTER)
- ALTER TABLE constraints (except for the primary key)

Distributed Transaction

A database server that uses the distributed-transaction facility can simultaneously use Enterprise Replication. A table that is accessed in a distributed transaction can also be a replicated table.

Capacity Planning

The following guidelines are provided to help you plan for the additional capacity Enterprise Replication uses.

Logical-Log Records

The database server uses the logical log to store a record of changes to the data since the last archive. Enterprise Replication requires the logical log to contain entire row images for updated rows.

Important: *Enterprise Replication deactivates the Logical-Log Record reduction option for tables that participate in Enterprise Replication. (The Logical-Log Record reduction option remains enabled for tables that do not participate in Enterprise Replication.) Enterprise Replication logs all columns, instead of the columns that have changed, which consequently increases the size of your logical log.*

When you determine the size of your logical log, examine your data activity for normal operations and for the replication system you defined. For example, if you are replicating a 100-column table, the logical log contains activity information about the original 100 columns *and* all of the columns that have changed.

Send and Receive Message Queues

The term *message queue* refers to both the send queue and the receive queue. The send queue resides at the source database server and contains committed transaction information. The receive queue resides at the target database server and contains information received by the target for application at the table level.

The send and receive queues are dbspaces. To determine the size of your send queue, determine your log usage and then consider how much data you can collect if your network goes down. For example, assume that you usually log 40 megabytes of data each day, but only 10 percent of that is replicated data. If your network is down for 24 hours, and you estimate that you will log 4 megabytes of replicated data each day, the dbspace you identify for the send queue must be at least 4 megabytes.

The receive queues are not always stably stored and usually do not require as much space as the send queue.

The send- and receive-queue dbspaces default to root dbspace. You cannot modify the send-queue dbspace after the database server is declared to Enterprise Replication. You can modify the receive-queue dbspace at any time. However, the new value takes effect only when the database server is brought down and brought back up or when Enterprise Replication is stopped and restarted. To modify the receive queue, see "Viewing Database Server Properties" on page 7-41, or see "cdr_modify_serv()" on page 12-24.

In addition, review the CDR_QUEUEMEM configuration parameter on page C-4 to help determine your capacity needs for the message queue.

Shadow Tables

By default, shadow tables are stored in the same dbspaces as their base tables. Shadow tables are only created if you choose to use time stamp and / or stored procedure as your conflict-resolution rules. They are not created if you choose the ignore conflict-resolution rule.

A good strategy is to determine the size of the base table and then multiply by 2 to estimate the space required in the dbspace. For example, if your base table is 100 megabytes and you defined the replicate with the time-stamp conflict-resolution rule, Enterprise Replication creates a shadow table that can contain up to 200 megabytes of data.

Tip: *Shadow tables are automatically deleted when the last replicate defined with conflict resolution on a table is deleted.*

Spooling Directories

Define the directories for aborted transaction spooling (ATS) and row information spooling (RIS).

Configuration Parameters

The database server configuration file ($ONCONFIG) includes four configuration parameters that affect the behavior of Enterprise Replication:

- CDR_LOGBUFFERS
- CDR_EVALTHREADS
- CDR_DSLOCKWAIT
- CDR_QUEUEMEM

These parameters are documented in Appendix C. For additional information about configuration parameters, refer to the *Administrator's Guide*.

Enterprise-Replication Threads

The following table summarizes all threads used by Enterprise Replication. You can use this information about threads when you evaluate memory usage.

Number of Threads	Thread Description
1	Performs physical I/O from logical log.
1	Verifies potential replication and sends applicable log-record entries to Enterprise Replication fan-out thread.
1	Receives log entries and passes entries to evaluator thread.
n	Evaluates log entry to determine if it should be replicated. (*n* is the number of evaluator threads specified by CDR_EVALTHREADS.) This thread also performs transaction compression on the receipt of COMMIT WORK and sends completed replication messages to the queue process.
1	Parses all WHERE clauses for replicate definitions.
1	Schedules Enterprise Replication events.
1	Takes replication message from the queue process and passes it to the appropriate sending threads.
1	Keeps track of current buffers in use.
1 per connection	Sending thread for site.
1 per connection	Receiving thread for site.
1 per connection	Accepts acknowledgments from site.
1 per connection	Accepts control messages (updates to the global catalog) from other sites and replicates those to the local global catalog.
1 per connection	Determines which replication messages can be processed in parallel and passes appropriate threads.

(1 of 2)

Number of Threads	Thread Description
1 per connection	Receives global catalog information during initialization of server.
1	This thread is created for parallel processing. At least one thread is created. The thread synchronizes data, receives the replication message and applies it to a local instance. The thread also performs conflict resolution and inserts information into shadow tables if required.
1	Schedules internal Enterprise Replication events.

(2 of 2)

Environmental Considerations

You need to evaluate various factors for any open-systems environment. Each replication system that you create can affect your environment uniquely.

Network Bandwidth Limitations

Enterprise Replication captures transactions from the logical log to prevent competition with transaction processing for production applications. However, your network bandwidth limits the amount of data that can be replicated from one site to another. Enterprise Replication applies header information to each row and acknowledgment that is transmitted across the network. To estimate how this affects your current network bandwidth limitations, evaluate approximately how many rows you will replicate and to how many sites. Check that you add extra room for growth. A suggested formula is:

```
(# transactions per hour) * (# bytes) * (# sites) * 130%
```

Transaction-Processing Effect

You must consider many variables when you try to determine the effect that replicating data has on your transaction processing. These variables are discussed in this section.

Using Unbuffered Logging

Informix recommends that you only replicate tables in databases created with unbuffered logging.

Enterprise Replication evaluates the logical log for transactions that modify tables defined for replication. If a table defined for replication resides in a database that uses buffered logging, the transactions are not immediately written to the logical log, but are instead buffered and then written to the logical log in a block of logical records. When this occurs, Enterprise Replication evaluates the buffer of logical-log records all at once, which consumes excess CPU time and memory. When you define a table for replication in a database created with unbuffered logging, Enterprise Replication can evaluate the transactions as they are produced.

Analyzing Replication Volume

Determine how many data rows change per day. For example, an application issues a simple DELETE statement that deletes 100 rows. If this table is replicated, these 100 rows must be propagated and analyzed before they are applied to the targets.

Using Complex SELECT Statements

Evaluate the complexity of the SELECT statement that specifies the data for replication. If you define your replication system with a SELECT *.*, Enterprise Replication is optimized and immediately continues to process. If your SELECT statement includes a WHERE clause, Enterprise Replication must evaluate the WHERE clause, which can affect performance.

Supported Data Types

Replicate definitions can include all Informix data types, including BYTE and TEXT. Enterprise Replication supports all primitive data types across heterogeneous hardware. If you define a replicate that includes non-primitive data types (for example, BYTE and TEXT data), then the application must resolve data-representation issues that are architecture dependent.

The data types of the original table and the replicated table must match. No data conversions are supported.

Using GLS with Enterprise Replication

An Enterprise Replication system can include databases in different locales, with the following restrictions:

- When you define a server to Enterprise Replication, that server must be running in the English locale.

  In other words, the **syscdr** database on every Enterprise Replication server must be in the English locale.
- You can replicate only between databases that are in the same locale.
- Replication names can be in the locale of the database.
- Replication names might be displayed incorrectly by the Replication Manager.

  The Replication Manager can run in any locale, but it can run in only one locale at a time. If the databases in your replication system are in multiple locales, the names of some of the replicates might display incorrectly.

For more information about using GLS, refer to the *Guide to GLS Functionality*.

Replicating Data

Section

Chapter

6

Preparing Data for Replication

The goal of data replication is to provide identical, or at least consistent, data on multiple database servers. This chapter describes how to prepare the information in your databases for replication.

When you define a new replicate on tables with existing data on different database servers, the data might not be consistent. Similarly, if you add a participant to an existing replicate you need to ensure that all of the databases in the replicate have consistent values.

You might also need to take special steps when you recover data. For example, if a disk fills up on a target database server, you can recover the data from the ATS files. However, it might be easier to unload the data from the source and apply that data to the target database.

Preparing Consistent Data

In most cases, preparing consistent data simply requires that you decide which of your databases has the most accurate data and then copy that data onto the target database. If the target database already has data, you must remove that data before adding the copied data.

For copying data, you can use the following tools:

- The High-Performance Loader (HPL)
- The **onunload** and **onload** utilities
- The **dbexport** and **dbimport** utilities
- The UNLOAD and LOAD statements

When you unload and load data, you must use the same type of utility for both the unload and load operations. For example, you cannot unload data with the **onunload** utility and then load the data with a LOAD statement.

Using the High-Performance Loader

The High-Performance Loader (HPL) provides a high-speed tool for moving data between databases. For information about how to use the HPL, refer to the *Guide to the High-Performance Loader*.

Using the HPL when Shadow Columns Are Present

How you use the HPL depends on how you defined the tables to replicate. If the table definition included the WITH CRCOLS clause, then you must take special steps when you unload the data.

When you use the WITH CRCOLS clause with the CREATE TABLE statement, the database server creates two shadow columns, **cdrserver** and **cdrtime**, that contain replication information.

If the table contains CRCOLS, you must perform the following additional steps:

1. You must include the **cdrserver** and **cdrtime** columns in the query statement when you unload the data.
2. You must include the **cdrserver** and **cdrtime** columns in your map when you load the data.
3. You must use Express mode to load data that contains the **cdrserver** and **cdrtime** columns. You must perform a level-0 archive after completion.

Using the onunload and onload Utilities

You can use the **onunload** and **onload** utilities only to unload and load an entire table. If you want to unload selected columns of a table, you must use either **unload** or the HPL.

For more information about **onunload** and **onload**, see the *Informix Migration Guide*.

Using the dbexport and dbimport Utilities

If you need to copy an entire database for replication, you can use the **dbexport** and **dbimport** utilities. These utilities unload an entire database, including its schema, and then recreate the database. If you want to move selected tables or selected columns of a table, you must use some other utility.

For more information about **dbexport** and **dbimport**, see the *Informix Migration Guide*.

Using the UNLOAD and LOAD statements

The UNLOAD and LOAD statements allow you to move data within the context of an SQL program.

If the table contains CRCOLS, you must perform the following two additional steps:

1. You must include the **cdrserver** and **cdrtime** columns in your SELECT statement when you unload your data.
2. You must list the columns that you want to load in the INSERT statement and include the cdrserver and cdrtime columns in the list when you load your data.

Blocking Replication

If the table that you are preparing for replication is in a database that already uses replication, you may need to block replication while you prepare the table. The BEGIN WORK WITHOUT REPLICATION statement starts a transaction that does not replicate to other database servers.

The following code fragment shows how you might use this statement:

```
BEGIN WORK WITHOUT REPLICATION
LOCK TABLE office
DELETE FROM office WHERE description = 'portlandR_D'
COMMIT WORK
```

The following list indicates actions that occur when a transaction starts with BEGIN WORK WITHOUT REPLICATION:

- All constraint checking for the transaction is automatically set to deferred.
- SQL does not generate any values for the **cdrserver** and **cdrtime** columns for the rows that are inserted, updated, or deleted within the transaction. You must supply these values. If you do not supply the values, the columns are null if the operation is an insert. If the operation is an update or delete, the values in the columns are not changed.
- If you use the CRCOLS clause to modify a table that is already defined to Enterprise Replication, you must explicitly list the columns to be modified. The following two examples show an SQL statement and the correct changes to the statement to modify columns:
  - If **tabname1** is a table defined for replication, the statement:

    ```
    LOAD FROM filename INSERT INTO tablename1;
    ```

    must be changed to:

    ```
    LOAD FROM filename INSERT INTO tablename1 (list of
    columns);
    ```

    The list of columns must match the order and the number of fields in the load file.
  - If **tabname1** and **tabname2** are tables defined for replication with the same schema, the statement:

    ```
    INSERT INTO tabname1 SELECT * FROM tabname2;
    ```

    must be changed to:

    ```
    INSERT INTO tabname1
           SELECT cdrserver, cdrtime, * FROM tabname2;
    ```

For more information about these statements, refer to the *Informix Guide to SQL: Syntax*.

Data Preparation Examples

The following examples illustrate how to prepare data for replication.

Adding a New Participant to an Existing Replicate

The following steps provide an example of how to add a new participant to an existing replicate with the LOAD and UNLOAD statements and the BEGIN WORK WITHOUT REPLICATION statement. Replicate **zebra** contains the following database servers: Alpha, Beta, and Gamma. You need to add Delta. Assume that Alpha is the database server from which you want to get the initial copy of the data.

1. Declare server Delta to Enterprise Replication.
2. Select server Alpha as the definitive server.
3. Suspend replication to Delta.
4. Add Delta as a participant to replicate **zebra** for table *tab1*.
5. Start replication for replicate **zebra** on Delta.
6. Unload the data from table *tab1* using the UNLOAD statement.
7. Copy the files that contain the data to Delta.
8. Start the transactions with BEGIN WORK WITHOUT REPLICATION, load the data using the LOAD statement, and commit the transactions.
9. Resume replication to Delta.

It may seem strange to declare replication on server Delta and then immediately suspend replication. You must do this because, while you are preparing the replicates and unloading and loading files, the other servers in the replicate (Alpha, Beta, and Gamma) may be collecting information that needs to be replicated. After you have finished loading the initial data to Delta and resume replication, the information that was generated during the loading process can be replicated.

Beginning Work Without Replication

You might need to put data into a database that you do not want replicated, perhaps for a new server or because you had to drop and recreate the table.

Using DB-Access to Begin Work Without Replication

The following example shows how to use DB-Access to begin work without replication as well as update the Enterprise Replication shadow columns **cdrserver** and **cdrtime**.

```
DATABASE adatabase;
BEGIN WORK WITHOUT REPLICATION
INSERT into mytable (cdrserver, cdrtime, col1, col2, ....)
    VALUES (10, 845484154, value1, value2, ....);
UPDATE mytable
    SET cdrserver = 10, cdrtime = 845484154
    WHERE col1 > col2;
COMMIT WORK
```

Using ESQL/C to Begin Work Without Replication

The following example shows how to use ESQL/C to begin work without replication as well as update the Enterprise Replication shadow columns **cdrserver** and **cdrtime**.

```
MAIN (argc, argv)
      INT   argc;
      CHAR  *argv[];
{
      EXEC SQL CHARstmt[256];
      EXEC SQL database mydatabase;

      sprintf(stmt, "BEGIN WORK WITHOUT REPLICATION");
      EXEC SQL execute immediate :stmt;

      EXEC SQL insert into mytable (cdrserver, cdrtime,
      col1, col2, ...)
      values (10, 845494154, value1, value2, ...);

      EXEC SQL update mytable
      set cdrserver = 10, cdrtime = 845494154
      where col1 > col2;
      EXEC SQL commit work;
}
```

Important: *You must use the following syntax when you issue the BEGIN WORK WITHOUT REPLICATION statement from ESQL/C programs:*

```
sprintf(stmt, "BEGIN WORK WITHOUT REPLICATION");
EXEC SQL execute immediate :stmt;
```

Chapter

7

Administering Enterprise Replication

The Enterprise Replication Manager (RM) is a Windows NT graphical user interface (GUI) provided to help you administer and monitor Enterprise Replication. It is part of the Informix Enterprise Command Center (IECC).

This chapter introduces the Replication Manager and guides you through some basic tasks. The on-line help provides extensive help for more advanced tasks. Chapter 9, "Monitoring Enterprise Replication," provides instructions for how to use the Replication Manager to monitor your replication system.

You can also perform all of the operations that are described in this chapter with the command line utility, as described in Chapter 11, or with API calls, as described in Chapter 12.

Starting the Replication Manager

To open the RM, choose **Tools→Enterprise Replication Manager** from the main window of the IECC.

Preparations

Before you can use the RM, you must add your database servers to the IECC. Use the IECC Add Database Server wizard to add the databases. This wizard is documented in the *Informix Enterprise Command Center User Guide*.

Terminology

Many of the pages in the RM ask you to select a *server*. In this chapter, and in Chapter 8, the term *server* refers to a *database server group*. Each database server that participates in a replication system must be a member of a database-server group. For more information about database-server groups, refer to "Database-Server Group" on page 11-10.

Opening Replication Manager

When you open the Replication Manager, it checks to see whether any database servers have been declared for Enterprise Replication. If at least one server has already been declared for Enterprise Replication, the Replication Manager main window appears, as shown in Figure 7-4 on page 7-7.

If no database servers have been declared (that is, if you are starting replication for the first time), you must declare a database server before you can continue.

Using the Replication Manager for the First Time

If no database servers have been declared for ER, you will see one of two messages, as shown in Figure 7-1 and Figure 7-2 on page 7-5.

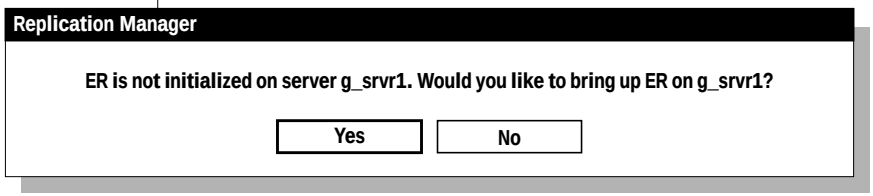


Figure 7-1
Initialize Dialog Box

You must declare a database server before you can continue. If you choose **Yes** in the dialog box in Figure 7-1, the RM continues with the process for declaring a server, as described in "Using the Replication Manager for the First Time".

However, you might want to declare a different server from the one that the RM has suggested. If you click **No** in the dialog box in Figure 7-1 on page 7-4, the display in Figure 7-2 appears.

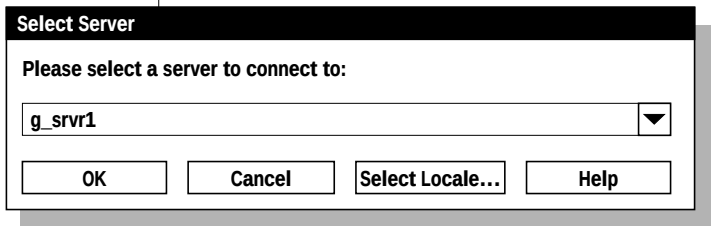


Figure 7-2
Select Server Dialog Box

After you select the server that you want to declare, the RM again asks whether you want to bring up the on ER, as in Figure 7-1. If you click **Yes**, the RM continues with the procedure for declaring the first server.

Declaring the First Enterprise Replication Database Server

The dialog box in Figure 7-3 allows you to declare a database server and its attributes.

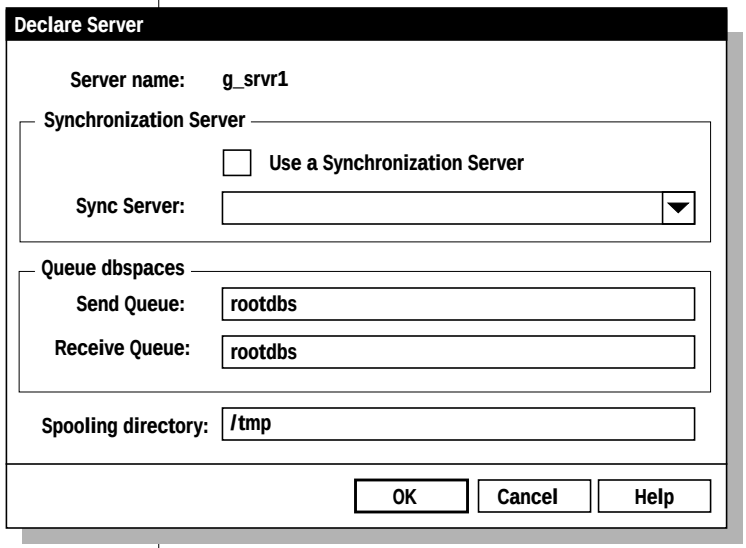


Figure 7-3
Declare Server Dialog

When you declare the first ER server, make the following choices:

- Synchronization Server

  The first server that you declare does not have a synchronization server. Make sure that the **Use a Synchronization Server** check box is *not* checked. The Synchronization Server pane should be gray.

- Queue dbspaces

  If you do not want the send and receive queues to go into the root dbspace, you must exit from the RM (choose **Cancel**), create the dbspaces, and then return to the RM.

- Spooling

  Make sure that the spooling directory exists. If the directory does not exist, the declare-server process fails.

Click **OK** to continue.

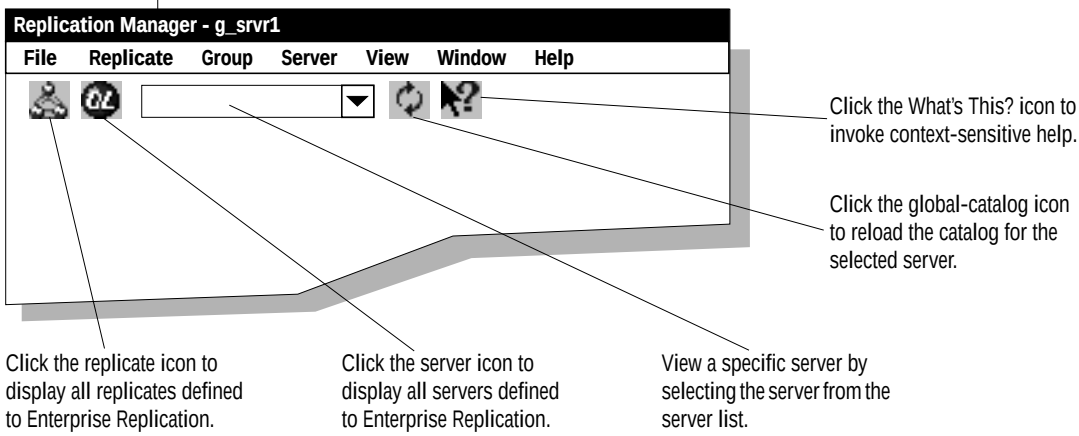
After you have declared the first database server for ER, the Replication Manager main window appears. The next time you enter the RM, the main window will appear immediately.

The Replication Manager Main Window

The main window provides a central point of control for the Replication Manager. You can define, modify, delete, and view database servers, replicates and groups, and monitor replication activity. Figure 7-4 on page 7-7 shows the header and toolbar of the main window.

To access menu items, click a menu from the title bar and then choose the menu item. You can also select a replicate, group, or database server and press the right key of your mouse to access the menu items.

Figure 7-4
Replication Manager Main Window



Standard Mode and Expert Mode

The Replication Manager has two modes:

- Non-expert (or standard)
- Expert

Standard mode assumes that all of your database servers are on Windows NT systems. It provides only *primary-to-secondary* replication. In primary-to-secondary, or *source-to-target* replication, one database server is the primary replication server. All changes to the primary, or source, database server are replicated to the secondary, or target, database servers. Changes to the secondary database servers are *not* replicated to the primary database server.

Expert mode allows you to declare the following types of replication:

- Secondary-to-primary replication
- Update-everything replication
- Hierarchical replication
- Sporadic connections

The following sections describe how to use the Replication Manager in standard mode.

For more information about expert mode refer to "Selecting Expert Mode" on page 7-46.

On-Line Help

The Replication Manager provides on-line help for both reference help and context-sensitive help. Reference help includes topics such as *What is a Replicate?* and *Defining a Primary Replicate*. Context-sensitive help provides a concise answer that describes the function of a menu item or icon that you are asking about.

Accessing Reference Help

To access reference help, choose **Help→Help Topics**. Figure 7-5 shows an example of a Help Topics dialog box.

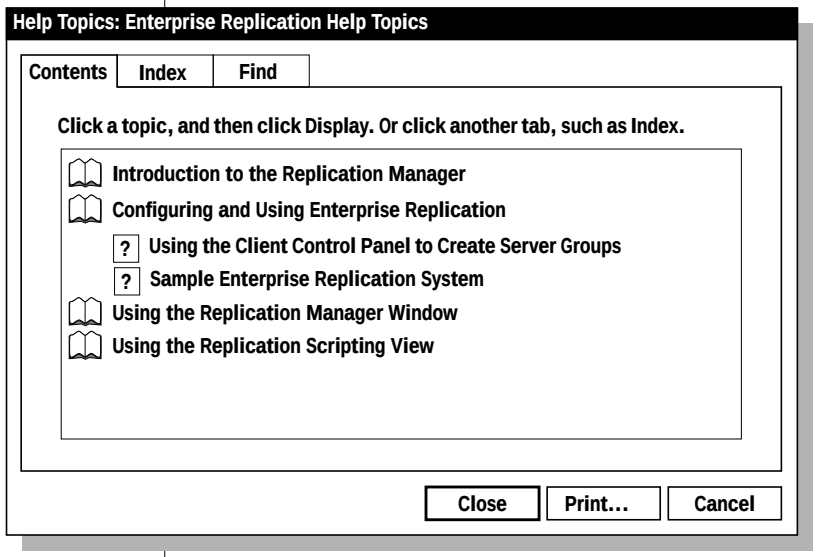


Figure 7-5
Help Topics Dialog Box

Accessing Context-Sensitive Help

At any time, you can choose one of the following options to obtain help.

- Click the **What's This?** icon in the title bar.

  When you click the **What's This?** icon, the arrow changes to an arrow with a question mark. Drag the **What's This** icon to the object for which you want help and click the left mouse button.
- Select the object and press F1.

Additional information is provided for all toolbar objects. When you point to an object on the toolbar, the status bar automatically provides information about that object.

Declaring the Additional Replication Database Servers

To declare an additional database server for Enterprise Replication, choose the server group from the server list on the tool bar. If that server is not already declared, the RM asks if you want to bring up ER on that server. When you click **Yes**, the Declare Server dialog in Figure 7-3 on page 7-5 again appears.

Follow the procedure described in "Using the Replication Manager for the First Time" on page 7-4, *except* for the following step:

- Use the **Synchronization Server** check box
- Select a server from the **Server** list box to serve as the synchronization server. The synchronization server must be a server that has already been declared for ER.

Defining a Replicate

To define a replicate, choose **Replicate→Define Replicate.**

The **Define Replicate...** menu item accesses the Define Replicate wizard (Figure 7-6 on page 7-11). The Define Replicate wizard provides a series of windows that guide you through the process to define a replicate.

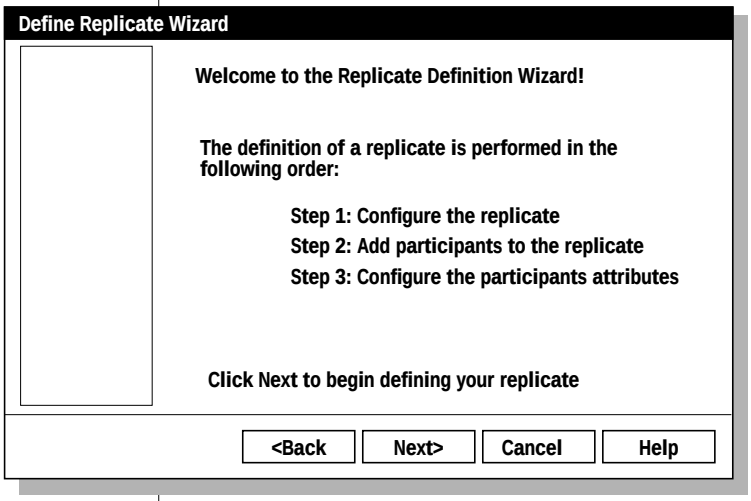


Figure 7-6
Define Replicate Wizard Welcome Dialog Box

The introductory wizard lists the steps required to define a replicate.

All wizards include command buttons at the bottom of a window to allow you to navigate through the wizard. The Replication Manager uses the following command buttons:

Command	Action
<Back	Returns to the previous page
Next>	Moves to the next page in sequence, keeping changes from previous pages
Finish	Completes the task
Cancel	Discards any changes and terminates the process
Help	Requests help

Defining a Replicate Name

The wizard asks for a replicate name, as Figure 7-7 shows:

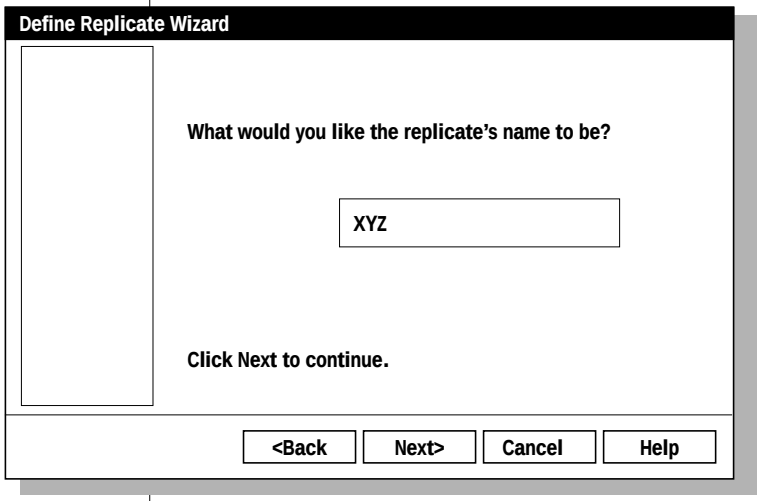


Figure 7-7
Define Replicate Name

Select the Replication Frequency

Next the wizard asks you to specify whether you want replication to be continuous or time-based. With the time-based option, you can choose the time (in hours and minutes) and day (day of week, day of month) that data replication occurs in a replicate. Time-based replication begins when you start the replicate.

In the example in Figure 7-8, time-based replication will occur every Sunday at 3:45 P.M.

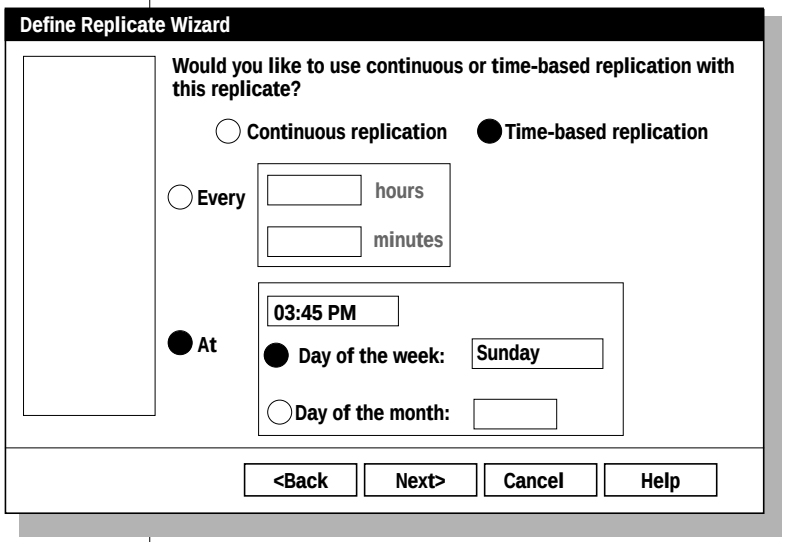


Figure 7-8
Select Continuous or Time-Based Replication

Frequency Options

The following table describes the options you can select to determine when replication occurs.

Value	Description
Continuous	This value is the default and infers that replication is continuous.
Time-based	Replicate data at a specified interval using the **Every** option or at a specific time on a specific day using the **At** option.

(1 of 2)

Value	Description
Day of the week	Replicate data once a week at the time and day specified. You can also replicate data at the same time each day of the week by specifying the value `Daily` in the **Day of the week** text box.
Day of the month	Replicate data once a month at the specified time and day of the month. To ensure that replication occurs on the last day of every month, select the word `Last` in the **Day of month** text box. If you select `31`, for example, replication takes place *only* in months that have 31 days.

(2 of 2)

Important: *Time-based replication is based on the local time, not Greenwich Mean Time (GMT). If database servers participating in the replicate are in different time zones than the database server on which the time is set, replication will occur at different times.*

Select Servers to Participate in the Replicate

In Figure 7-9 on page 7-15 the wizard asks you to select the servers that you want to participate in the replicate. All database servers in an update-anywhere replicate must have read / write capabilities to allow peer-to-peer replication.

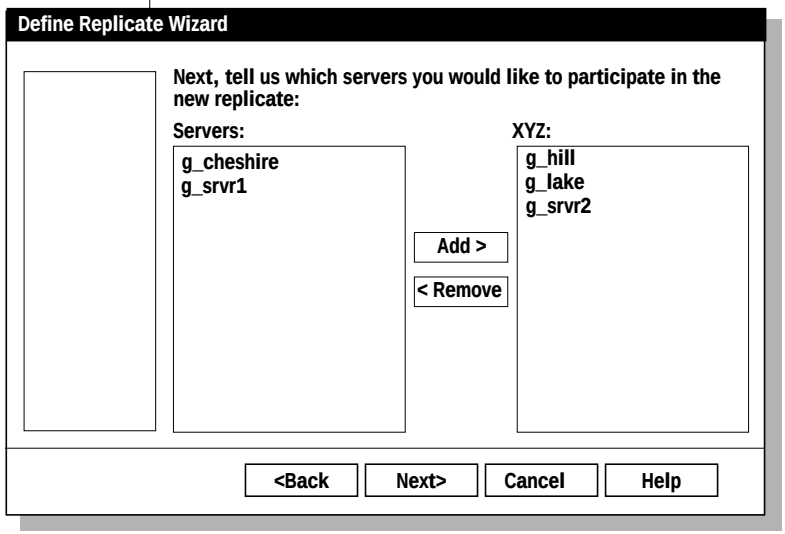


Figure 7-9
Select Database Servers for a Replicate

All database-server groups that are declared to Enterprise Replication are listed in the **Servers** list box. To select the servers for your replicate, click the server name (or several server names) and click **Add**. To remove a server from your replicate, click the server name and click **Remove**. Figure 7-9 shows that **g_hill**, **g_lake**, and **g_srvr2** were added to replicate XYZ.

Specify Identical or Non-Identical Participants

The Define Replicate wizard page in Figure 7-10 on page 7-16 asks two questions:

- Are the participant attributes are identical?
- Which participant should be the source?

Specifying Identical Participants

A participant consists of a database server and the participant attributes: the table name, owner of the table, database in which the tables resides, and the SELECT statement. If all the participant attributes are identical, click **Yes**.

For example, if user **informix** owns (identical) **stores7** databases on all database servers in the replicate, and you want to replicate the same columns to each **customers** table, then the participant attributes are identical.

If, on the other hand, the databases in the replicate have identical columns, but one database is named **mansions** while the other database is named **villas**, then the attributes are not identical and you must click **No**.

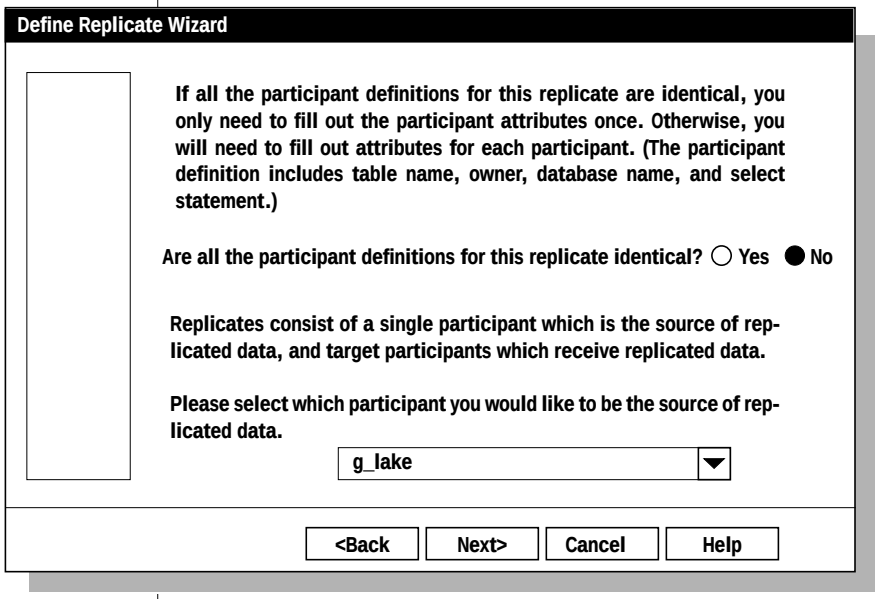


Figure 7-10
Describe Attributes and Select Source

Defining Participant Attributes

The Define Replicate wizard page in Figure 7-12 on page 7-19 asks you to define the table you want to replicate, the owner of the table, and the name of the database in which the table resides.

Important: *Enterprise Replication places some restrictions on certain ALTER and SET statements you can issue to tables that are participants in a replicate. For more information on SQL statement usage, see* *"Setting Up Your Replication Environment" on page 5-1*.

Defining Identical Attributes

If the participants in the replicate are identical, then you only need to define the attributes once, as illustrated in Figure 7-11.

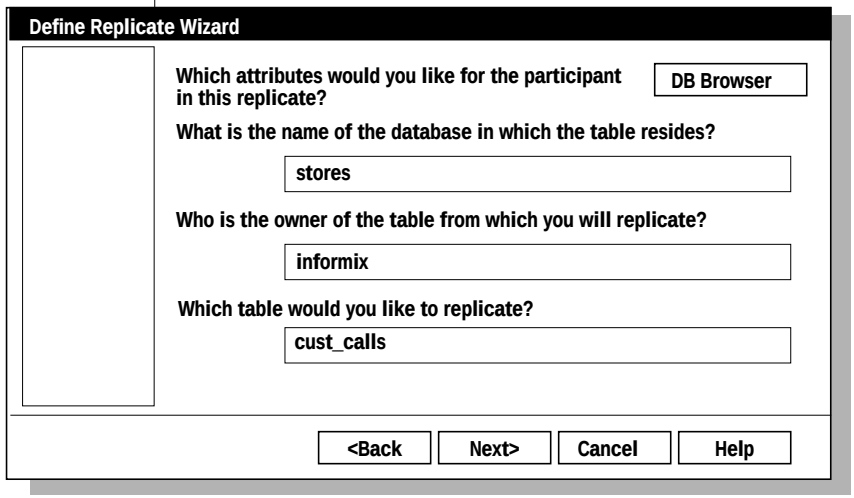


Figure 7-11 *Define Data Attributes for Identical Participants*

When the attributes are identical, the database name, owner, and table that you fill in for Figure 7-12 on page 7-19 are the same for all database servers that participate in the replicate.

Defining Non-Identical Attributes

If you specified in Figure 7-10 on page 7-16 that the attributes of the participants are not identical, the wizard guides you through the attribute definition once for each database server. The display, illustrated in Figure 7-12 on page 7-19 differs only in the question on the first line, which specifies the server name.

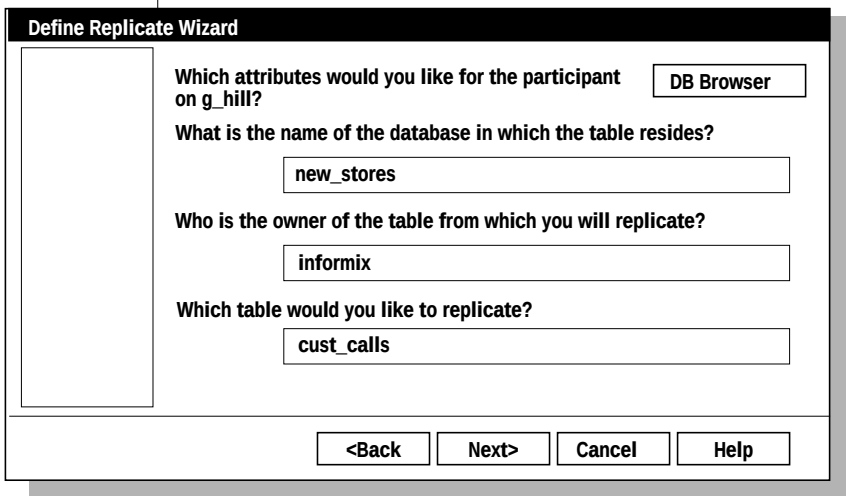


Figure 7-12
Define Data Attributes for Non-Identical Participants

Using the Browser Button

The **DB Browser** button appears in both Figure 7-12 and Figure 7-13 on page 7-20. The **DB Browser** button opens a **Database Query** window that allows you to browse through the databases of the database server of the participant. You can use the **Database Query** window to select the database, table, and columns for the participant.

Choose the Columns to Replicate

The Define Replicate wizard page in Figure 7-13 on page 7-20 asks you to select columns for replication. An asterisk appears in the column selection (the first text box) to indicate that you want to select all columns. You can select individual column names to replicate by typing the name of each column. The column names you select (either specific or all) are automatically added to the SELECT statement in the second box.You can also create a WHERE clause to define the exact data you want to replicate.

If you used the **DB Browser** button on the Define Attributes page, you might have selected columns to replicate. If not, the page in Figure 7-13 allows you to either type in the selected columns or to use the **DB Browser** button to choose the columns for replication.

Important: *You must include a primary key in your column selection. You cannot use a join or aggregated columns in the SELECT statement.*

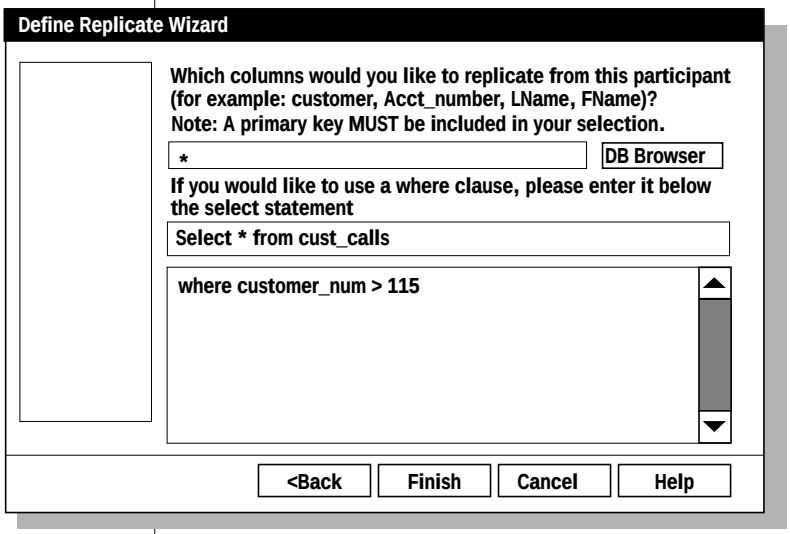


Figure 7-13
Select Data for Replication

Enter a WHERE Clause

After you choose the columns for replication, you can enter a WHERE clause in the lower pane to define the exact data you want to replicate in your replicate. Figure 7-13 includes a sample WHERE clause. Informix recommends that you create a WHERE clause on each participant. If a primary participant experiences hardware errors or you have to remove it from a replicate because of connectivity problems, you already have the target participant defined correctly and can easily change the target to a primary participant.

Column Mapping

You can perform column mapping with your column selection. Each participant in the replicate must define the identical number of columns (n to n mappings) and the data types must be the same. See "Column Mapping" on page B-5 for more information.

The Replicate Summary

After you have finished entering information about the participants in the wizard, the RM shows a summary of the replicate that you just defined.

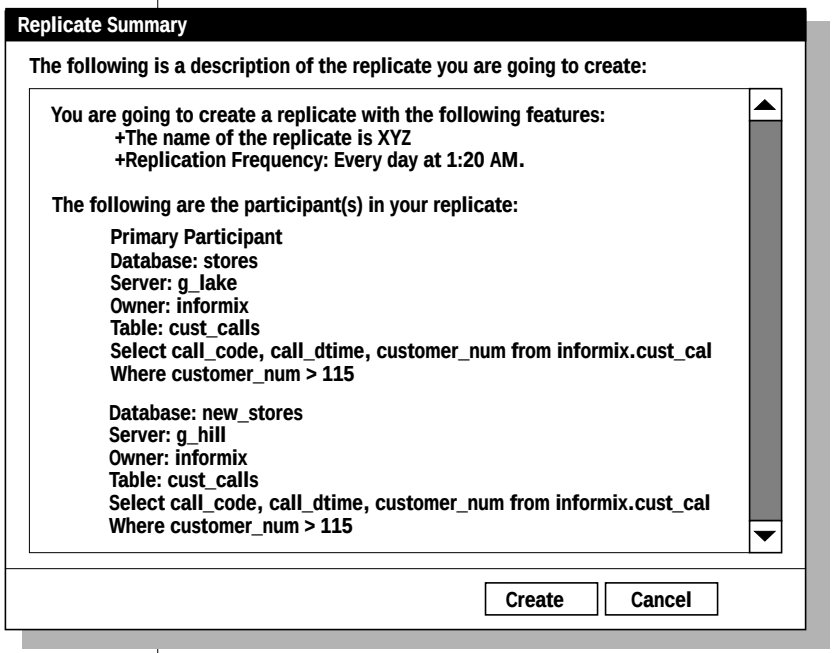


Figure 7-14
Replicate Summary Window

The Replicate Summary in Figure 7-14 provides a summary of the attributes you selected for your replicate. Use the scroll bar to review all information for each database server that participates in your replicate. If you discover errors, click **Cancel**. You are returned to the window where you select data for replication (Figure 7-13 on page 7-20). Click **<Back** to find the appropriate window and make the necessary change(s).

Click **Create** to create the replicate.

After you create the replicate, the RM returns to the main window, as shown in Figure 7-15 on page 7-23.

Important: *Do not create a clustered index on a table after you have defined a replicate on that table; replication on the table will cease.*

Using the Replicate Menu

In the main window, you can click the **Replicate** icon (at the left end of the tool bar) to see replicates and participants, as illustrated in Figure 7-15.

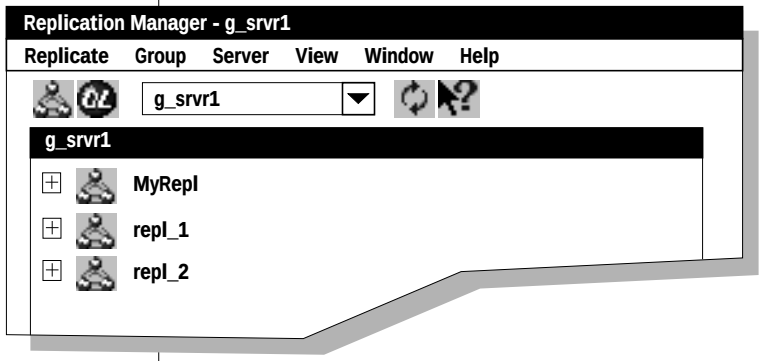


Figure 7-15
Replicate View

The **Replicate** menu in Figure 7-16 allows you to define and control replicates.

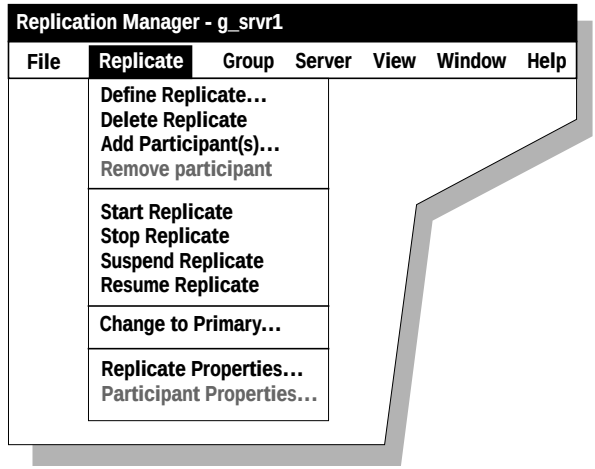


Figure 7-16
Replicate Menu

A menu item that contains an *ellipsis* (…) (for example **Define Replicate…**) indicates that a wizard, dialog box, or a property sheet follows to help you complete a task. All other items on the Replicate menu are equivalent to the command-line utility options (see Chapter 11) and the API calls (see Chapter 12).

Tip: *To guarantee an accurate display of replication objects, select the database server from the database server list. (When you select the database server from the database server list, Replication Manager reloads the global catalog.)*

Managing Replicates and Participants

The first four items on the Replicate menu allow you to define and delete replicates and to add and remove participants.

Defining a Replicate

To define a new replicate, choose **Replicate→ Define Replicate**. "Defining a Replicate" on page 7-10 describes the steps for defining a replicate.

Deleting a Replicate

To delete a replicate, select the replicate on the main window of the Replication Manager, and then choose **Replicate→Delete Replicate**. When ER receives this command, it immediately purges all data from the queues associated with the replicate and removes the replicate from the global catalog.

Important: *If you delete an active replicate, Enterprise Replication cannot guarantee consistent tables unless the replicate is stopped and the queues are properly emptied.*

Adding a Participant

To add participant(s) to an existing replicate, select the replicate from the main window of the Replication Manager, and choose **Replicate→Add Participant(s)...**.

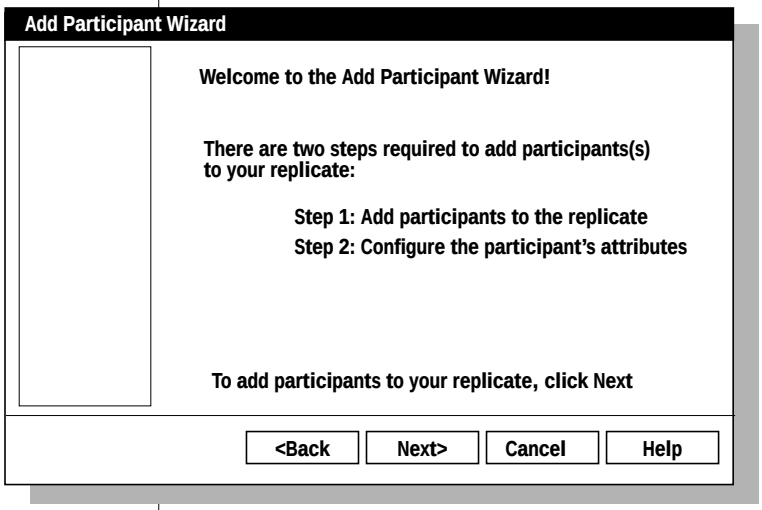


Figure 7-17 *Add Participant Wizard Welcome Page*

The Add Participant wizard page in Figure 7-17 lists the steps necessary to add participant(s) to an existing replicate. The Add Participant wizard continues with the steps to add a participant that parallel the steps to define a replicate in Figures 7-8 through Figure 7-13.

Warning: *If you add a participant to an existing replicate, you must synchronize the databases before you start replication with the new participant. For information on how to synchronize your databases, see Chapter 6, "Preparing Data for Replication."*

Removing a Participant

To remove a participant from a replicate, click the replicate to display the participants. Select a participant and choose **Replicate→Remove Participant**. The RM asks you to verify that the participant should be removed, as in Figure 7-18.

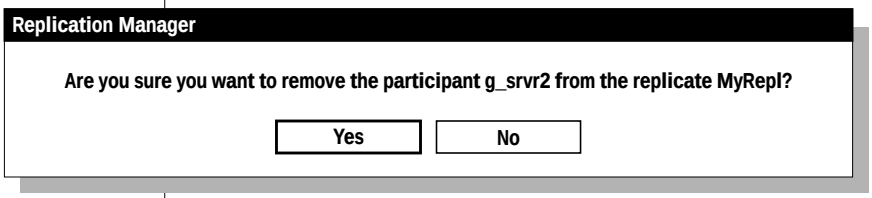


Figure 7-18
Remove Participant Confirmation

Changing Replicate States

The second group of choices on the Replicate menu allow you to change the state of a replicate. A replicate can be in one of the following states:

- Inactive
- Active
- Suspended
- Quiescent

Inactive

When a database is in inactive state, no database changes are captured, transmitted, or processed. When you first define a replicate, it is inactive.

To change the state of a replicate to inactive, select the replicate and then choose **Replicate→ Stop Replicate**.

Active

When a database is in active state, ER captures data from the logical log and transmits the captured data to participants. To change the state of a replicate to active, select the replicate and then choose **Replicate→ Start Replicate**. The Start Replicate dialog box lets you select the participants to start in the replicate, as shown in Figure 7-19.

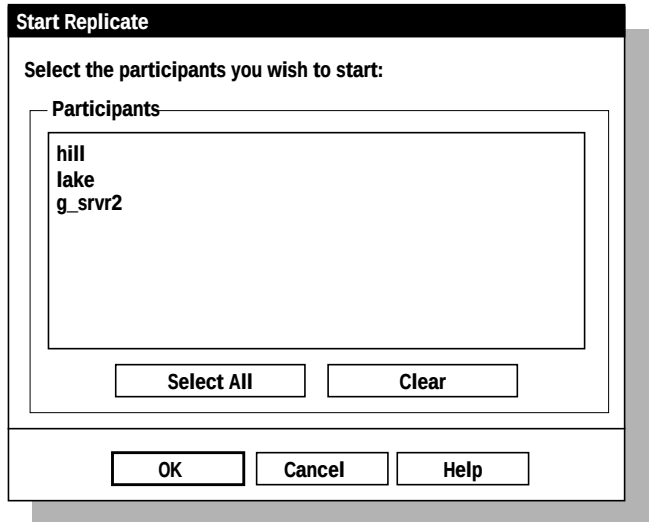


Figure 7-19
Start Replicate Dialog Box

Suspended

In suspended state, the replicate still captures and accumulates database changes, but does not transmit any of the captured data. The data is queued and waits until the state is changed to active to transfer the captured data.

To change the state to suspended, choose **Replicate→ Suspend Replicate**.

To change from suspended back to active, choose **Replicate→ Resume Replicate**. The database server transfers all data accumulated during the suspended state. After all of the data is transferred, the database returns to active state.

Quiescent

If you choose **Replicate→ Stop Replicate,** the replicate enters a quiescent state. No more database changes are captured for the replicate. However, activity continues until all changes on transactions that are already committed are delivered. Only complete transactions are delivered. None of the changes from a transaction that is committed after **Stop Replicate** are delivered.

The quiescent state is a temporary state. You cannot control the quiescent state from the Replication Manager.

Changing the Primary Participant

One of the participants in a replicate is the primary database server. Changes on the primary database server are replicated to the other participants in the replicate. To specify a different participant as the primary database server in a replicate, select the participant that should become the primary database server and choose **Replicate→Change to Primary** from the Replicate menu.

Viewing Properties

The final two choices on the Replicate menu let you view the properties of replicates and their participants.

Replicate Properties

The Replicate Properties window summarizes the properties you have assigned to a replicate.To view replicate properties, select a replicate and then choose **Replicate→ Replicate Properties**. The RM displays the properties of the replicate, as shown in Figure 7-20.

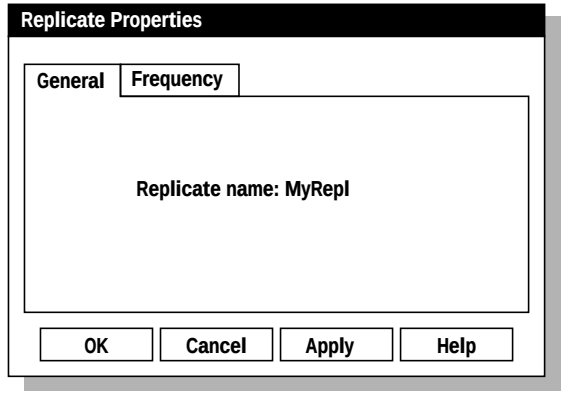


Figure 7-20
Replicate Properties Window

You can also use this window to change the frequency of replication. To change the frequency, click the **Frequency** tab, change the frequency, and then click **Apply**.

Important: *Enterprise Replication calculates a new replication schedule at the time you select* ***Apply****. For example, if you change the time-based frequency value from every hour to every two hours and select* ***Apply*** *at 12:20 P.M., replication occurs at 2:20 P.M.*

Participant Properties

The Participant Properties window shows the columns and the WHERE clause selected for the participant. To view participant properties, select a participant and then choose **Replicate→ Participant Properties**. The RM displays the properties of the participant, as shown in Figure 7-21.

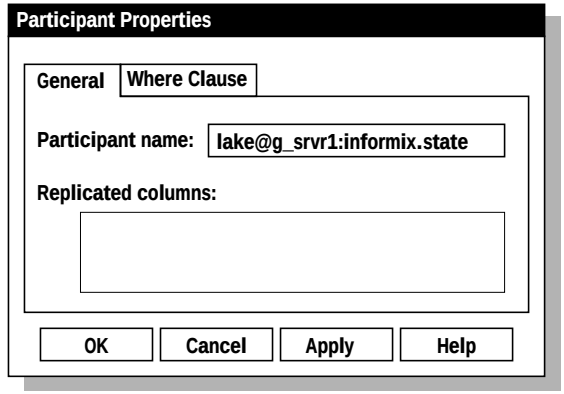


Figure 7-21
Participant Properties Window

Group Menu

Use the **Group** menu in Figure 7-22 to define and control replicate groups and review replicate group properties.

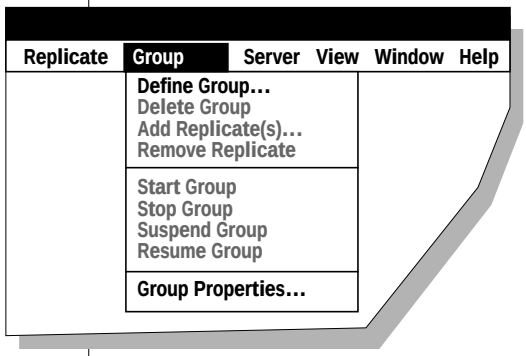


Figure 7-22
Group Menu

Defining a Replicate Group

To define a replicate group, choose **Group→Define Group**. The RM starts the Define Replicate Group wizard, as shown in Figure 7-23. The Define Replicate Group wizard provides a series of pages that guide you through the process to define a replicate group.

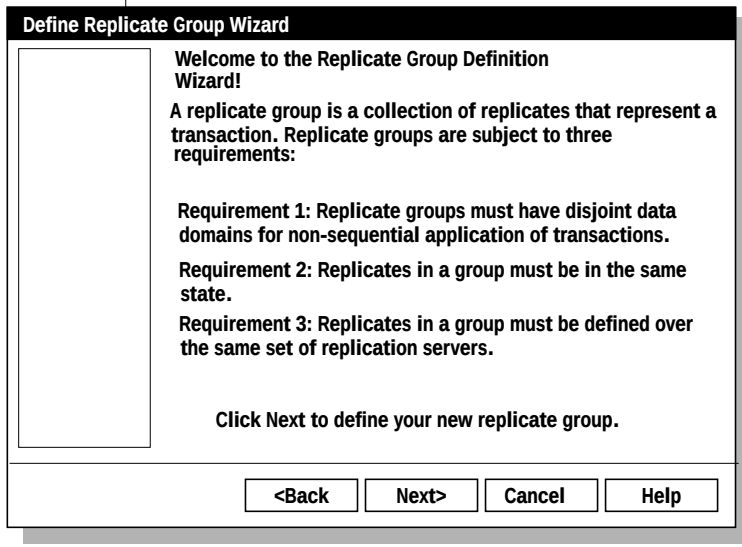


Figure 7-23
Define Replicate Group Wizard Welcome Page

Choose a Group Name

In Figure 7-24, the wizard asks you to choose a name for the replicate group.

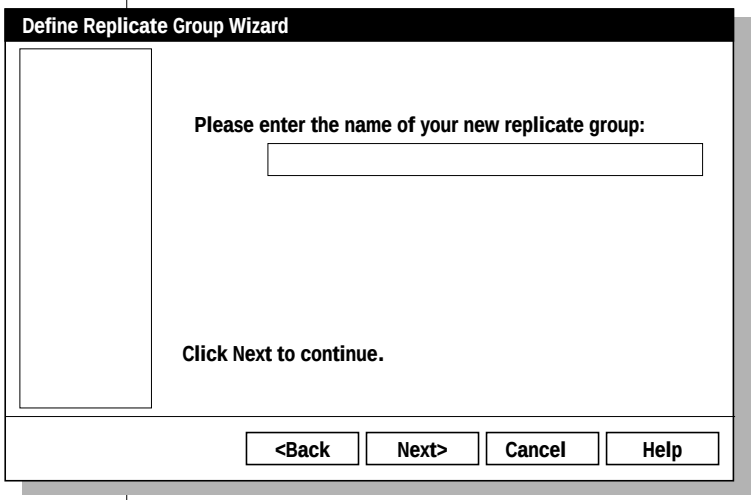


Figure 7-24
Define Replicate Group Name

Choose the Replication Interval for the Group

Next the wizard asks you to choose a replication interval for your replicate group. The display for specifying the replication interval for the group is the same as the display for specifying the replication interval for a replicate, as shown in Figure 7-8 on page 7-13.

Important: *Each replicate in a replicate group* ***must*** *have the same time-based replication frequency value (or all replicates must have continuous replication enabled).*

Select Replicates for the Replicate Group

The Define Replicate Group wizard page in Figure 7-25 asks you to select the replicates you want to participate in the replicate group. All replicates that are defined to Enterprise Replication are listed in the **Replicates** list box. Select the replicates for your replicate group.

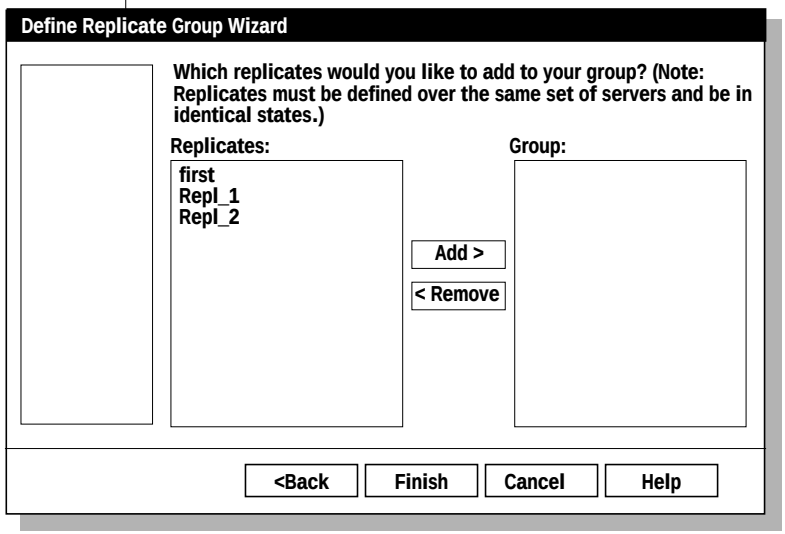


Figure 7-25 *Select Replicates for the Replicate Group*

If you try to select replicates that have different replication frequencies, the RM notifies you that you have selected inconsistent replicates, as in Figure 7-26.

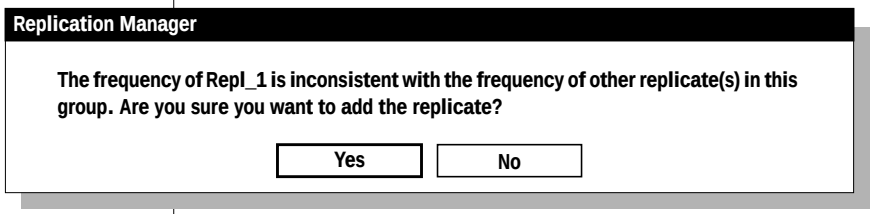


Figure 7-26 *Inconsistent Frequency Warning*

Modifying Groups

After you have defined at least one replication group, you can use the choices on the Group menu to modify the groups.

Deleting a Replicate Group

To delete a replicate group, select the replicate group. Choose **Group→Delete Group**. When ER receives this command it deletes the replicate group from the global catalog. The **Delete Group** command does not affect the replicates or the associated data in the replicate group. If you have time-based replication enabled for the replicate group, the replicates defined to the replicate group retain the time-based frequency that was set for the replicate group.

Adding Replicate(s) to an Existing Replicate Group

To add replicates to an existing replicate group, select the replicate group and then choose **Group→Add Replicates**. The Add Replicates dialog box in Figure 7-27 lets you add new replicates to a group or remove current replicates from a group.

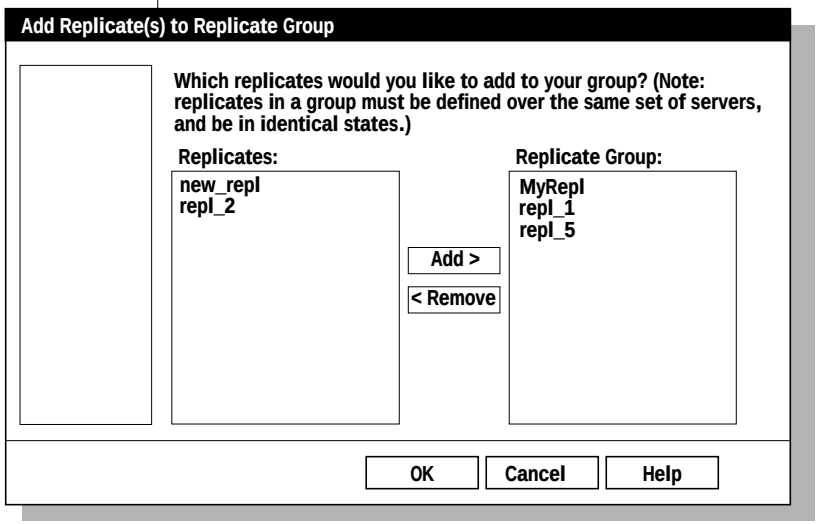


Figure 7-27
Add Replicates to an Existing Replicate Group

The **Replicates** list box shows all replicates that are defined to ER that are not already included in the replicate group. To select the replicates for your replicate group, select the replicate name (or a group of replicates) and click **Add**. To remove a replicate from your replicate group, select the replicate name (or a group of replicates) and click **Remove**.

Removing Replicate(s) from an Existing Replicate Group

To remove a replicate from an existing replicate group, select the replicate and choose **Group→Remove.**

Changing the State of a Group

The second group of choices on the Group menu allow you to change the state of a group. A group can be in one of the following states:

- Inactive
- Active
- Suspended
- Quiescent

Inactive

When you first define a replicate group, the replicate group assumes the state of the replicates in the replicate group. Therefore, if the replicates are inactive, the replicate group is inactive.

Active

A replicate group assumes an *active* state if all replicates in the replicate group are active. To change the state of a replicate group from inactive to active, choose **Group→Start Group**. When you choose **Start Group**, Enterprise Replication begins capturing data from the logical log and transmitting captured data to replicates.

Suspend

When you choose **Suspend Group,** the replicate group enters the *suspend* state. Suspended replicate groups still capture and accumulate database changes but do not transmit any of the captured data. To change the state to suspended, choose **Group→Suspend Group.** The data is queued and waits until the state is changed to active to transfer the captured data. To change the state to active, choose **Group→Resume Group**. All data accumulated during the suspended state is transferred; the database change capture continues and transmission resumes.

Quiescent

If you choose **Stop Group** the replicate group enters a quiescent state. No more data is captured for the replicate group. However, data transmission continues until all changes are delivered. The quiescent state is a temporary state. You cannot control quiescent state from the Replication Manager.

Important: *A transaction that is committed after a stop command will have none of the transaction delivered (only complete transactions are delivered).*

Viewing Replicate Group Properties

The Replicate Group Properties dialog box summarizes the properties you assigned to a replicate group. To view replicate group properties, select the replicate group and then choose **Group→Group Properties**. The RM displays a property sheet that shows the properties of the replicate group, as illustrated in Figure 7-28.

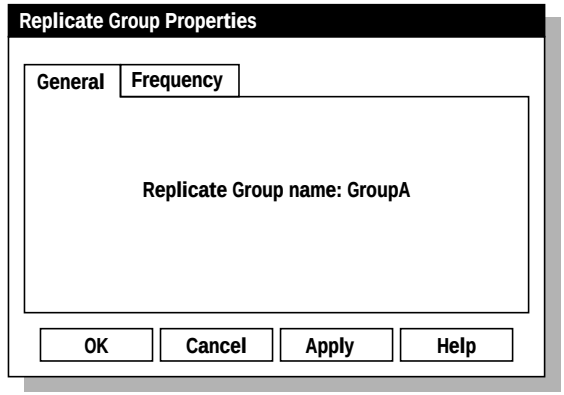
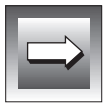


Figure 7-28
Replicate Group Properties Window

You can also use this dialog box to change the frequency of replication. To change the frequency value, click the **Frequency** tab, change the frequency value, and then click **Apply**.



Important: *Enterprise Replication calculates a new replication schedule at the time you select* ***Apply****. For example, if you change the time-based frequency value from every hour to every two hours and select* ***Apply*** *at 12:20 P.M., replication occurs at 2:20 P.M..*

Server Menu

Use the **Server** menu in Figure 7-29 to manage the database servers that you have declared to Enterprise Replication.

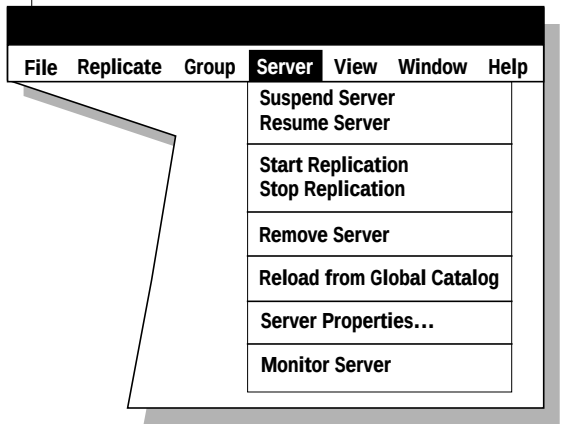


Figure 7-29
Server Menu

Suspending and Resuming a Database Server

To suspend a database server, select the database server and choose **Server→Suspend Server**. **Suspend Server** suspends the delivery of replication to a database server.

To resume a database server from the **Server** menu, select the database server and choose **Server→Resume Server**. **Resume Server** resumes the delivery of replication to a database server.

Starting and Stopping Enterprise Replication

Starting or stopping ER on a database server can cause your databases to get out of sychronization. Informix recommends that you start and stop ER only when the databases are not in active use.

Stopping Replication

To stop Enterprise Replication on a single database server instance, select the database server and choose **Server→Stop Replication**. Stopping ER shuts down all ER threads and frees all in-memory structures associated with that database server declared to ER. No global catalog tables are destroyed with this command. However, if information is not stabilized (all replication information is current and synchronized on all active replicates and replicate groups), data might become inconsistent.

Stopping a database server can be useful if you need to resynchronize all tables that are to be replicated. This command might also be appropriate when you want to stop ER in a database server.

Starting Replication

To start Enterprise Replication on a single database server, select the database server on the main window and choose **Server→Start Replication**. When you start ER for a database server, the global catalog tables are reread and all necessary ER threads restarted. If the database server from which you are starting ER is the primary database server, ER restarts the evaluation of the data marked for replication in the logical log. Database server connections are either accepted or created as required based on information in the global catalog. The start command is only effective after a **Stop Replication** command is issued.

Removing a Database Server from Enterprise Replication

Important: *If you remove a database server before the queues are properly emptied, all data that has not been delivered for the database server might also be deleted.*

To remove a database server from ER, select the server and then choose **Server→Remove Server**. The RM asks you to confirm that you want to remove the database server, as in Figure 7-30

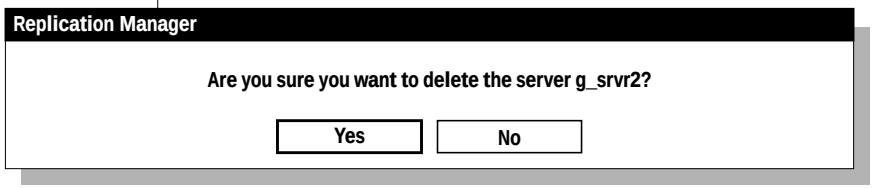


Figure 7-30
Confirm Server Deletion

If you confirm that you want to delete the database server, the RM removes all information about the database server from all ER catalogs. This process can take a few minutes. After completing the deletion, RM asks you to connect to another database server.

Tip: *You need to remove the database server twice to remove all links to replicates and replicate groups in the global catalog.*

Reloading the Global Catalog

To reload the global catalog for a database server, select the database server and choose **Server→Reload from Global Catalog**. Reloading the global catalog ensures that the RM is using up-to-date information about the database server.

Viewing Database Server Properties

To view database server properties, choose **Server→Server Properties...**. **Server Properties...** accesses the Server Properties dialog box..

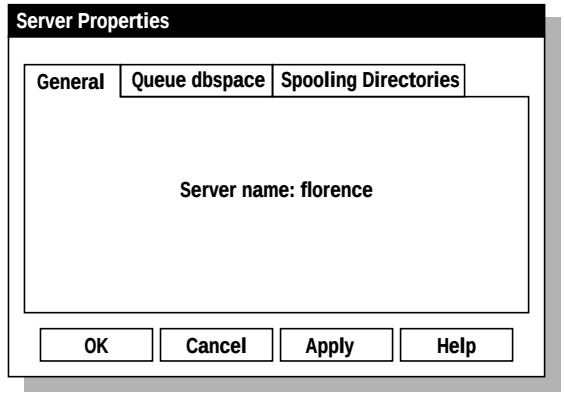


Figure 7-31
Server Properties Dialog Box

The Server Properties dialog box in Figure 7-31 summarizes the properties you have assigned to a database server. Click the **General** tab to review the properties.

The **Queue dbspace** tab lets you modify the location of the Receive queue. You cannot change the location of the Send queue.

The **Spooling Directories** table lets you change the locations of the ATS and RIS directories.

Monitoring Enterprise Replication

To monitor performance, choose **Server→Monitor**. For information on the monitoring capabilities of the Replication Manager, see Chapter 9, "Monitoring Enterprise Replication."

File Menu

The File menu has only two items. The choice **File→Exit** exits from the Replication Manager. The choice **File→Event Monitor** displays the Replication Event Monitor.

Monitoring Replication Events

The Replication Event Monitor notifies you when a warning or severe error occurs on a *remote* database server that is participating in Enterprise Replication. Enterprise Replication does not send error messages from the local database server to the Replication Event Monitor.

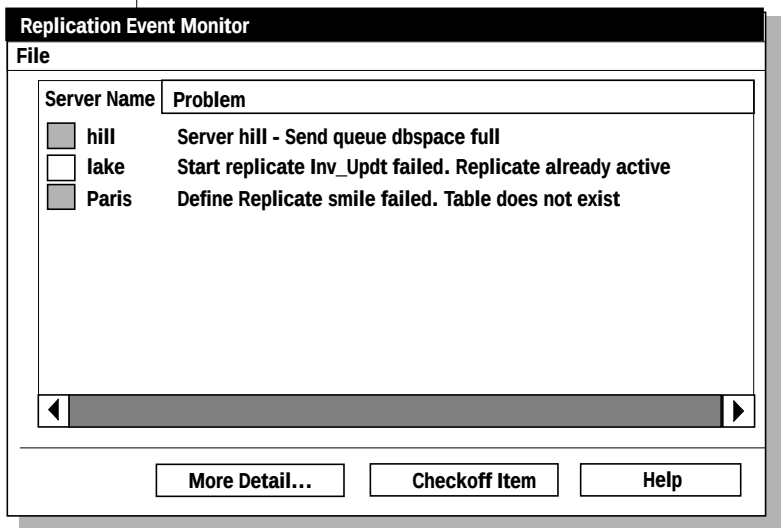


Figure 7-32
Replication Event Monitor

In Figure 7-32, database servers that are experiencing replication errors are listed in the Replication Event Monitor. The shaded box (this box is red on a Windows NT workstation monitor) indicates a severe error. The white box (this box is yellow on a Windows NT workstation monitor) indicates a warning. For more information about an error, select the database server name and then click **More Detail...**.

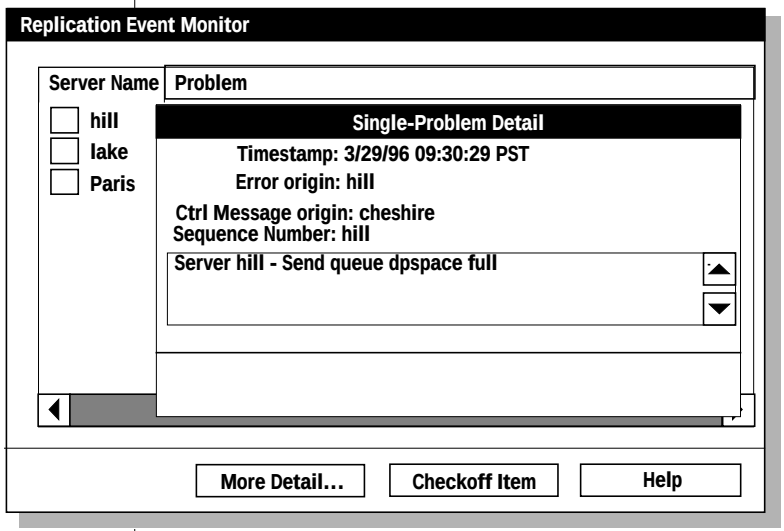


Figure 7-33
Single-Problem Checklist

In Figure 7-33, information is provided about database server **hill**. When you have reviewed this information, click **Close** to return to the Replication Event Monitor.

When you return to the main dialog box of the Replication Event Monitor, you can select database server **hill** and then click **Checkoff Item.** This option marks the error and indicates it was reviewed. From the **File** menu of the Replication Event Monitor you can also delete selected events or export the information to a text file.

View Menu

The **View** menu in Figure 7-34 lets you choose how information is displayed.

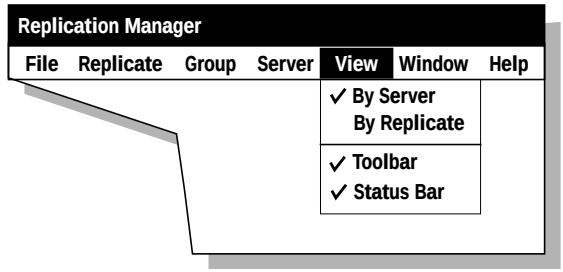


Figure 7-34
View Menu

Use the **View** menu to view information by database server or by replicate. These menu items produce the same results as the replicate and database server icon. You can also choose to view (or hide) the toolbar and the status bar.

The **Toolbar** and **Status Bar** menu items cause the Toolbar and Status Bar to appear or disappear from the display.

Window Menu

The **Window** menu lets you arrange the windows on your display, change Replication Manager modes, and activate different windows.

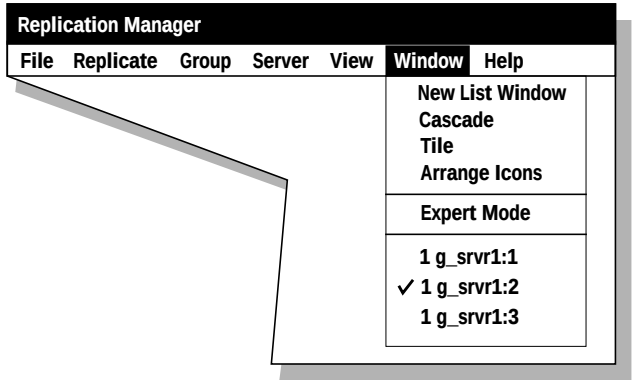


Figure 7-35
Window Menu

Managing the Window Display

Use the **Window** menu in Figure 7-35 to arrange your windows on your workstation. The following options appear on the **Window** menu.

Option	Function
New List Window	Creates a new window. This option is helpful if you want to view different information simultaneously. For example, you can show database server states in one window, and replicate states in another window.
Cascade	Cascades the windows (overlaps windows on top of each other in a hierarchical order).
Tile	Tiles the windows (places them next to each other).
Arrange Icons	Arranges the icons.
Database server name:*number*	Activates the selected window.

Activating a Selected Window

Each time you create a new list window, the RM creates a new window and adds an entry to the numbered list at the bottom of the **Window** menu. To activate a specific window, choose its entry from the numbered list.

Selecting Expert Mode

Expert mode increases the number of options that the Replication Manager menu offers and lets you choose additional types of replications. Choose **Window→Expert Mode** to change the mode. Expert mode is described in Chapter 8, "Using Expert Mode."

Chapter

8

Using Expert Mode

The Enterprise Replication Manager (RM) has two modes, expert and non-expert (or standard).

This chapter describes the advanced features that are available when you use the expert mode of the RM. Expert mode lets you design many-to-one, update-anywhere, and hierarchical replication systems. Expert mode also allows you to specify conflict-resolution rules for transactions.

Tip: *Some of the menu options on expert mode are not available in the current release. If you choose an option that is not available, a pop-up message states that the feature is not supported in this release.*

Chapter 7, "Administering Enterprise Replication," introduces the Replication Manager and describes how to use non-expert mode. Chapter 9, "Monitoring Enterprise Replication," documents the monitoring features of the RM.

You can perform all of the operations that are described in this chapter with the command-line utility, as described in Chapter 11, or with API calls, as described in Chapter 12.

Opening Expert Mode

When you open the Replication Manager, the RM checks to see whether any database servers were declared for Enterprise Replication.

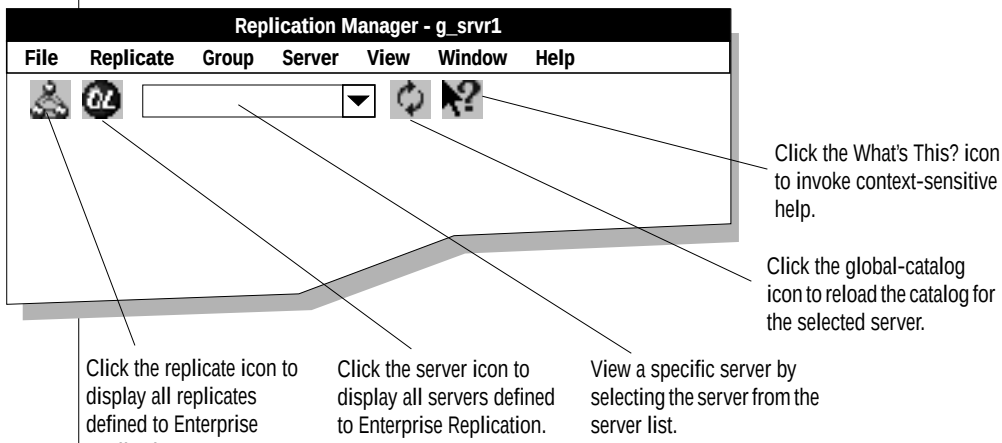
Declaring the First Database Server

If no database servers were declared (that is, if you are starting replication for the first time), you must declare a database server before you can continue. The instructions for declaring the first database server are in "Declaring the First Enterprise Replication Database Server" on page 7-5. You must declare at least one database server before you can enter expert mode.

Choosing Expert Mode

If you have already declared at least one server for Enterprise Replication, the Replication manager displays the main window, illustrated in Figure 8-1. You can enter expert mode by choosing **Window→Expert Mode** on the RM main window.

Figure 8-1
Replication Manager Main Window



Declaring a Server in Expert Mode

To declare a database server, choose the database-server group from the server list on the tool bar. If that server was not already declared, the RM asks if you want to bring up ER on that server. When you click **Yes**, the Declare Server dialog box in Figure 8-2 on page 8-8 appears.

The dialog box in Figure 8-2 asks you to declare a database server and its attributes to Enterprise Replication.

- Server type

  The drop-down list box shows the types that you can assign to ER database servers. For a discussion of the different types of database servers, refer to Chapter 3, "Understanding Enterprise Replication Objects, Topologies, and Processes."

- Synchronization server

  Enterprise Replication uses the synchronization server to synchronize the global catalogs on each database server participating in a replicate or replicate group. When you identify a database server as the synchronization server, this indicates to ER that the database server has the authoritative global catalog. When you declare a subsequent database server to a replicate, ER copies the authoritative global catalog to the subsequent database server during synchronization.

- ATS and RIS directories

  The aborted-transaction spooling (ATS) and row-information spooling (RIS) files contain information about failed transactions. These files are documented in Chapter 10, "Diagnosing Enterprise Replication."

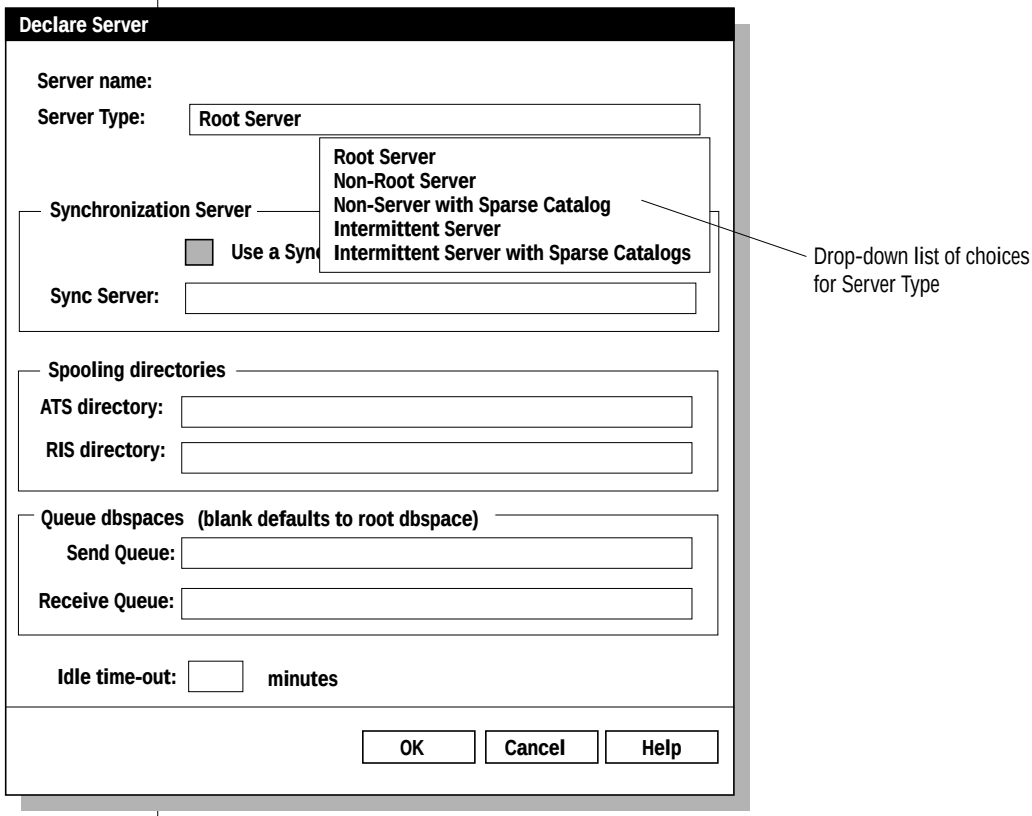
- Queue dbspaces

  For information about the send and receive queue dbspaces, refer to "Send and Receive Message Queues" on page 5-6.

- Idle time out

  The idle time out specifies the minutes you allow a database server to remain idle before timing out the connection to that database server.

Figure 8-2
Declare Database Server in Expert Mode



Defining a Replicate

To define a replicate, choose **Replicate→Define Replicate.**

The **Define Replicate...** menu item accesses the Define Replicate wizard. The Define Replicate wizard provides a series of windows that guide you through the process to define a replicate.

The introductory wizard page in Figure 8-3 lists the steps required to define a replicate.

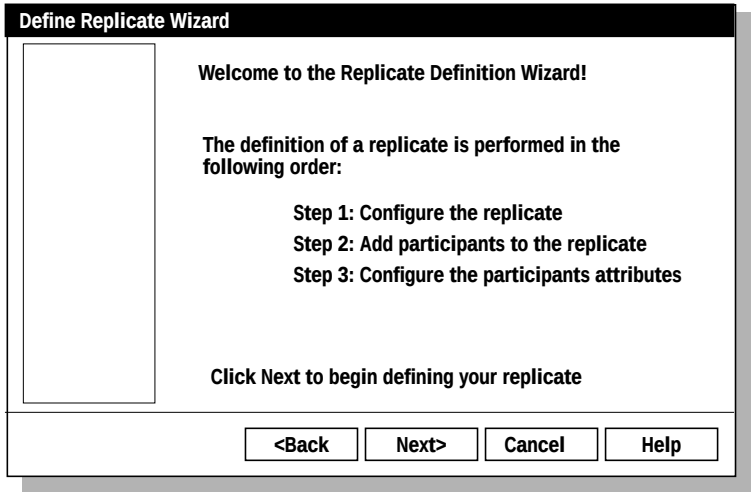


Figure 8-3
Define Replicate Wizard Welcome Dialog Box

Defining a Replicate Name

The next wizard page, in Figure 8-4, requests a name for the replicate and asks you to define the type of replication.

You can create an update-anywhere or a primary-target replicate. For information about the replicate types, refer to Chapter 4, "Designing Enterprise-Replication Systems."

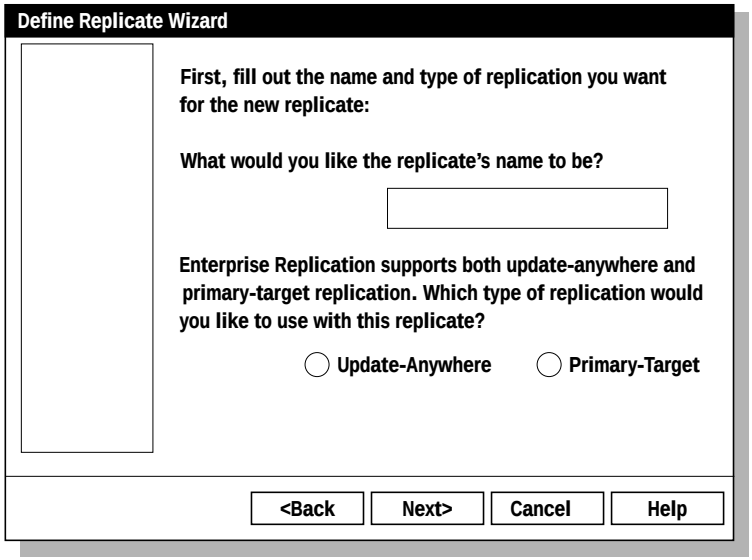


Figure 8-4
Define Replicate Name

Selecting Conflict-Resolution Attributes

When you define an update-anywhere replicate, you must specify conflict-resolution rules. The Define Replicate wizard page in Figure 8-5 on page 8-11 appears only when you define an update-anywhere replicate. For information about conflict-resolution rules, see "Choosing Conflict-Resolution Rules and Scopes" on page 3-24.

If you choose to use a stored procedure, specify the full pathname of the stored procedure. You also must determine whether execution should be optimized. If the execution of the stored procedure is optimized, the stored procedure is called only if the detected collision cannot be resolved. If you choose not to optimize execution, the stored procedure is called each time a collision is detected.

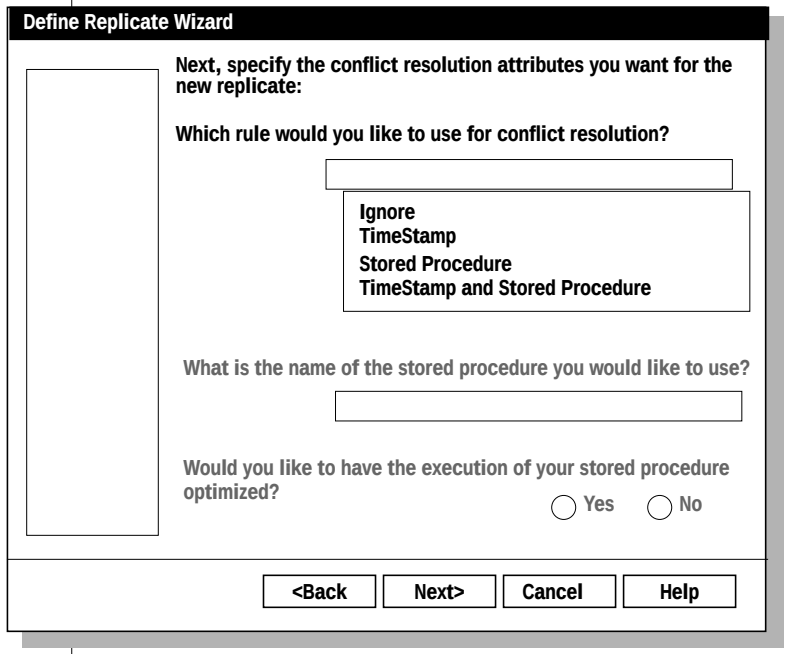


Figure 8-5
Select Conflict-Resolution Attributes

Shadow Columns

If you select any conflict-resolution option except **Ignore**, the table that you are replicating must have *shadow columns* that keep conflict-resolution information. If your table does not have shadow columns, you will see the error message in Figure 8-6. (This error does not appear until you finish describing the replicate and attempt to create it. However, the choice you make in Figure 8-5 determines whether or not the error might appear.)

Figure 8-6
Shadow Column Error Message

To create shadow columns for your table, use the SQL statement CREATE TABLE or MODIFY TABLE with the CRCOLS clause. For information about these commands, refer to the *Informix Guide to SQL: Syntax*.

Selecting Replicate Options

The next wizard page allows you to set options for the new replicate, as shown in Figure 8-7.

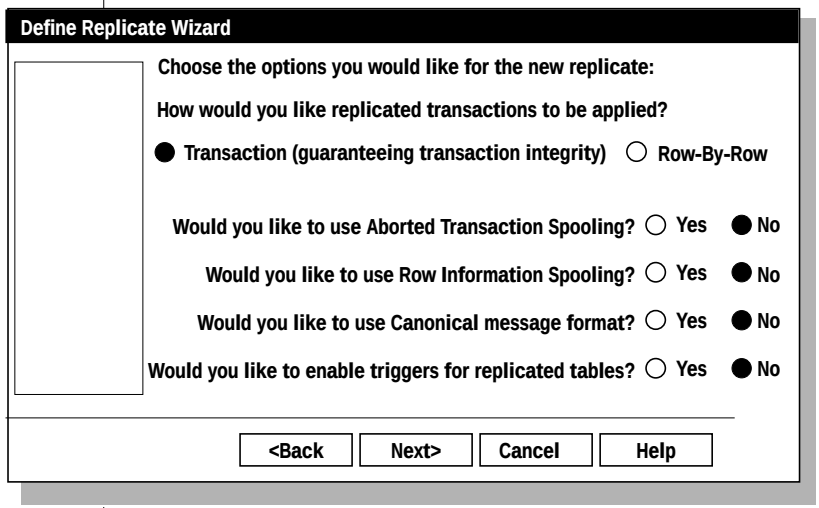


Figure 8-7
Select Replicate Options

You can specify the following options for the replicate:

- Application of Transactions

  How you apply transactions refers to the scope of how conflict-resolution rules affect replication in a replicate. For more information on conflict-resolution scope, see "Choosing Conflict-Resolution Rules and Scopes" on page 3-24.

- Aborted-Transaction Spooling

  Select Aborted-Transaction Spooling to log transactions that cannot be applied at the target. If a transaction fails for any reason, all of the buffers in the replication message that compose the transaction are written to the aborted-transaction spool file. See Chapter 10, "Diagnosing Enterprise Replication," for more information.

- Row-Information Spooling

  Select Row-Information Spooling to log rows that are not replicated. If a row fails conflict resolution (including replication-order error failures), the row is written to the row-information spooling file. See Chapter 10, "Diagnosing Enterprise Replication," for more information.

- Canonical-Message Format

  If you are replicating data between heterogeneous hardware and are replicating float or small-float data types, you must select canonical-message format. Enterprise Replication translates the replication message information for these data types to a machine-independent format for transfer between the dissimilar hardware platforms.

- Database Triggers

  Select the database-trigger option if you wish triggers to process on your target database servers.

  For example, assume you have a retail application that fires a trigger every time a new customer is added to the customer table. When a new customer is added, the trigger fires a process that creates a 10 percent discount for the new customer. If this database trigger is replicated at each database-target server, your new customer could receive many 10 percent discounts.

Select the Replication Frequency

The next wizard page asks you to specify whether you want replication to be continuous or time based. With the time-based option, you can choose the time (in hours and minutes) and day (day of week, day of month) that data replication occurs in a replicate.Time-based replication begins when you start the replicate.

In the example in Figure 8-8, time-based replication will occur every Sunday at 3:45 P.M. For more information about the frequency options, refer to "Frequency Options" on page 7-13.

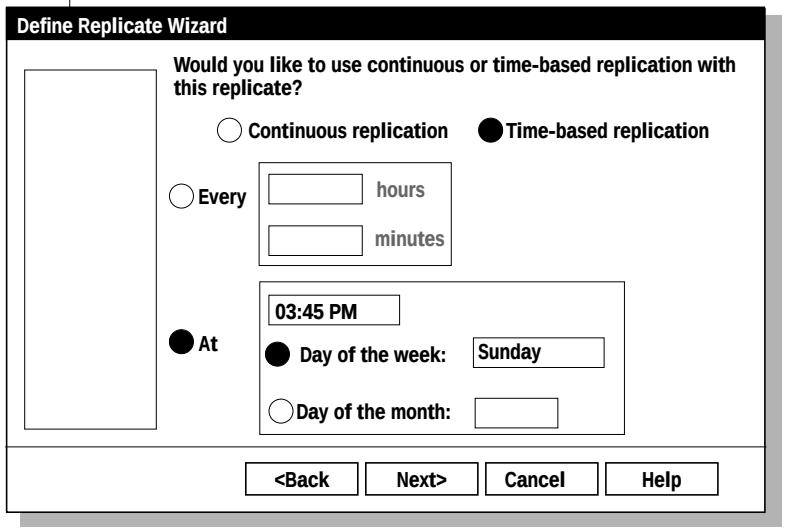


Figure 8-8 *Select Continuous or Time-Based Replication*

Important: *Time-based replication is based on the local time, not Greenwich Mean Time (GMT). If database servers participating in the replicate are in different time zones than the database server on which the time is set, replication will occur at different times.*

Selecting Database Servers to Participate in the Replicate

After you specify the replication frequency, you must select the database servers that you want to participate in the replicate. Update-anywhere replication and primary-target replication have different wizard pages.

Selecting Database Servers for Update-Anywhere Replication

All database servers in an update-anywhere replicate must have read / write capabilities to allow peer-to-peer replication.

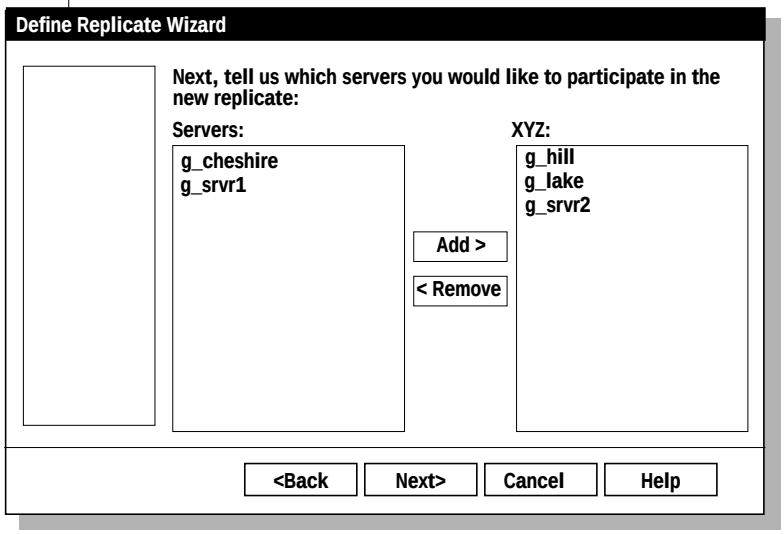


Figure 8-9
Select Database Servers for a Replicate

All servers that are declared to Enterprise Replication are listed in the **Servers** list box. To select the database servers for your replicate, click on the database server name (or a group of database server names) and click **Add**. To remove a database server from your replicate, click on the database server name (or a group of database server names) and click **Remove**. Figure 8-9 on page 8-16 shows that **hill**, **lake**, and **srvr2** have been added to replicate XYZ.

Selecting Database Servers for Primary-Target Replication

The Define Replicate wizard page in Figure 8-10 asks you to select the database servers you want to add to the primary-target replicate.

All database servers that are declared to Enterprise Replication are listed in the **Servers** list box. The database servers you assign to primary participant or target participant reflect the business model that the primary-target replicate is using.

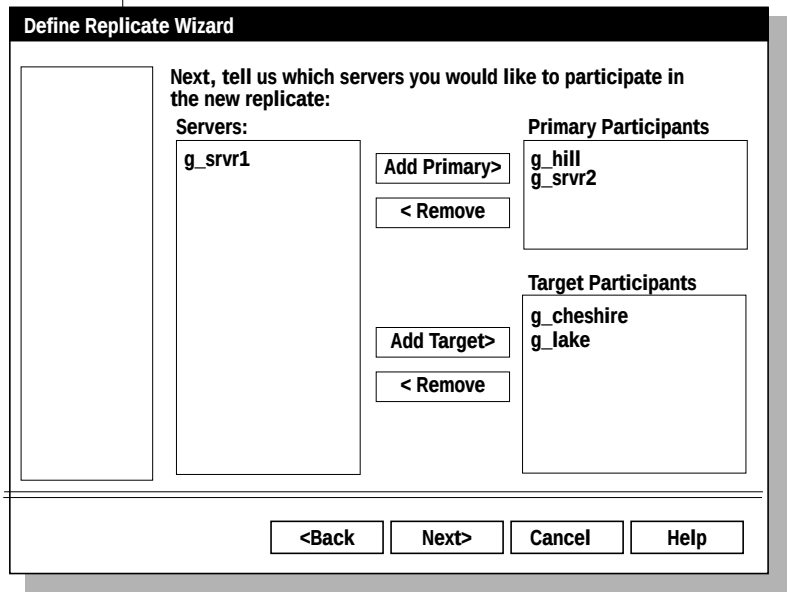


Figure 8-10 *Select Database Servers for a Primary-Target Replicate*

Specifying Identical or Non-Identical Participants

The Define Replicate wizard page in Figure 8-11 asks whether the participant attributes are identical. A participant consists of a database server and the participant attributes: the table name, owner of the table, database in which the tables resides, and the SELECT statement. If all the participant attributes are identical, click **Yes**. If, on the other hand, the databases in the replicate have identical columns, but one database is named **mansions** while the other database is named **villas**, then the attributes are not identical and you must click **No**.

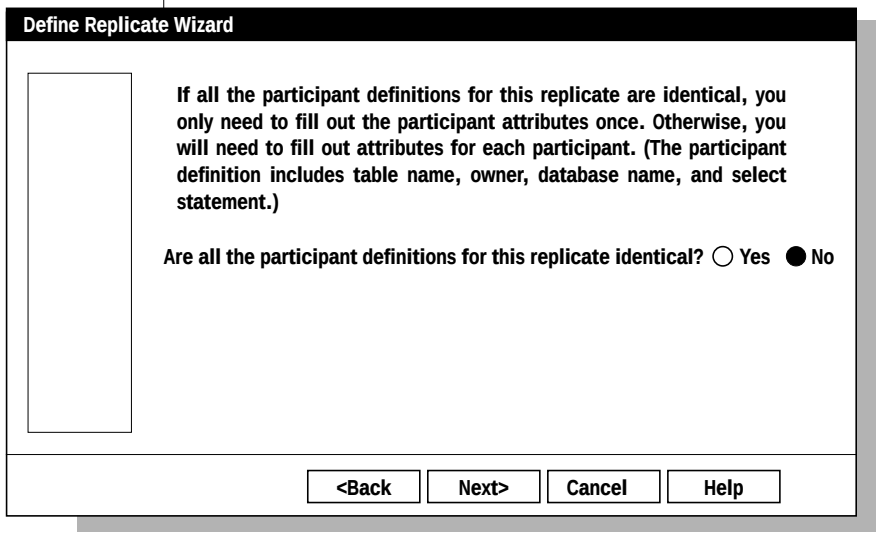


Figure 8-11
Describe Attributes and Select Source

Defining Participant Attributes

The next wizard pages ask you to define the table you want to replicate, the owner of the table, and the name of the database in which the table resides.

If your participants are identical, you only need to go through this set of pages once. If your participants are not identical, you must fill in the same set of pages for each participant in the replicate. For a more detailed discussion of how to define participant attributes, refer to "Defining Participant Attributes" on page 7-17.

Important: *Enterprise Replication places some restrictions on certain ALTER and SET statements you can issue to tables that are participants in a replicate. For more information, on SQL statement usage, see "Setting Up Your Replication Environment" on page 5-1.*

Specifying Participant Attributes

A wizard page, such as the one in Figure 8-12 on page 8-19, asks you to enter the attributes for your participant.

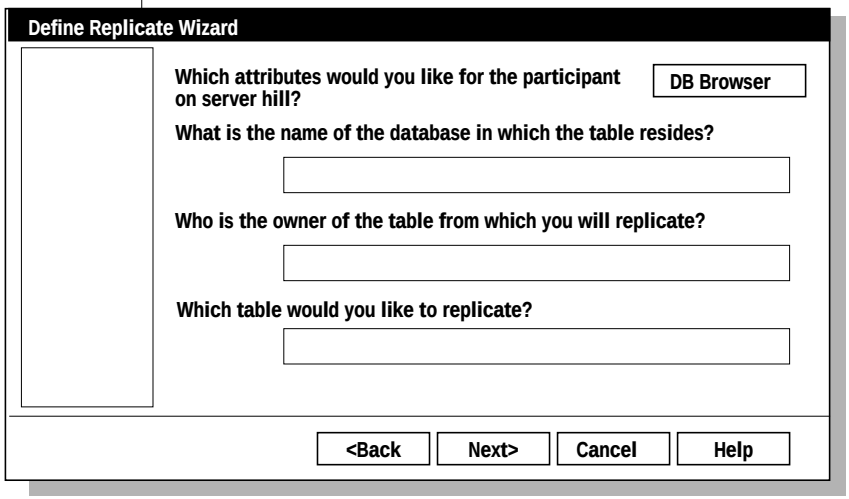


Figure 8-12 *Define Data Attributes*

Using the DB Browser Button

The **DB Browser** button appears in both Figure 8-12 on page 8-19 and Figure 8-13 on page 8-21. You can use it to select the database, table, and columns for replication.

Choosing the Columns to Replicate

The wizard page in Figure 8-13 asks you to select columns for replication. An asterisk appears in the column selection (the first text box) to indicate that you want to select all columns. To select individual column names to replicate, type the name of each column. The column names you select (either specific or all) are automatically added to the SELECT statement in the second box.You can also create a WHERE clause to define the exact data you want to replicate in your replicate.

If you used the **DB Browser** button on the Define Attributes display (Figure 8-12 on page 8-19), you might have selected columns to replicate. If not, the wizard page in Figure 8-13 on page 8-21 allows you to either type in the selected columns, or to use the **DB Browser** button to choose the columns for replication.

Important: *You must include a primary key in your column selection. You cannot use a join or aggregated columns in the SELECT statement.*

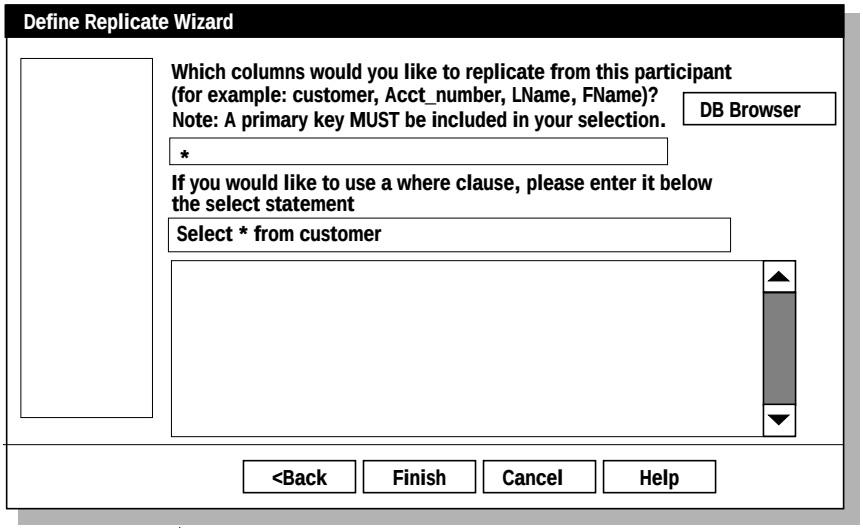


Figure 8-13
Select Data for Replication

Enter a WHERE Clause

After you choose the columns for replication, you can enter a WHERE clause in the lower panel to define the exact data you want to replicate in your replicate.

Informix recommends that you create a WHERE clause on all participants. If a primary participant experiences hardware errors or has to be removed from a replicate because of connectivity problems, you already have the target participant defined correctly and can easily change the target to a primary participant.

Column Mapping

You can perform column mapping with your column selection. Each participant in the replicate must define the identical number of columns (n to n mappings) and the data types must be the same. See "Column Mapping" on page B-5 for more information.

The Replicate Summary

After you finish entering information about the participants, the RM shows a summary of the replicate that you just defined, such as in Figure 8-14.

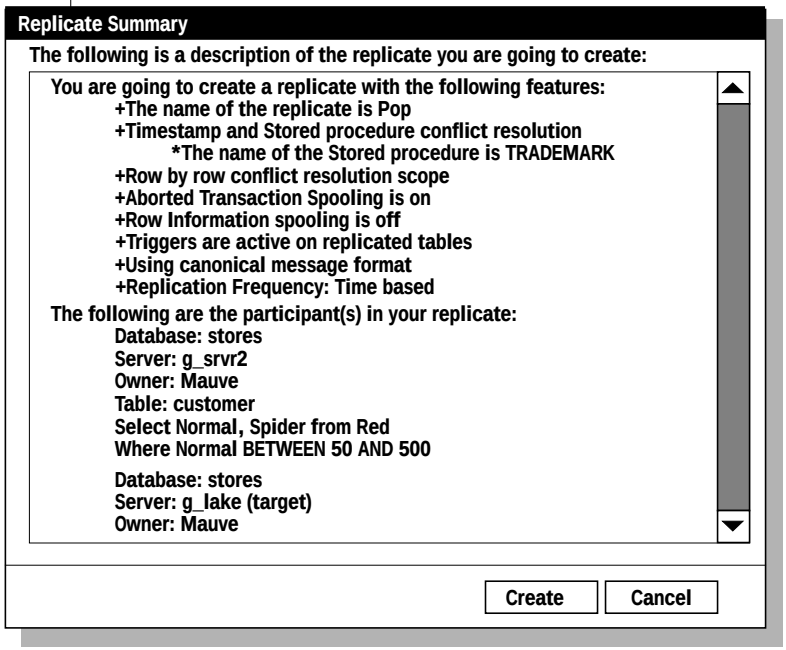


Figure 8-14
Replicate Summary Window

The Replicate Summary in Figure 8-14 provides a summary of the attributes you selected for your replicate. Use the scroll bar to review all information for each database server that participates in your replicate. Click **Create** to create the replicate.

If you discover errors, click **Cancel**. You are returned to the window where you select data for replication (Figure 8-13 on page 8-21). Click **<Back** to find the appropriate window and make the necessary change(s).

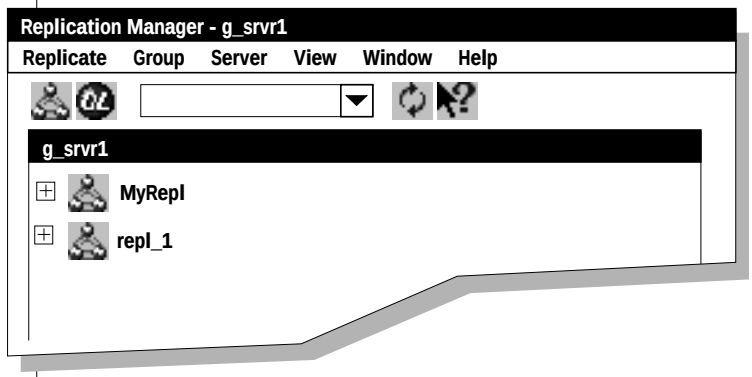
After you create the replicate, the RM returns to the main window, as shown in Figure 8-15. A newly created replicate is inactive. For information about changing the replicate state to active, refer to "Changing Replicate States" on page 8-27.

Important: *Do not create a clustered index on a table after you have defined a replicate on that table; replication on the table will cease.*

Using the Replicate Menu

In the main window, you can click the **Replicate** icon (at the left end of the tool bar) to see replicates and participants, as illustrated in Figure 8-15. A newly-created replicate is inactive.

Figure 8-15
Displaying Database Server Status



The **Replicate** menu in Figure 8-16 allows you to define and control replicates.

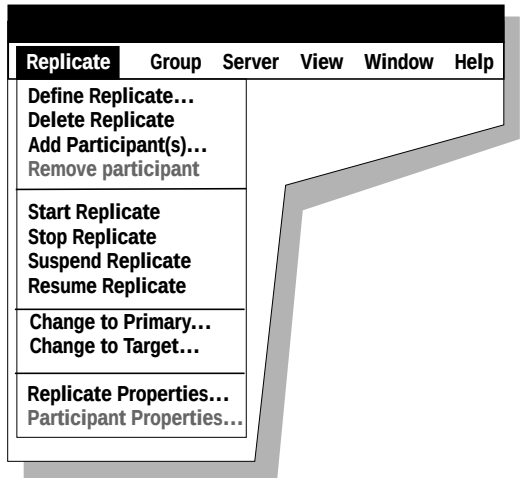


Figure 8-16
Replicate Menu

A menu item that contains an *ellipsis* (…) (for example **Define Replicate…**) indicates that a wizard, dialog box, or a property sheet follows to help you complete a task. All other items on the Replicate menu are equivalent to the command-line utility options (see Chapter 11) and the API calls (see Chapter 12).

Tip: *To guarantee an accurate display of replication objects, select the database server from the database server list. (When you select the database server from the database server list, Replication Manager reloads the global catalog.)*

Managing Replicates and Participants

The first four items on the Replicate menu allow you to define and delete replicates and to add and remove participants.

Defining a Replicate

To define a new replicate, choose **Replicate→ Define Replicate**. "Defining a Replicate" on page 8-9 describes the steps for defining a replicate.

Deleting a Replicate

To delete a replicate, select the replicate on the main window of the Replication Manager, and then choose **Replicate→Delete Replicate**. When ER receives this command, it immediately purges all data from the queues associated with the replicate and removes the replicate from the global catalog.

Important: *If you delete an active replicate, Enterprise Replication cannot guarantee consistent tables unless the replicate is stopped and the queues are properly emptied.*

Adding a Participant

To add participant(s) to an existing replicate, select the replicate from the main window of the Replication Manager, and choose **Replicate→Add Participant(s)…**.

Figure 8-17
Add Participant Wizard Welcome Page

The Add Participant wizard page in Figure 8-17 lists the steps necessary to add participant(s) to an existing replicate.

The wizard page in Figure 8-18 lets you choose the server(s) to add to the replicate.

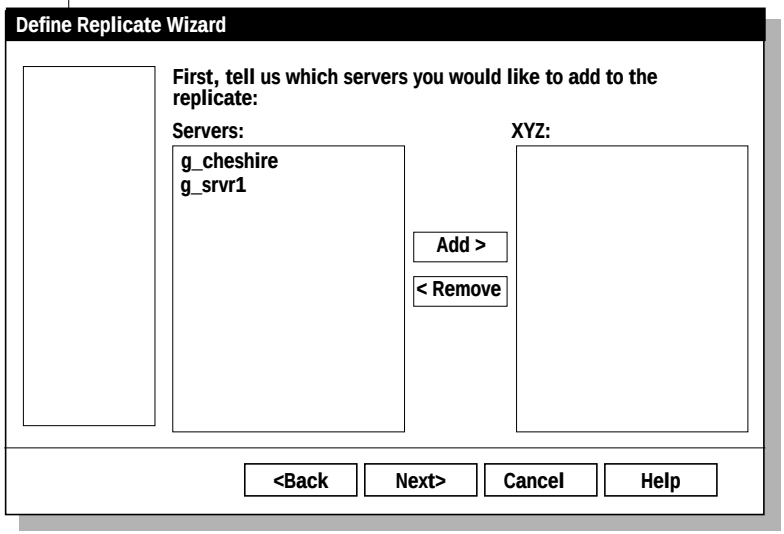


Figure 8-18
Select Servers to Add to the Replicate

The Add Participant wizard continues with steps for defining the attributes of participants as described in "Defining Participant Attributes" on page 8-19.

Warning: *If you add a participant to an existing replicate, you must synchronize the databases before you start replication with the new participant. For information on how to synchronize your databases, see Chapter 6, "Preparing Data for Replication."*

Removing a Participant

To remove a participant from a replicate, click the replicate to display the participants. Select the participant that you want to remove and then choose **Replicate→Remove Participant**. The RM asks you to verify that the participant should be removed, as in Figure 8-19.

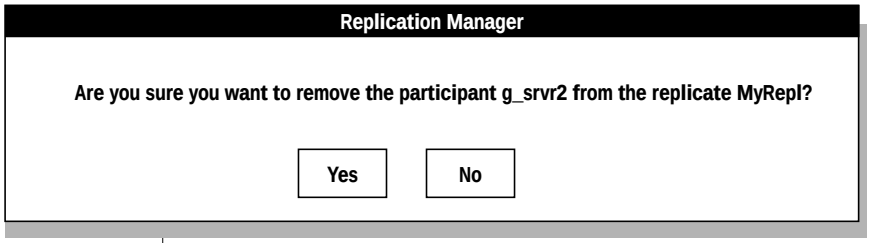


Figure 8-19 *Remove Participant Confirmation*

Changing Replicate States

The second group of choices on the Replicate menu allow you to change the state of a replicate. A replicate can be in one of the following states, which are describedin "Changing Replicate States" on page 26:

- Inactive
- Active
- Suspended
- Quiescent

Changing the Type of a Participant

The next two items on the Replicate menu let you change the type of a participant from primary to secondary (or from secondary to primary). When you chose **Replicate→Change to Primary** or **Replicate→Change to Secondary**, the RM displays a dialog box that asks you to select the participants whose state will change.

When you change the status of a participant, you might change the nature of the replication. If that is the case, the RM displays a warning similar to the warning in Figure 8-20 that asks if you really want to make this change.

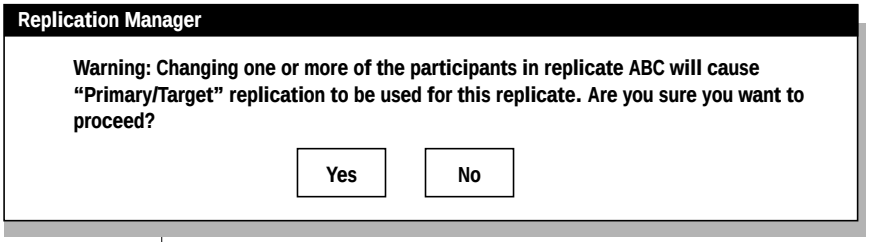


Figure 8-20
Warning Message for Changing Replicate Type

Viewing Properties

The final two choices on the **Replicate** menu let you view the properties of replicates and their participants.

Replicate Properties

The Replicate Properties window summarizes the properties you have assigned to a replicate.To view replicate properties, select a replicate and then choose **Replicate→ Replicate Properties**. The RM displays the properties of the replicate, as shown in Figure 8-21.

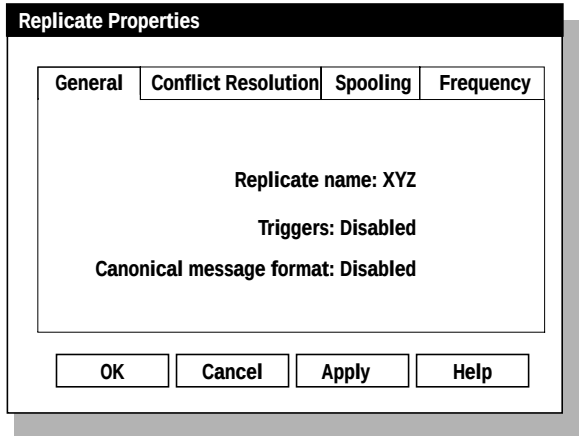


Figure 8-21
Replicate Properties Window

You can also use this window to change the frequency of replication. To change the frequency, click the **Frequency** tab, change the frequency, and then click **Apply**.

Important: *Enterprise Replication calculates a new replication schedule at the time you select* ***Apply****. For example, if you change the time-based frequency value from every hour to every two hours and select* ***Apply*** *at 12:20 P.M., replication occurs at 2:20 P.M.*

Participant Properties

The Participant Properties window shows the columns and the WHERE clause selected for the participant.To view participant properties, select a participant and then choose **Replicate→ Participant Properties**. The RM displays the properties of the participant, as shown in Figure 8-22.

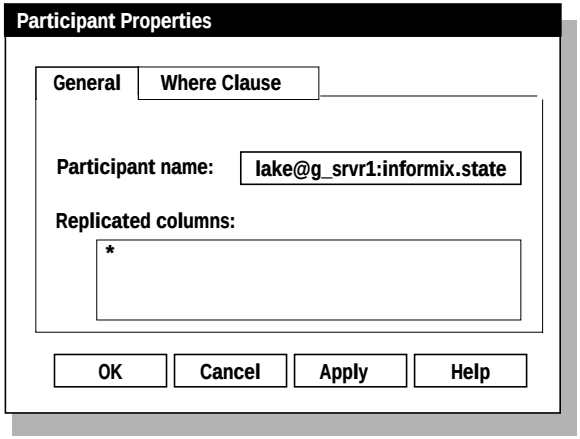


Figure 8-22
Participant Properties Window

Group Menu

Use the **Group** menu in Figure 8-23 to define and control replicate groups and review replicate group properties.

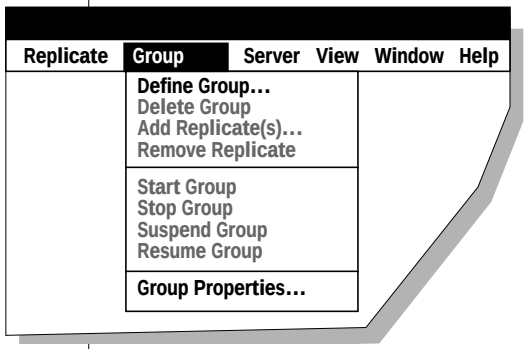


Figure 8-23
Group Menu

Defining a Replicate Group

To define a replicate group, choose **Group→Define Group**. The RM starts the Define Replicate Group wizard, as shown in Figure 8-24. The Define Replicate Group wizard provides a series of pages that guide you through the process to define a replicate group.

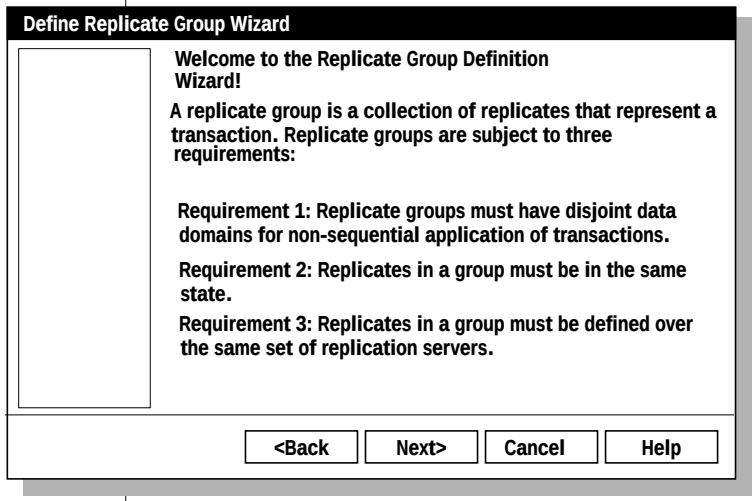


Figure 8-24 *Define Replicate Group Wizard Welcome Page*

Choose a Group Name

In Figure 8-25 on page 8-32, the wizard asks you to choose a name for the replicate group and whether data should be applied sequentially.

Enter the name for your new replicate group. You can click the check box to turn on sequential application. For more information about parallel and sequential application, refer to "Replicate Group" on page 3-7.

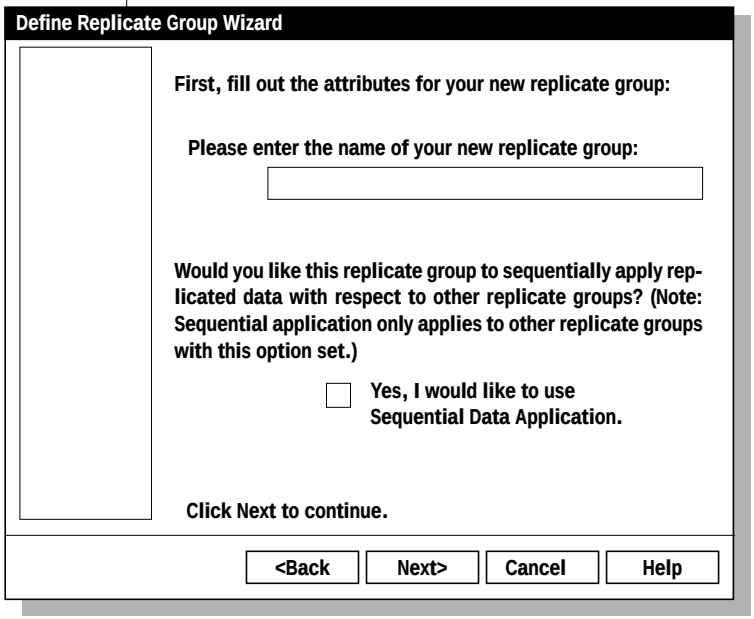


Figure 8-25
Define Replicate Group Name

Choose the Replication Interval for the Group

Next the wizard asks you to choose a replication interval for the replicate group. The display for specifying the replication interval for the group is the same as the display for specifying the replication interval for a replicate, shown in Figure 8-8 on page 8-15.

Important: *Each replicate in a replicate group* ***must*** *have the same replication frequency (or all replicates must have continuous replication enabled).*

Select Replicates for the Replicate Group

The Define Replicate Group wizard page in Figure 8-26 asks you to select the replicates you want to participate in the replicate group. All replicates that are defined to Enterprise Replication are listed in the **Replicates** list box. Select the replicates for your replicate group.

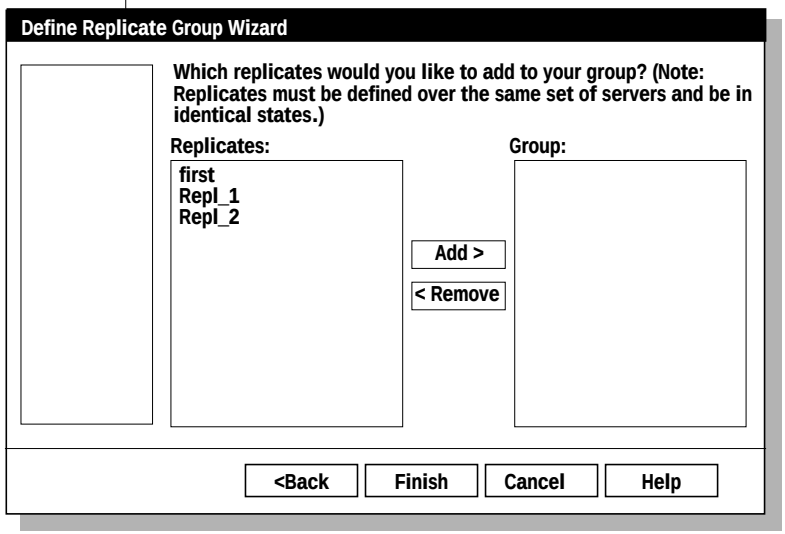


Figure 8-26
Select Replicates for the Replicate Group

If you try to select replicates that have different replication frequencies, the RM notifies you that you have selected inconsistent replicates. In Figure 8-27, the selected participants have different replication frequencies.

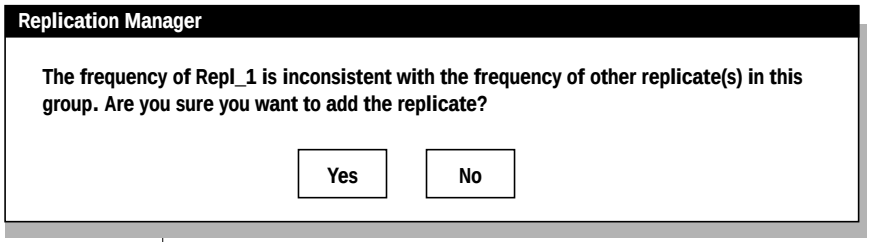


Figure 8-27
Inconsistent Frequency Warning

Modifying Replication Groups

After you have defined at least one replication group, you can use the choices on the Group menu to modify the groups.

Deleting a Replicate Group

To delete a replicate group, select the replicate group. Choose **Group→Delete Group**. When ER receives this command it deletes the replicate group from the global catalog. The **Delete Group** command does not affect the replicates or the associated data in the replicate group. If you have time-based replication enabled for the replicate group, the replicates defined to the replicate group retain the time-based frequency that was set for the replicate group.

Adding Replicate(s) to an Existing Replicate Group

To add replicates to an existing replicate group, select the replicate group and then choose **Group→Add Replicates**. The Add Replicates dialog box in Figure 8-28 lets you add new replicates to a group or remove current replicates from a group.

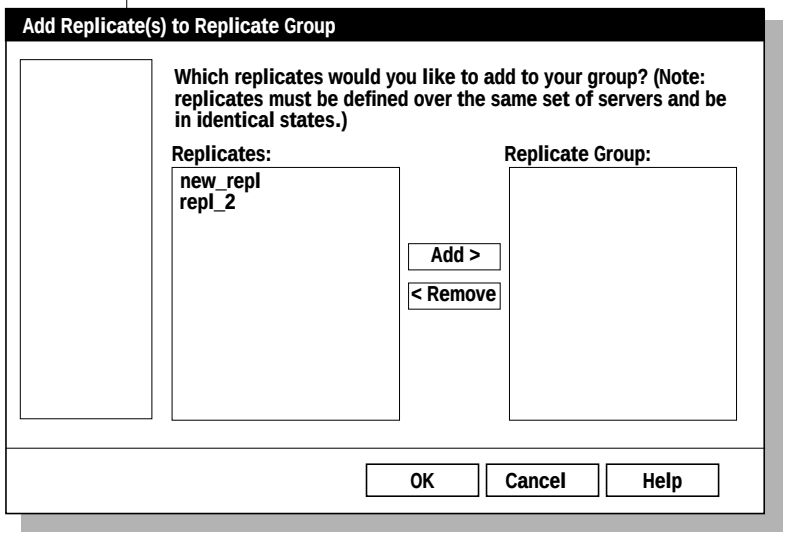


Figure 8-28
Add Replicates to an Existing Replicate Group

The **Replicates** list box shows all replicates that are defined to ER that are not already included in the replicate group. To select the replicates for your replicate group, select the replicate name (or a group of replicates) and click **Add**.

Removing Replicate(s) from an Existing Replicate Group

To remove a replicate from an existing replicate group, select the replicate in the replicate group and then choose **Group→Remove Replicate**.

Changing the State of a Group

The second group of choices on the Group menu allow you to change the state of a group. A group can be in one of the following states:

- Inactive
- Active
- Suspended
- Quiescent

For more information about states, refer to "Changing the State of a Group" on page 7-35.

Viewing Replicate Group Properties

The Replicate Group Properties dialog box summarizes the properties you assigned to a replicate group. To view replicate group properties, select the replicate group and then choose **Group→Group Properties**. The RM displays a property sheet that shows the properties of the replicate group, as illustrated in Figure 8-29.

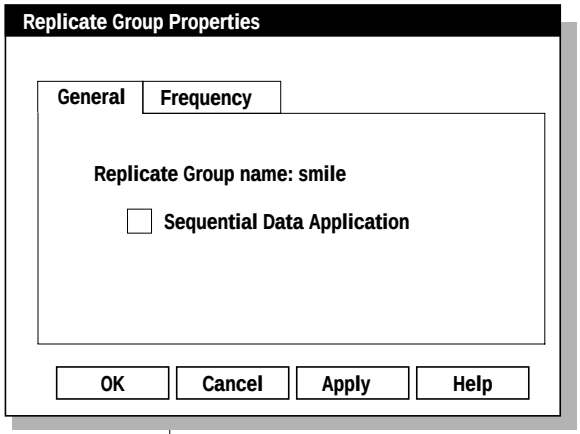


Figure 8-29
Replicate Group Properties Window

You can also use this dialog box to change the frequency of replication. To change the frequency value, click the **Frequency** tab, change the frequency value, and then click **Apply**.

Server Menu

Use the **Server** menu in Figure 8-30 to manage the database servers that you declared to Enterprise Replication.

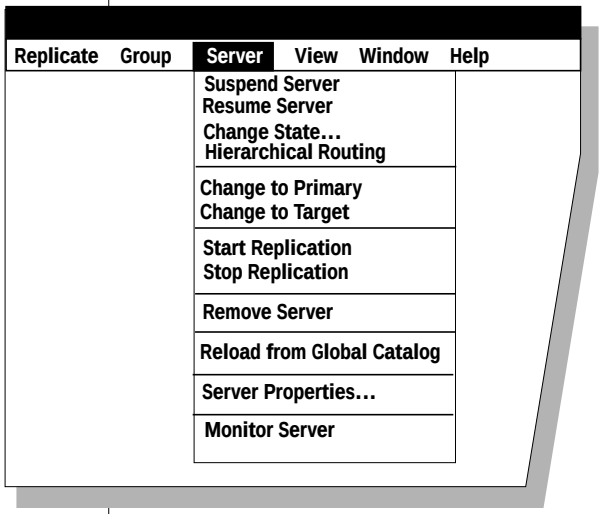


Figure 8-30
Server Menu in Expert Mode

Suspending and Resuming a Database Server

To suspend a database server, select the database server and choose **Server→Suspend Server**. **Suspend Server** suspends the delivery of replication to a database server.

To resume a database server from the **Server** menu, select the database server and choose **Server→Resume Server**. **Resume Server** resumes the delivery of replication to a database server.

Changing the State of a Database Server

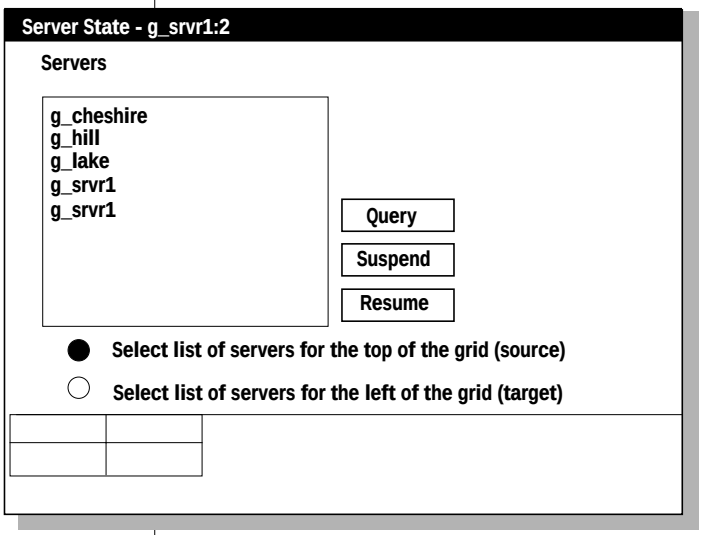
Replication Manager provides several ways for you to change the state of a database server. You can suspend or resume a database server through the selection of menu items in the **Server** menu or you can use the Change State wizard.

To suspend a database server from the **Server** menu, select the server from the main window of the Replication Manager and choose **Server→Suspend Server**. **Suspend Server** suspends the delivery of replication to a database server.

To resume a database server from the **Server** menu, select the server from the main window of the Replication Manager and choose **Server→Resume Server**. **Resume Server** resumes the delivery of replication to a database server.

To change the state of a database server from the Change State dialog box, choose **Server→Change State…**.

Figure 8-31
Changing Database Server States



The window in Figure 8-31 on page 8-38 allows you to query, suspend, and resume database servers that were previously declared to Enterprise Replication. You use the grid to query and change the states of database servers.

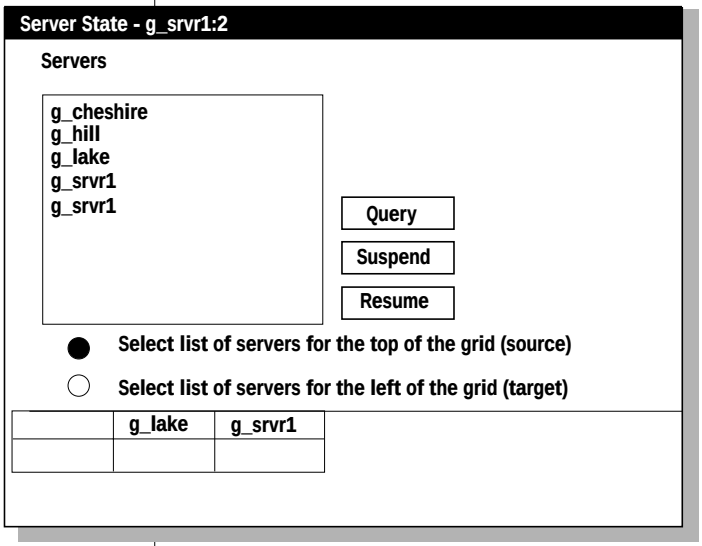
The following examples explain how to use the Change State dialog box to query and change the state of two database servers: **lake** and **srvr1**. Each database server can send *and* receive replicated data. Therefore, you need to be able to query and control communication in both directions on each database server.

Select Servers for the Top of the Grid

To select source servers for the top of the grid, select the desired servers in the **Servers** list box. (You can also remove a database server from the grid by selecting the server.) In Figure 8-32, servers **g_lake** and **g_srvr1** are selected for the top grid.

Important: *Each Enterprise Replication database server can have two states of replication (send and receive)**.** Therefore, you need to select both **g_lake** and **g_srvr1** as source database servers.*

Figure 8-32
Selecting Database Server for the Top Grid

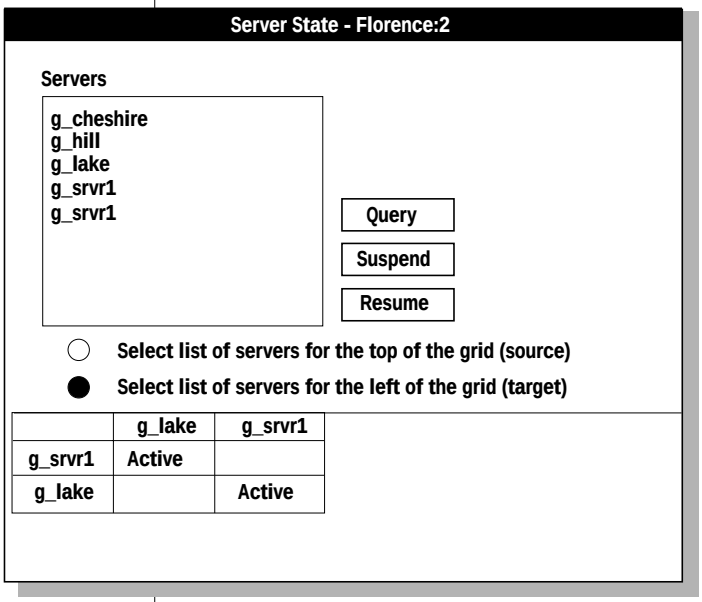


Select Servers for the Left of the Grid

To select target servers for the left of the grid, select the desired servers in the **Servers** list box. (You can also remove a server from the grid by selecting the server.) In Figure 8-33, servers **g_srvr1** and **g_lake** are selected for the left grid.

Important: *Each Enterprise Replication database server can have two states of replication. Therefore, you need to select both* ***g_srvr1*** *and* ***g_lake*** *as target servers.*

Figure 8-33
Selecting Database Server for the Left Grid



The states of the database servers are displayed in the grid. The example shows that the state of the both database servers is active.

Querying, Suspending, and Resuming Database Servers

To query, suspend, or resume a database server, you first need to select a database server from the grid. Select a database server and then click **Query**, **Suspend**, or **Resume**.

The example in Figure 8-34 shows two results from the previous actions of an administrator:

- The administrator selected the source server **g_lake** (the administrator clicked **g_lake** on the top of the grid) and then clicked **Suspend**. The grid shows that **g_lake** is suspended on **g_srvr1**, which means **lake** is not sending replication messages to **srvr1**.
- The administrator selected the source server **g_srvr1** and then clicked **Query**. The grid shows that **g_srvr1** is active on **g_lake**, which means that database server **srvr1** is sending replication messages to database server **lake**.

Figure 8-34
Suspend Database Server Example

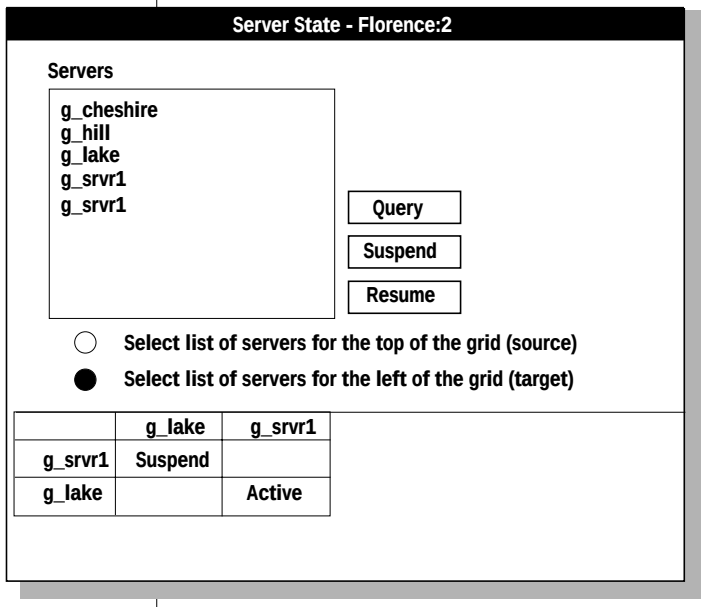


Figure 8-35 illustrates the replication activity on database servers **srvr1** and **lake**. Database server **lake** is capturing the replicated messages sent from database server **lake** (this communication direction is active), but database server **lake** does not send replicated messages to database server **srvr1.** (This communication direction is Suspended.)

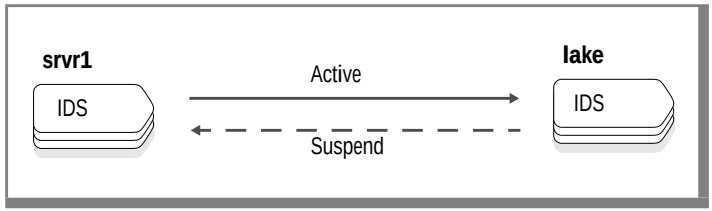


Figure 8-35
Replication Activity on srvr1 and lake

Hierarchical Routing

The menu choice **Server→Start Replication** displays the hierarchy of the replication database servers. A telephone icon represents an intermittent server. A database-server icon with little wires coming out the bottom represents a non-root server.

Changing Between Primary and Secondary

To change all participants on a server from primary to target, select the server and choose **Server→Change to Target**. To change all participants on a server from target to primary, select the server and choose **Server→Change to Primary**. For both actions, the RM asks you to verify that you want to take that action, as in Figure 8-36.

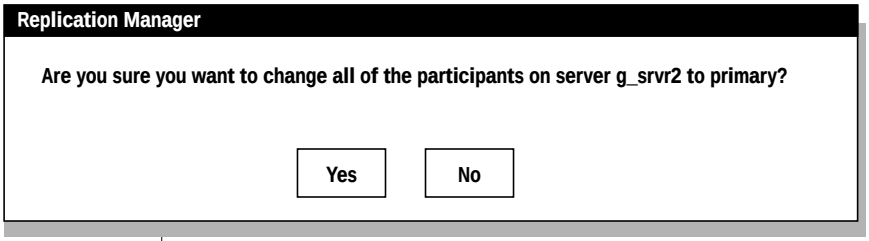


Figure 8-36
Inconsistent Frequency Warning

Starting and Stopping Enterprise Replication

Starting or stopping ER on a database server can cause your databases to become out of synchronization. Informix recommends that you start and stop ER only when the databases are not in active use.

Stopping Replication

To stop Enterprise Replication on a single database server instance, select the database server and choose **Server→Stop Replication**. Stopping ER shuts down all ER threads and frees all in-memory structures associated with that database server declared to ER. No global catalog tables are destroyed with this command. However, if information was not stabilized (all replication information is current and synchronized on all active replicates and replicate groups), data might become inconsistent.

Stopping a database server can be useful if you need to resynchronize all tables that are to be replicated. This command might also be appropriate when you want to stop ER within a database server.

Starting Replication

To start Enterprise Replication on a single database server, highlight the database server and choose **Server→Start Replication**. When you start replication for a database server, the global catalog tables are reread and all necessary ER threads restarted. If the database server from which you are starting replication is the primary database server, ER restarts the evaluation of the data marked for replication in the logical log. Database server connections are either accepted or created, as required, based on information in the global catalog. The start command is only effective after a **Stop Replication** command is issued.

Removing a Database Server from Enterprise Replication

Important: *If you remove a database server before the queues are properly emptied, all data was not delivered for the database server might also be deleted.*

To remove a database server from Enterprise Replication, select the server and then choose **Server→Remove Server**. The RM asks you to confirm that you want to remove the database server, as in Figure 8-37.

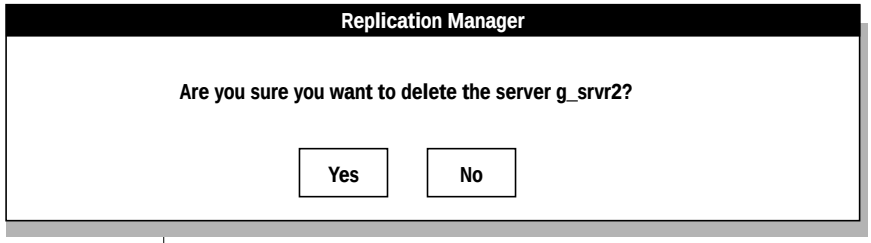


Figure 8-37
Confirm Server Deletion

If you confirm that you want to delete the database server, the RM removes all information about the database server from all ER catalogs. This process can take a few minutes. After completing the deletion, RM asks you to connect to another database server.

Tip: *You need to remove the database server twice to remove all links to replicates and replicate groups in the global catalog.*

Reloading the Global Catalog

To reload the global catalog for a database server, select the database server and choose **Server→Reload from Global Catalog**. Reloading the global catalog ensures that the RM is using up-to-date information about the database server.

Viewing Database Server Properties

To view database server properties, choose **Server→Server Properties...**. **Server Properties...** accesses the Server Properties dialog box.

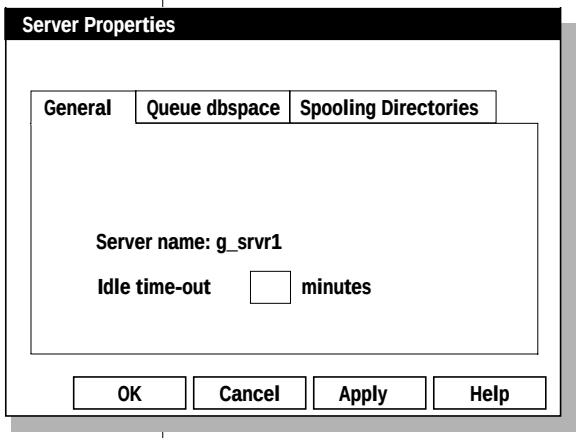


Figure 8-38
Server Properties Dialog Box

The Server Properties dialog box in Figure 8-38 summarizes the properties you assigned to a server.

The **Queue dbspace** tab lets you modify the location of the Receive queue. You cannot change the location of the Send queue.

The **Spooling Directories** tab lets you change the locations of the ATS and RIS directories.

Monitoring the Server

For information on the monitoring capabilities of the Replication Manager, see Chapter 9, "Monitoring Enterprise Replication."

View Menu

The **View** menu in Figure 8-39 lets you choose how information is displayed.

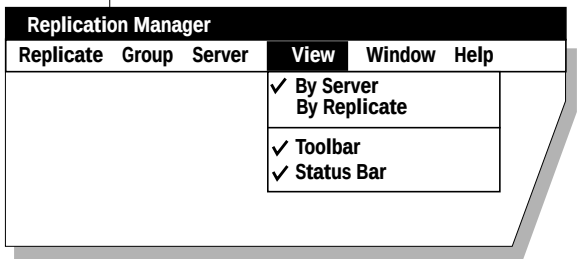


Figure 8-39
View Menu

Use the **View** menu to view information by database server or by replicate. These menu items produce the same results as the replicate and database server icon. You can also choose to view (or hide) the toolbar and the status bar.

The **Toolbar** and **Status Bar** menu items cause the Toolbar and Status Bar to appear or disappear from the display.

Window Menu

The **Window** menu lets you arrange the windows on your display, change Replication Manager modes, and activate different windows.

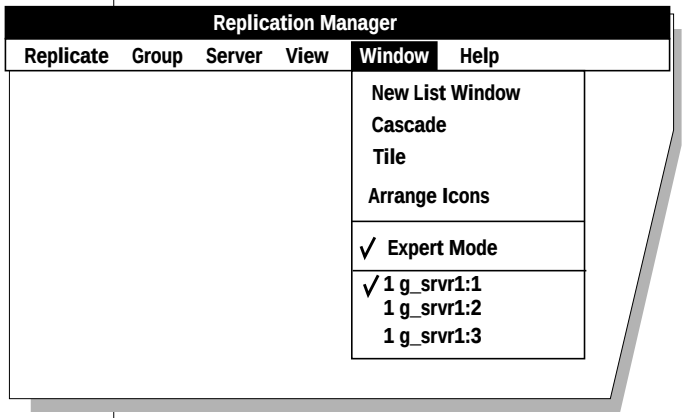


Figure 8-40
Window Menu

Managing the Window Display

Use the **Window** menu in Figure 8-40 to arrange your windows on your workstation. The following options appear on the **Window** menu.

Option	Function
New List Window	Creates a new window. This option is helpful if you want to view different information simultaneously. For example, you can show database server states in one window, and replicate states in another window.
Cascade	Cascades the windows (overlaps windows on top of each other in a hierarchical order).

(1 of 2)

Option	Function
Tile	Tiles the windows (places them next to each other).
Arrange Icons	Arranges the icons.
Database server name:*number*	Activates the selected window. Each time you create a new list window, the RM creates a new window and adds an entry to the numbered list at the bottom of the **Window** menu. To activate a specific window, choose its entry from the numbered list.

(2 of 2)

Selecting Non-Expert Mode

When you are in expert mode the **Window** menu shows a check on the **Expert Mode** item. Choose **Window→Expert Mode** to change the mode back to non-expert. Non-expert mode is described in Chapter 7, "Administering Enterprise Replication."

File Menu

The New Script and Open Script choices from the **File** menu are documented in the next section,"Using the Scripting View" on page 8-51. The choice **File→Exit** exits from the Replication Manager. The choice **File→Event Monitor** displays the Replication Event Monitor.

Monitoring Replication Events

The Replication Event Monitor notifies you when a warning or severe error occurs on a *remote* database server that is participating in Enterprise Replication. ER does not send error messages from the local database server to the Replication Event Monitor.

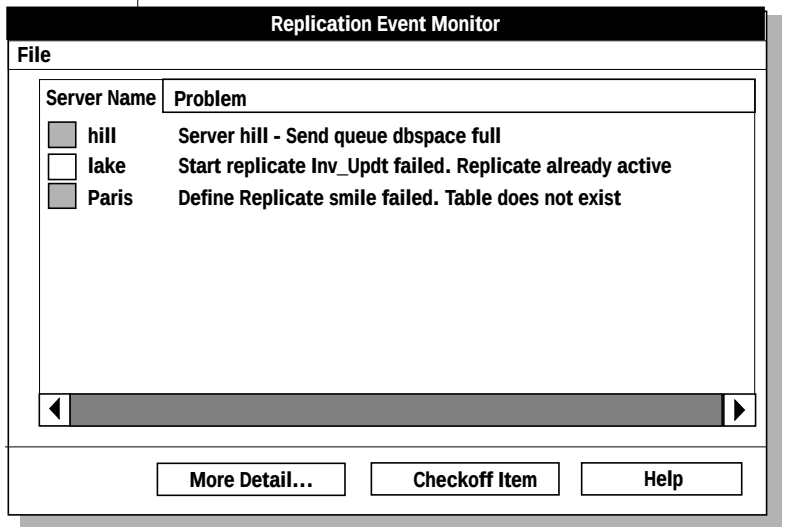


Figure 8-41
Replication Event Monitor

In Figure 8-41, database servers that are experiencing replication errors are listed in the Replication Event Monitor. The shaded box (this box is red on a Windows NT workstation monitor) indicates a severe error. The white box (this box is yellow on a Windows NT workstation monitor) indicates a warning. For more information about an error, select the database server name and then click **More Detail...**.

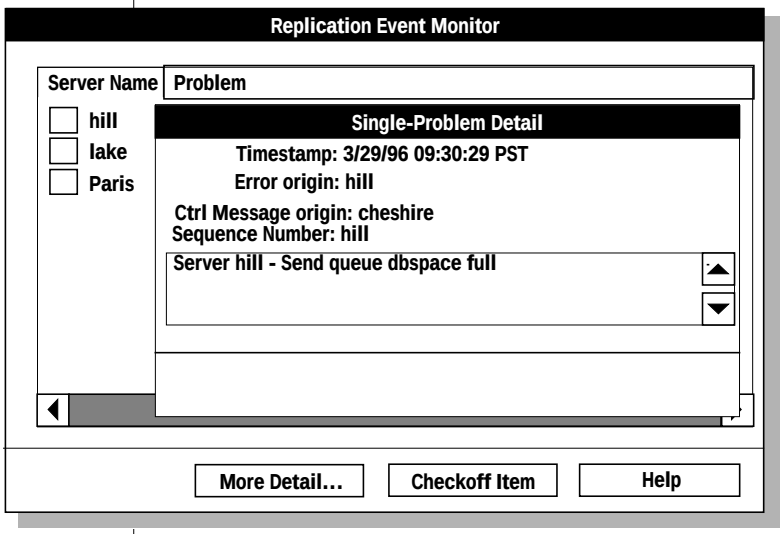


Figure 8-42
Single-Problem Checklist

In Figure 8-42, information is provided about database server **hill**. When you have reviewed this information, click **Close** to return to the Replication Event Monitor.

When you return to the main dialog box of the Replication Event Monitor, you can select database server **hill** and then click **Checkoff Item.** This option marks the error and indicates it has been reviewed. From the **File** menu of the Replication Event Monitor you can also delete selected events or export the information to a text file.

Using the Scripting View

The Scripting view allows you to create and save replication systems. To use the Scripting view, choose **File→New Script** or **File→Open Script**.

When you open the Scripting View, the title bar of the Replication Manager changes. The menu bar has the following menus:

- File
- Object
- Edit
- View
- Window
- Help

The File Menu in Scripting View

The **File** menu in Figure 8-43 lets you manipulate replication scripts. You can open, close, save, and print from this menu.

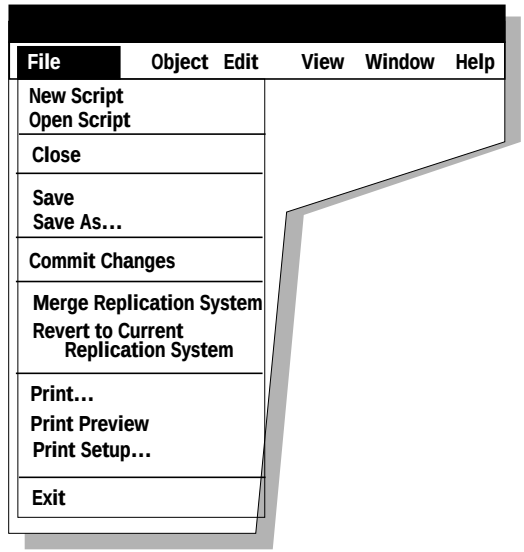


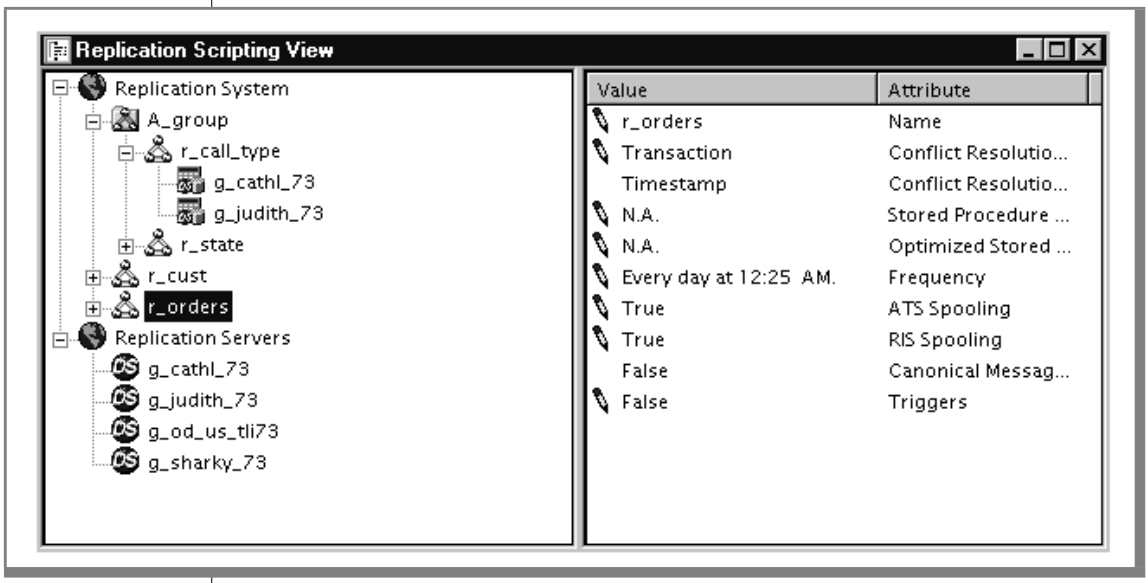
Figure 8-43 *File Menu in Scripting View*

New Script

When you choose **File→New Script**, the RM asks if you want to make a copy of your current replication system.

If you respond **Yes**, the RM copies your current replication system into the Scripting View. Figure 8-44 shows an example of a Scripting View. The Scripting View has two parts, the Replication System and the Replication Servers. The Replication Servers part of the display shows all of the servers that are declared to Enterprise Replication.

Figure 8-44
Create New Script from Current Replication System



If you respond **No**, the Scripting View displays the servers, but the Replication System is empty, because you did not define any replicates.

Open Script

You can open a script that you already prepared. After you open the script, you can add new objects to the script or merge the current replication system into the script.

Save and Save As

The **Save** and **Save As** menu items let you save a copy of the replication script as a text file. "Example of a Replication Script" on page 8-58 shows sample replication script.

Commit Changes

When you choose **Commit Changes**, the RM asks how you want these changes applied, as in Figure 8-45.

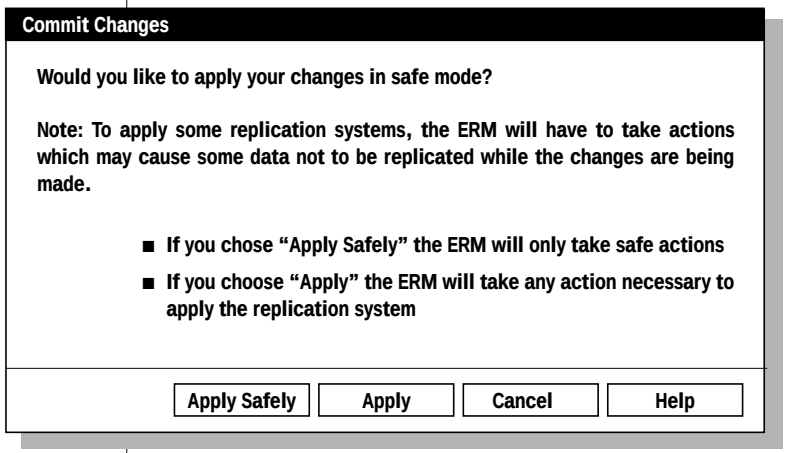


Figure 8-45
Commit Changes Dialog Box

If you choose **Apply Safely** or **Apply**, the RM displays an Applier Summary that lists the proposed changes and gives you another chance to **Apply** or **Cancel**.

If you again choose **Apply**, the RM might take a few minutes to make all of the changes. The RM shows a status box to report on its progress making the changes.

Merge Replication Script

The **Merge Replication Script** option merges the current replication system into your current scripting view.

Revert to the Current Replication System

The **Revert** option lets you cancel any changes that you made from the scripting view.

The Object Menu in Scripting View

The **Add** option of the **Object** menu lets you create a new replication object.

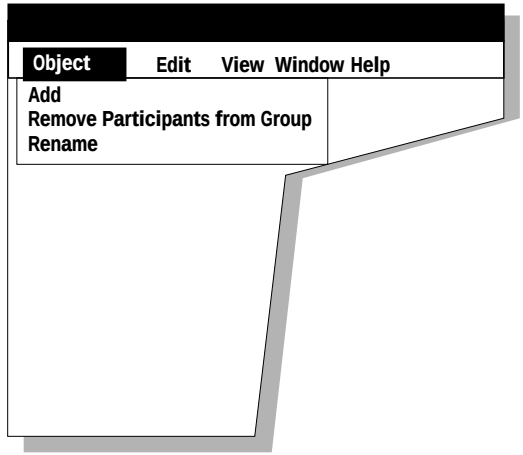


Figure 8-46
Object Menu in Scripting View

Add

When you add a new object, the Scripting View assigns default attributes to the new object and displays those attributes in the right-hand pane of the view. To modify or assign a value to an attribute, select the attribute. Then click the left button to type in a value or click the right button for a list of values or to enter a browser.

Remove Participants from Group

You can remove participants from a group that you created with the Scripting View.

The Edit Menu in Scripting View

The **Edit** menu lets you copy and paste replication objects so that you can create new objects in the replication script.

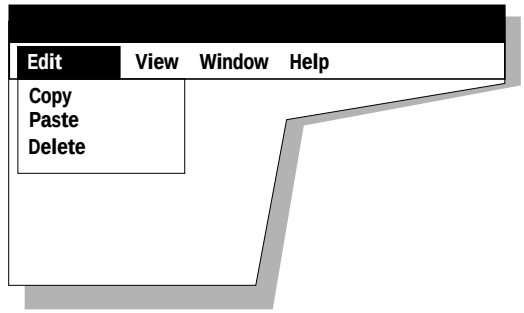


Figure 8-47 *Edit Menu in Scripting View*

For example, select the **r_orders** replicate in Figure 8-44 on page 8-52 and choose **Copy** and then **Paste**. The Scripting View creates a new replicate named **r_orders1** that has the same attributes as **r_orders**. You can now modify **r_orders1** to create a unique replicate by changing its attributes or adding (or deleting) participants.

The View Menu in Scripting View

The **View** menu allows you to control your display.

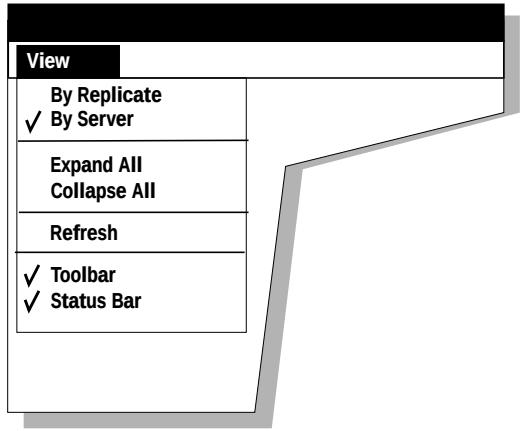


Figure 8-48 *View Menu in Scripting View*

The default view is **By Server**. When you choose **By Replicate**, the display changes so that the replicates and groups are at the top of the left-hand panel, instead of the servers.

The Window Menu in Scripting View

The **Window** menu allows you to arrange the sub-windows on your screen. The checkmark in the lower list indicates which window is active.

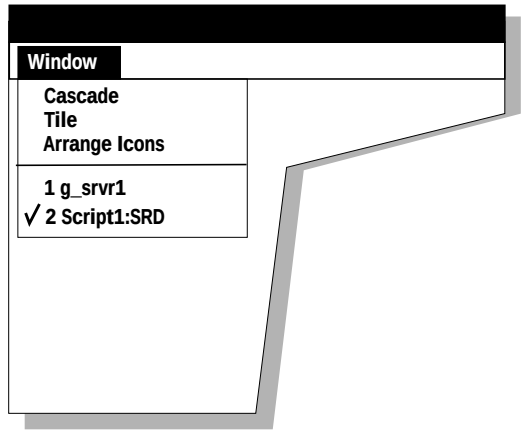


Figure 8-49 *Window Menu in Scripting View*

The Help Menu in Scripting View

The **Help** menu accesses on-line help and displays information about the product.

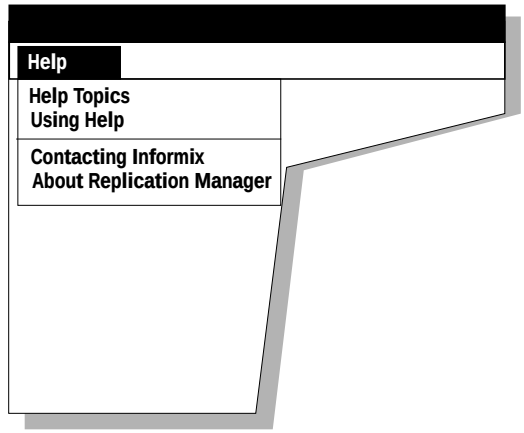


Figure 8-50 *Help Menu in Scripting View*

Example of a Replication Script

The following text file is the result of saving a replication script:

```
Scripting View Archive

************ Definition of Servers **************

ObjectType:                  Server
Name:                        g_sharky_73
ServerType:                  ROOT_SERVER
Sync/ParentServer:           -
ATSDirectory:                /tmp
RISDirectory:                /tmp
SendQueueDBSpace:            -
ReceiveQueueDBSpace:         -
IdleTimeOut(mins):           0

ObjectType:                  Server
Name:                        g_cathl_73
ServerType:                  ROOT_SERVER
Sync/ParentServer:           -
ATSDirectory:                /tmp
RISDirectory:                /tmp
SendQueueDBSpace:            -
ReceiveQueueDBSpace:         -
IdleTimeOut(mins):           0

ObjectType:                  Server
Name:                        g_od_us_tli73
ServerType:                  NONROOT_SERVER
Sync/ParentServer:           g_judith_73
ATSDirectory:                /tmp
RISDirectory:                /tmp
SendQueueDBSpace:            -
ReceiveQueueDBSpace:         -
IdleTimeOut(mins):           0

ObjectType:                  Server
Name:                        g_judith_73
ServerType:                  ROOT_SERVER
Sync/ParentServer:           -
ATSDirectory:                /tmp
RISDirectory:                /tmp
SendQueueDBSpace:            -
ReceiveQueueDBSpace:         -
IdleTimeOut(mins):           0

************ Definition of Groups (Replicates & Participants) **************

ObjectType:                  Group
Name:                        A_group
```

```
Frequency
    Type:                   FREQ_TIME_CHAR
    Hour:                   7
    Minute:                 36
    Day:                    76
    Month:                  0
Sequential:                 TRUE

        ObjectType:                 Replicate
        Name:                       r_state
        ConflictResolutionScope:    TRANSACTION
        ConflictResolutionType:     IGNORE
        StoredProcedureName:        -
        Frequency
            Type:                   FREQ_IMMED_CHAR
            Hour:                   0
            Minute:                 0
            Day:                    0
            Month:                  0
        ATSSpooling:                FALSE
        RISSpooling:                FALSE
        CanonicalMessageFormat:     FALSE
        FireTriggers:               FALSE
        GroupName:                  A_group

                ObjectType:                 Participant
                Name:                       g_cathl_73
                DataBase:                   odin7
                Owner:                      informix
                Table:                      state
                Primary/Target:             Primary
                SelectStatement:            Select * from informix.state
                ReplicateName:              r_state

                ObjectType:                 Participant
                Name:                       g_judith_73
                DataBase:                   smoke7
                Owner:                      informix
                Table:                      state
                Primary/Target:             Primary
                SelectStatement:            Select * from informix.state
                ReplicateName:              r_state

        ObjectType:                 Replicate
        Name:                       r_call_type
        ConflictResolutionScope:    TRANSACTION
        ConflictResolutionType:     IGNORE
        StoredProcedureName:        -
        Frequency
            Type:                   FREQ_IMMED_CHAR
            Hour:                   0
            Minute:                 0
            Day:                    0
            Month:                  0
```

```
            ATSSpooling:                FALSE
            RISSpooling:                FALSE
            CanonicalMessageFormat:     FALSE
            FireTriggers:               FALSE
            GroupName:                  A_group

                    ObjectType:                 Participant
                    Name:                       g_cathl_73
                    DataBase:                   odin7
                    Owner:                      informix
                    Table:                      call_type
                    Primary/Target:             Primary
                    SelectStatement:            Select * from informix.call_type

                    ReplicateName:              r_call_type

                    ObjectType:                 Participant
                    Name:                       g_judith_73
                    DataBase:                   smoke7
                    Owner:                      informix
                    Table:                      call_type
                    Primary/Target:             Primary
                    SelectStatement:            Select * from informix.call_type

                    ReplicateName:              r_call_type

***********  Definition of Replicates (Participants) **************

ObjectType:                 Replicate
Name:                       r_orders
ConflictResolutionScope:    TRANSACTION
ConflictResolutionType:     TIMESTAMP
StoredProcedureName:        -
Frequency
    Type:                   FREQ_TIME_CHAR
    Hour:                   0
    Minute:                 25
    Day:                    76
    Month:                  0
ATSSpooling:                TRUE
RISSpooling:                TRUE
CanonicalMessageFormat:     FALSE
FireTriggers:               FALSE
GroupName:                  -

            ObjectType:                 Participant
            Name:                       g_sharky_73
            DataBase:                   shark7
            Owner:                      informix
            Table:                      orders
            Primary/Target:             Primary
            SelectStatement:            Select * from informix.orders
            ReplicateName:              r_orders

            ObjectType:                 Participant
```

```
        Name:                       g_judith_73
        DataBase:                   smoke7
        Owner:                      informix
        Table:                      orders
        Primary/Target:             Primary
        SelectStatement:            Select * from informix.orders
        ReplicateName:              r_orders

ObjectType:                 Replicate
Name:                       r_cust
ConflictResolutionScope:    ROWBYROW
ConflictResolutionType:     IGNORE
StoredProcedureName:        -
Frequency
    Type:                   FREQ_IMMED_CHAR
    Hour:                   0
    Minute:                 0
    Day:                    0
    Month:                  0
ATSSpooling:                TRUE
RISSpooling:                TRUE
CanonicalMessageFormat:     FALSE
FireTriggers:               FALSE
GroupName:                  -

        ObjectType:                 Participant
        Name:                       g_sharky_73
        DataBase:                   shark7
        Owner:                      informix
        Table:                      customer
        Primary/Target:             Primary
        SelectStatement:            Select * from informix.customer
        ReplicateName:              r_cust

        ObjectType:                 Participant
        Name:                       g_cathl_73
        DataBase:                   odin7
        Owner:                      informix
        Table:                      customer
        Primary/Target:             Primary
        SelectStatement:            Select * from informix.customer
        ReplicateName:              r_cust

        ObjectType:                 Participant
        Name:                       g_judith_73
        DataBase:                   smoke7
        Owner:                      informix
        Table:                      customer
        Primary/Target:             Primary
        SelectStatement:            Select * from informix.customer
        ReplicateName:              r_cust
```

Chapter

9

Monitoring Enterprise Replication

The Enterprise Replication Monitor (ERM) helps you to monitor the replication activity of any database server declared to Enterprise Replication. This chapter explains how to monitor statistics and how to use those statistics effectively. The ERM statistics include information about:

- ER send queues.
- commit times between database servers.
- database server connections.
- ER transactions.

The ERM allows you to print bar, line, and dynamic text graphs, as well as export data in a tab-delimited text format to spreadsheet applications.

Preparing to Use Monitoring

The ERM uses SNMP subagents that are part of your Informix product. If the SNMP subagents are not already active, you must start them before you can use the monitoring tools.

UNIX

Starting Subagents on UNIX

On most UNIX workstations, you can use the **ps** command to check whether an **onsnmp** process is running. Take these steps to start the SNMP subagent:

1. Log in as user **root**.
2. Set the **INFORMIXDIR** environment variable.
3. Set the **SR_AGT_CONF_DIR** environment variable to **$INFORMIXDIR/snmpsnmpr**.
4. Execute **$INFORMIXDIR/bin/snmpdm**.

5. Log in as user **informix**.
6. Set the **INFORMIXDIR** and **SR_AGT_CONF_DIR** environment variables.
7. Execute **$INFORMIXDIR/bin/onservapd**.

For more information about the SNMP subagent, refer to the *Informix SNMP Subagent Guide*.

Monitoring Enterprise Replication Servers

To open the ERM, choose **Server → Monitor Server** from the Replication Manager main window. The ERM opens a window that displays all of the servers declared to Enterprise Replication, as shown in Figure 9-1.

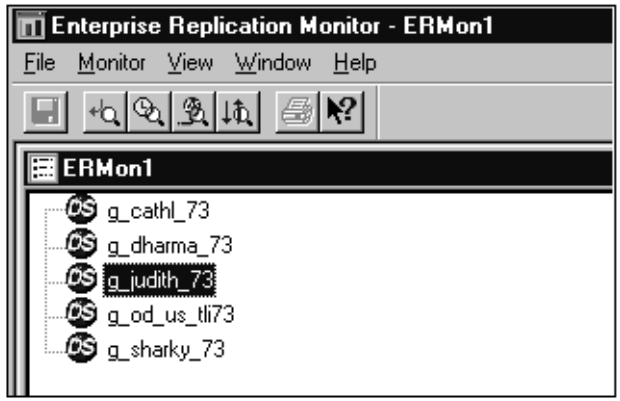


Figure 9-1
Main Window of Enterprise Replication Monitor

The **View**, **Window**, and **Help** menus are analogous to the same menus on the Replication Manager main window. The **File** menu allows you to reload the list of servers, format a print, and exit. For information about the print options, refer to "Printing Graphs" on page 9-15.

The **Monitor** menu has the following menu items:

- Monitor SendQ...
- Monitor Commit Time...
- Monitor Connections...
- Monitor Transactions...

The ERM monitors database server functions on a per-server basis. Therefore, you must select a database server before you begin to monitor data replication activities.

Customizing the Monitor Displays

When you select an item from the **Monitor** menu, the ERM displays tabbed pages that allow you to design the monitoring display. Figure 9-2 on page 9-6 shows the **General** page. This page specifies:

- Type of graph
- Range

  The range is the amount of data you want to display in a graph.
- Domain

  The domain is the level of detail for the data you are monitoring.

Defining the Graph Type

The Replication Manager supports three types of dynamic graphs: bar, line, and dynamic text, as shown in Figure 9-2.

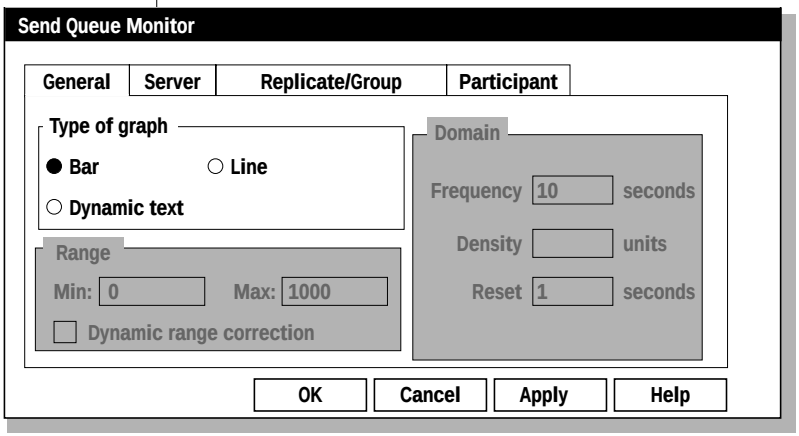


Figure 9-2
Send Queue Monitor General Page: Type of Graph

Defining the Range

The range defines the amount of data displayed on the *y*-axis of a bar or line graph. The default range is 0 through 1000, as shown in Figure 9-3 on page 9-7. If you activate **Dynamic range correction**, the display automatically adjusts the range in the graph to accommodate the data. For example, if you select a range of 0 through 1000 and your data is in a range of 500 to 1500, the graph starts displaying data at 500 on the *y*-axis.

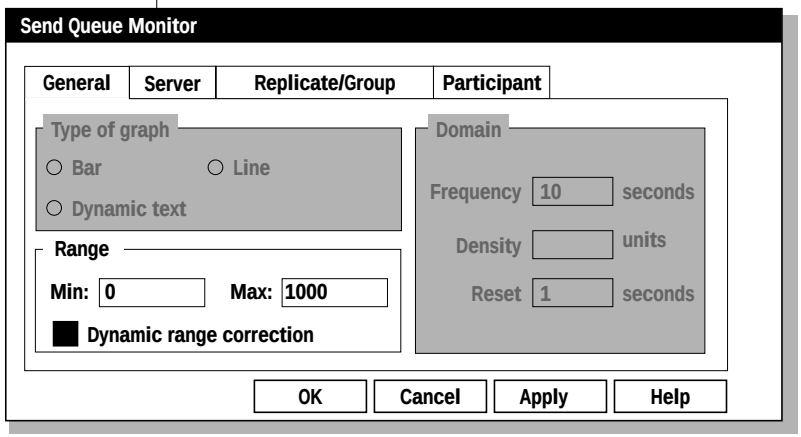


Figure 9-3
Send Queue Monitor General Page: Range

Defining the Domain

You can select three options to define the detail of data in your graphs, as shown in Figure 9-4.

- Frequency
- Density
- Reset

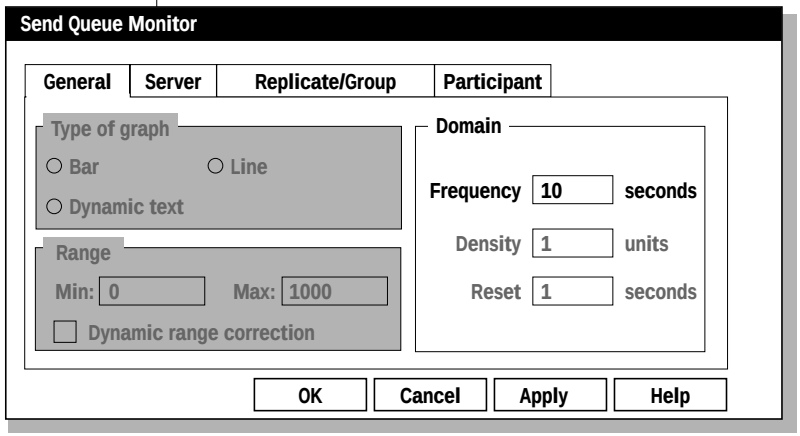


Figure 9-4
Send Queue Monitor General Page: Domain

Frequency

The **Frequency** option directly affects the data in your graph. The time you specify in seconds determines how often the ERM queries data and refreshes the graph you are viewing. The default is 10 seconds.

Density

The **Density** option multiplied by the **Frequency** option defines how much data is displayed on the *x*-axis. For example, if you define a frequency of 10 seconds and a density of 10 units, your graph will display 100 seconds on the *x*-axis, in which 10 units of data are displayed across the 100 seconds.

Reset

The **Reset** value resets the data displayed in your graph. All ERM graphs are dynamic and are refreshed based on the frequency you choose, moving the oldest displayed data off the graph after the number of seconds you specify. For example, if you are viewing a line graph and the reset option is set to 10 seconds, the last 10 seconds of the graph (which is displayed on the left-hand side of the graph) is rolled off and the newest 10 seconds of data is displayed on the right-hand side of the graph.

Monitoring the Send Queue

The **Monitor → Commit Time...** option generates send-queue statistics that are useful when you are generating replicated transactions. In a primary-target replication system, monitor the statistics from the sending database server. In an update-anywhere replication system, monitor both the sending and receiving database servers.

First, select the characteristics for your graphs with the general property sheet, and then select what you want to monitor.

The ERM helps you monitor send-queue data for three different entities:

- Database servers
- Replicates and / or groups
- Participants

Selecting Servers from the Send-Queue Monitor

In Figure 9-5, server **cheshire** is selected. The ERM asks you to select the database servers that are waiting for data from the Send queue on server **cheshire**.

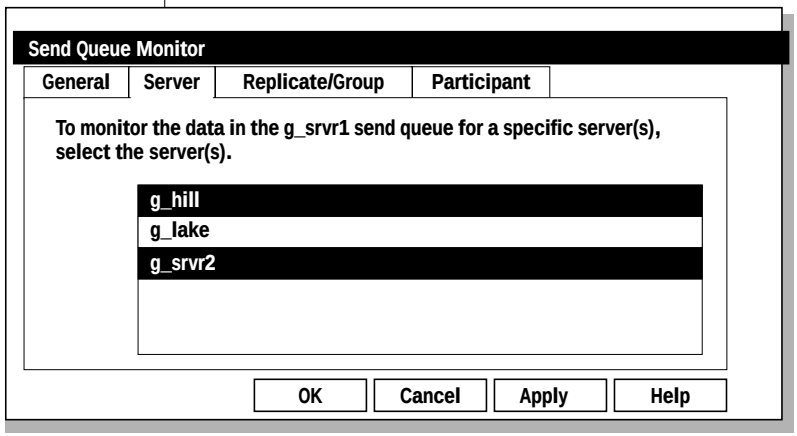


Figure 9-5
Send Queue Monitor: Server Page

Selecting Replicates and/or Groups from the Send Queue Monitor

In Figure 9-6 on page 9-10 server **g_srvr1** is selected. The ERM asks you to select the replicate(s) and / or groups that are waiting for data from the Send queue on **g_srvr1**.

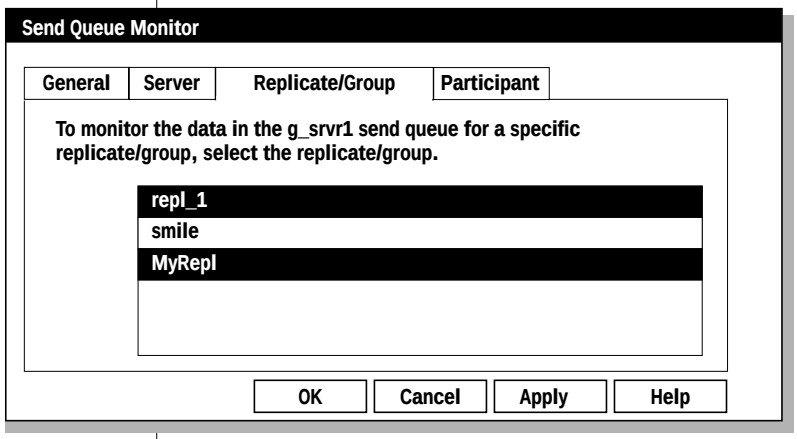


Figure 9-6
Send Queue Monitor: Replicate/Group Page

Selecting Participant(s) from the Send Queue Monitor

In Figure 9-7 server **g_srvr1** is selected. The ERM asks you to select the participants that are waiting for data from the send queue on server **g_srvr1**.

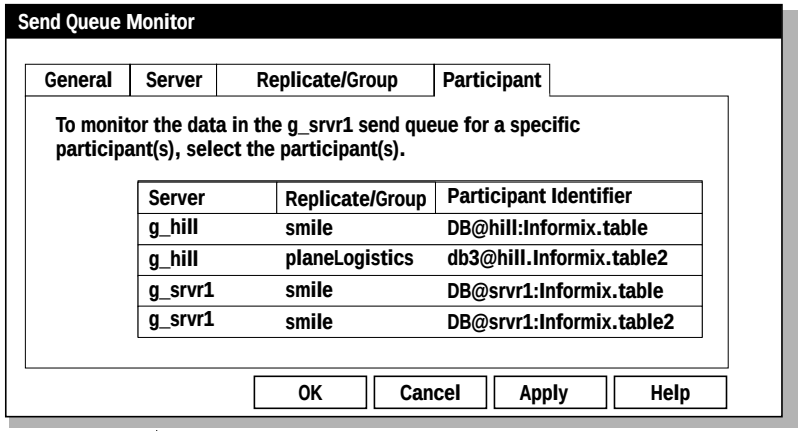


Figure 9-7
Send Queue Monitor: Participant Page

Monitoring Commit Times

The **Monitor → Commit Time...** option monitors how long it takes Enterprise Replication to process a transaction and how long it takes to send the transaction from a source database server to a target database server.

Commit-time statistics are useful when you are generating replicated transactions. In a primary-target replication system, monitor the statistics from the sending database server. In an update-anywhere replication system, monitor both the sending and receiving database servers.

The **Statistics** page in Figure 9-8 lets you choose which replicates to monitor and what information to display.

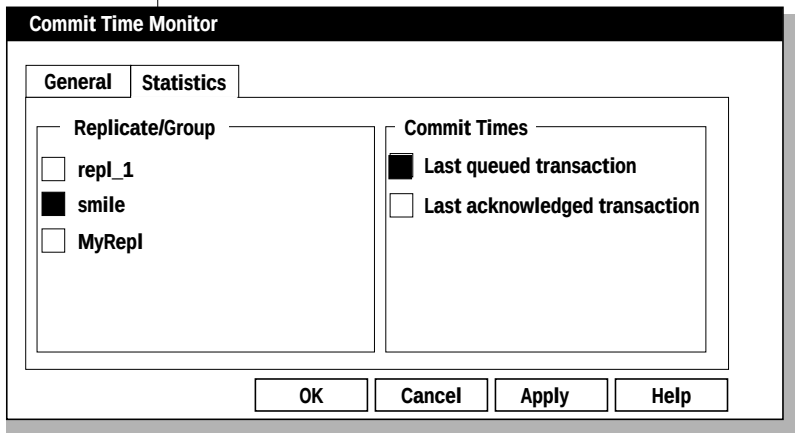


Figure 9-8
Commit Time Monitor: Statistics Page

The ERM lists all of the replicates and / or groups in which the database server is participating. The commit times shown in the **Statistics** folder represent commit times on the database server you are currently monitoring that is participating in the replicates and / or groups. The **Commit Times** options give the following information:

Commit Time	Function
Last-queued transaction	Provides the number of seconds since the last-queued transaction
Last-acknowledged transaction	Transaction provides the time (in seconds) of the last-acknowledged transaction

Monitoring Connections

The **Monitor → Commit Time...** option monitors the number of messages and bytes received and sent between database servers. Figure 9-9 shows the Connection Monitor dialog box.

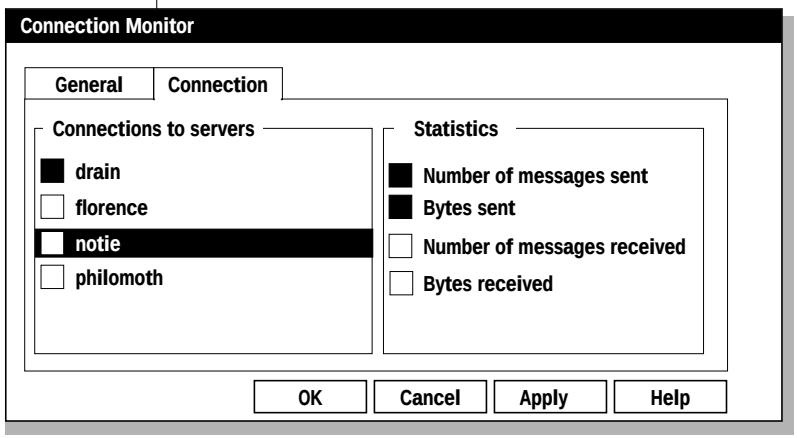


Figure 9-9
Connection Monitor: Connection Page

Connection statistics provide information about how much data is being received and sent from each database server. Use the **Connection Monitor** to monitor the following statistics:

Statistic	Description
Number of messages sent	The number of messages sent from a database server to selected database servers.
Bytes sent	The number of bytes sent from a database server to selected database servers.
Number of messages received	The number of messages received from a database server to selected database servers.
Bytes received	The number of bytes received from a database server to selected database servers.

Monitoring Transactions

The **Monitor → Commit Time...** option monitors transaction latency and committed and aborted transactions and rows. Figure 9-10 on page 9-14 shows the **Monitor Transaction...** dialog box.

In a primary-target replication system, monitor the statistics at the receiving database server. In an update-anywhere replication system, monitor both the sending and receiving database servers.

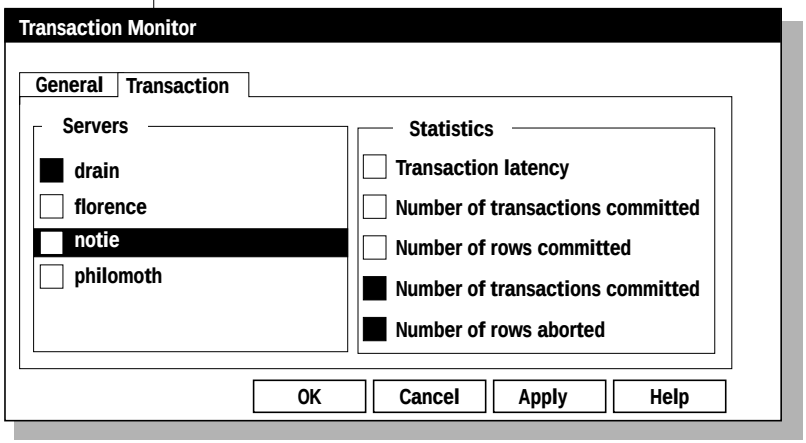


Figure 9-10
Transaction Monitor: Transaction Page

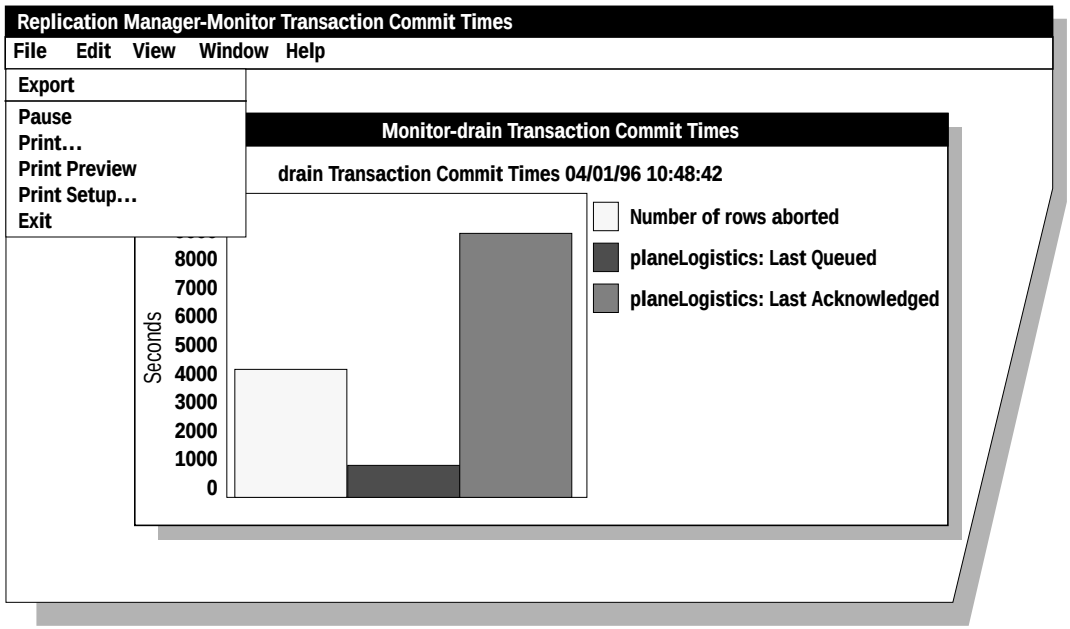
The statistics options of the Transaction Monitor provide the following information.

Statistic	Description
Transaction latency	The amount of time of transaction latency. (Transaction latency is the time that passes between when a transaction commits on the source database server and when a transaction commits on the target database server.)
Number of transactions committed	The number of transactions committed on database server(s).
Number of transactions aborted	The number of transactions aborted on database server(s).
Number of rows committed	The number of rows committed on database server(s).
Number of rows aborted	The number of rows aborted on database server(s).

Printing Graphs

When you view a graph, the **File** menu changes to include export and printing options, as shown in Figure 9-11.

Figure 9-11
Replication Manager Printing Option



Export

The **Export** menu item allows you to export data in a tab-delimited format.

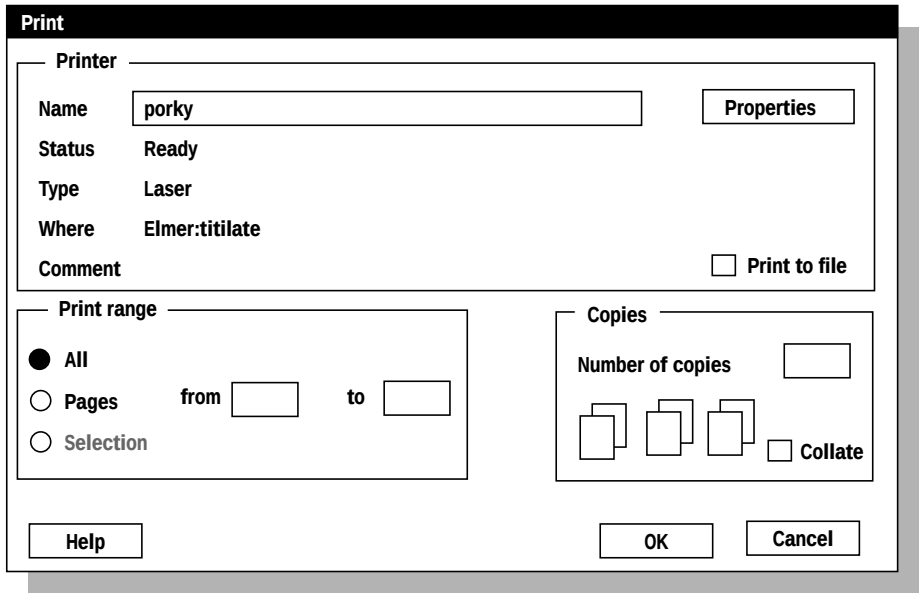
Pause

The **Pause** menu item pauses the graph and does not allow the graph to be refreshed at the frequency you selected. This option can be helpful if you have a low frequency set and the graph refreshes frequently. To get a hardcopy of the exact graph you are viewing, choose the **Pause** menu item and then choose **Print**.

Print

The **Print** menu item accesses the Print dialog box in Figure 9-12. If the information in the Print dialog box is incorrect, you can make changes with the **Print Setup** menu item.

Figure 9-12
Replication Manager Print Dialog Box

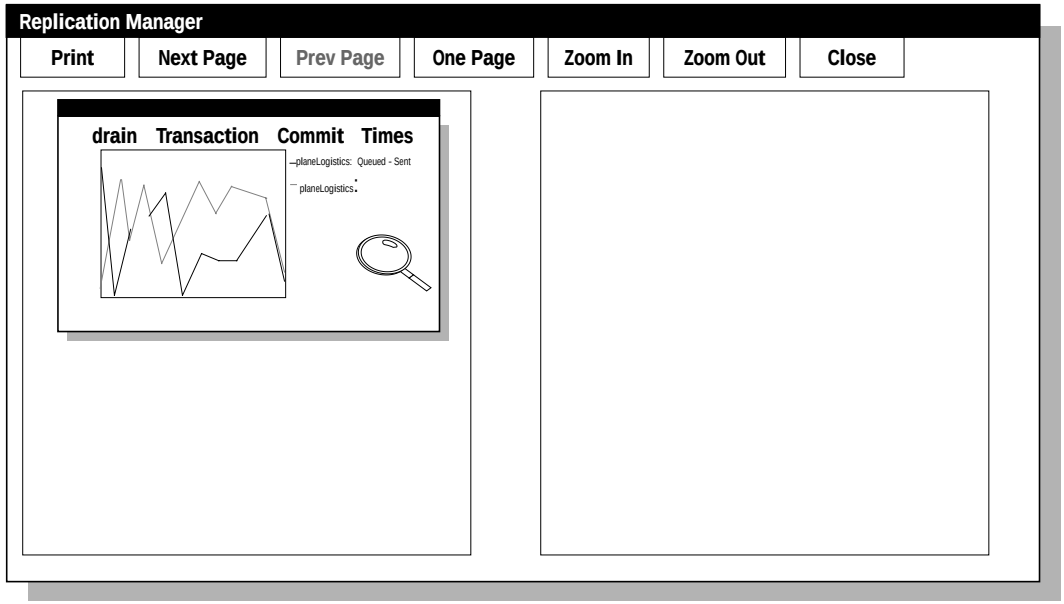


Print Preview

The **Print Preview** menu item allows you to preview your graph before you print it. The Print Preview dialog box shown in Figure 9-13 allows you to enlarge the graph display. You can use the **Zoom In** or **Zoom Out** buttons, or double-click on the magnifying glass.

Tip: *The Replication Manager only allows you to print one graph at a time.*

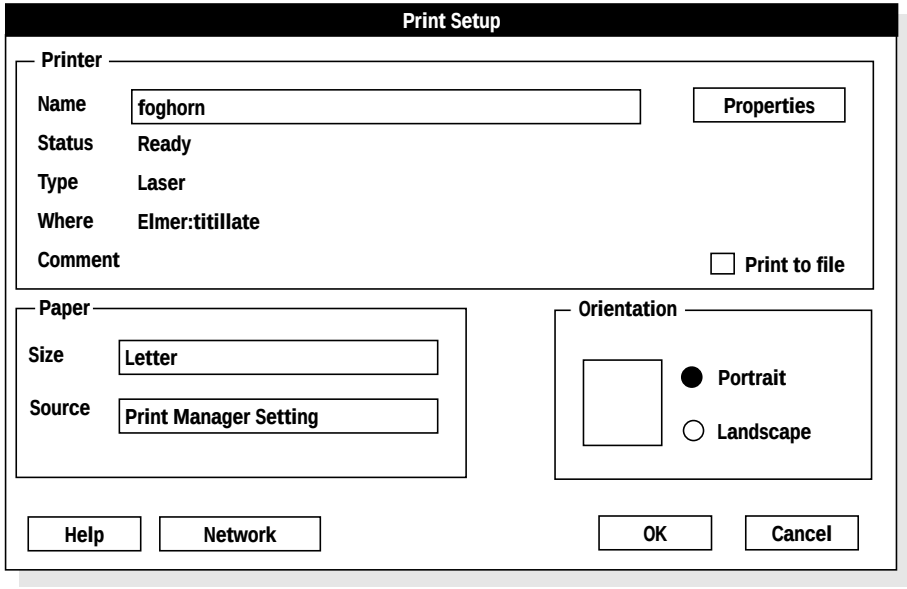
Figure 9-13
Print Preview Dialog Box



Print Setup

The **Print Setup** dialog box allows you to choose printers and various printer options, as shown in Figure 9-14.

Figure 9-14
Replication Manager Print Setup Dialog Box



Chapter

10

Diagnosing Enterprise Replication

Enterprise Replication provides tools to help you diagnose problems that arise during replications. The aborted-transaction spooling (ATS) and row-information spooling (RIS) files contain information about failed transactions. The Replication Manager logs events in the Replication Event Monitor. This chapter describes the ATS and RIS files and provides a list of Replication Event Monitor log events.

Tip: *Some of the features described in this chapter are not available in the current release. If you try to use an option that is not available, the server responds with the message, "Feature not supported."*

Aborted-Transaction Spooling

When ATS is active, all failed replication transactions are recorded in ATS spool files. Each file produced by ATS contains all the information pertinent to a single failed transaction. If a replicated transaction fails for any reason (constraint violation, duplicate, and so on), all of the buffers in the replication message that compose the transaction are written to a local operating-system file. You can use the ATS files to identify problems, or as input to custom-written utilities that extract or reapply the aborted rows.

Preparing to Use ATS

Failed transactions are not automatically recorded in ATS files. To collect ATS information, you must do the following:

1. Specify a directory where ATS files will be stored when you define a server for ER.
   - With the Enterprise Replication Manager (RM), set the spooling directories on the Declare Server display in Figure 8-2 on page 8-8.
   - With the APIs, set the **charats_dir** attribute in **cdr_define_serv()**.
   - With the CLU, include the **--ats** option with **cdr define server**.
2. Specify that ATS should be active when you define a replicate. In expert mode, you can use ATS for some replicates and not for others.
   - With the Enterprise Replication Manager (RM), activate ATS spooling on the replicate options page, "Defining Participant Attributes" on page 8-19.
   - With the APIs, set the **charats_dir** attribute in **cdr_define_repl()**.
   - With the CLU, include the **--ats** option with **cdr define replicate**.

Naming of ATS Spool File

When ATS is active, each aborted transaction is written to a file in the directory specified in the server definition. The following table provides the naming convention for ATS spool files:

```
ats.target.source.threadId.timestamp.sequence
```

Name	Description
target	The name of the database server receiving this replicate transaction.
source	The name of the database server that originated the transaction.
threadId	The identifier of the thread which processed this transaction.
timestamp	The value of the internal time stamp at the time that this ATS instance was started.
sequence	A unique integer. This integer is incremented by ATS each time a transaction is spooled.

The naming convention ensures that all filenames generated by ATS are unique, and therefore, name collision is unlikely. However, when ATS opens a file for writing, any previous file contents will be over-written. (ATS does not append to a spool file; if a name collision does occur with an existing file, its original contents will be lost.)

The first three characters in each line of the ATS spool file describe the type of information for the line, as defined in the following table.

Label	Name	Description
TXH	Transaction Heading	Information from the transaction header that includes the sending server ID and the commit time, the receiving server ID and the received time, and any Enterprise Replication, SQL, or ISAM error information for the transaction.
RRH	Replicated Row Heading	Header information from the replicated rows that includes the row number within the transaction, the group ID, the replicate ID (same as replicate group ID if replicate is not part of any replicate group), the database, owner, and table name, and the database operation.
RRS	Replicated-Row Shadow Columns	Shadow column information from replicated rows that includes the source server ID and the time when the row is updated on the source server. This line is printed only if the replicate is defined with a conflict-resolution rule.
RRD	Replicated-Row Data	Replicated row data is the list of replicated columns in the same order as in the SELECT statement in the define replicate command. Each column is separated by a '|' and displayed in ASCII format. When the spooling program encounters severe errors (for example, cannot retrieve replicate id for the replicated row, unable to determine the replicated column's type, size, or length), it displays this row data in hexadecimal format.

Aborted-transaction spooling only occurs if the entire transaction is aborted. Transactions defined with row scope that have aborted rows but are successfully committed on the target tables are not logged.

The following output is an example of an ATS file for a transaction sent by server **sls4** to server **pur1**. The file is named:

```
ats.pur1.sls4.D_2.960529_23:27:16.6spooled
```

The file contents are:

```
TXH RIS file:/tmp/p1s1/ris.pur1.sls4.D_2.960529_23:27:16.5 has also been
created for this transaction
==========
TXH Source ID:390 / Name:sls4 / CommitTime:96-05-29 23:26:24
TXH Target ID:392 / Name:pur1 / ReceiveTime:96-05-29 23:27:16
TXH Number of rows processed when transaction was aborted:4
TXH All rows in a transaction defined with row scope were rejected
----------
RRH Row:1 / Group Id: 24707077 / Replicate Id: 24707077 / Table: stores@cdr.item
/ DbOp:Update
RRS 390(sls4)|833437578(96/05/29 23:26:18)
RRD 20|VGA Card 439|A|Acer|Chevrolet|VGA Card|439|422.00|538.00|738.00|13
.00|12.00|8.00|Normal Demand Moderate Value Item
----------
RRH Row:2 / Group Id: 24707077 / Replicate Id: 24707077 / Table: stores@cdr.item
/ DbOp:Update
RRS 390(sls4)|833437580(96/05/29 23:26:20)
RRD 10|Digitizer 319|A|Apple|Toyota|Digitizer|319|29.00|36.00|581.00|28.0
0|2.00|2.00|Normal Demand Moderate Value Item
----------
RRH Row:3 / Group Id: 24707080 / Replicate Id: 24707080 / Table:
stores@cdr.customer / DbOp:Update
RRS 390(sls4)|833437582(96/05/29 23:26:22)
RRD 18|Larraine Hustin|2021 South Blvd.|A|8|(399)-2423745|0.00|0.00|0.00|
0.00|83429.00|Untouched|This Customer belongs to the Nation of Kamas
----------
RRH Row:4 / Group Id: 24707077 / Replicate Id: 24707077 / Table: stores@cdr.item
/ DbOp:Update
RRS 390(sls4)|833437583(96/05/29 23:26:23)
RRD 9|Keyboard173|A|ALR|Datsun|Kboard|173|978.00|1399.00|866.00|23.00|1
8.00|12.00|Slow Moving Moderate Value Item
```

BYTE and TEXT Information in ATS Files

When the information recorded in the ATS file includes BYTE or TEXT data, the RRD information is reported as shown in the following examples.

Example 1

```
"<1200, TEXT, PB 877(necromsv) 840338515(96/08/17 20:21:55)>
```

where

- 1200 is the size of the data
- TEXT is the data type (it is either BYTE or TEXT)
- PB is the storage type (PB when the BYTE or TEXT is stored in the tablespace, BB for blobspace storage)
- The next two fields are the server identifier and the time stamp for the column if conflict resolution rule is defined for this replicate and the column is stored in a tablespace.

Example 2

```
"<500 (NoChange), TEXT, PB 877(necromsv) 840338478(96/08/17 20:21:18)>
```

where

- (NoChange) after the 500 indicates that the TEXT data has a size of 500 but the data has not been changed on the source server. Therefore the data is not sent from the source server.

Example 3

```
<(Keep local blob),75400, BYTE, PB 877(necromsv) 840338515(96/08/17 20:21:55)>")
```

where

- `(Keep local blob)` indicates that the replicated data for this column was not applied on the target table, but the local BYTE data was kept. This usually happens when timestamp conflict resolution is defined and the local column has a time stamp greater than the replicated column.

Row-Information Spooling

Enterprise Replication provides another spooling option called row-information spooling (RIS). RIS logs the following types of information:

- Individual aborted row errors
- Replication exceptions (such as when a row is converted by Enterprise Replication from insert to update, or from update to insert, and so on)
- Special stored procedure return codes as defined by the application if a stored procedure is called to resolve a conflict

The RIS row information is written at the time that the data-synchronization component finishes processing the replicated row and can therefore also provide the local row information. (The ATS file is written after the transaction is rolled back, the locks for all the local rows are released and therefore does not include accurate local-row data.) The RIS filename algorithm is analogous to the one used by ATS, with the **ats** prefix replaced by **ris**. The RIS file that corresponds to the ATS file described in the previous section is, for example:

```
ris.pur1.sls4.D_2.960529_23:27:16.5
```

In addition to the four types of records printed in ATS, the RIS file also includes the following information.

Label	Name	Description
LRH	Local-row header	Indicates if the local row is found in the delete table and not in the target table.
LRS	Local-row shadow columns	Contains the server id and the time when the row is updated on the target server. This line is printed only if the replicate is defined with a conflict-resolution rule.
LRD	Local-row data	Contains the list of replicated columns extracted from the local row and displayed in the same order as the replicated row data. Similar to the replicated row data, each column is separated by a '|' and written in ASCII format or in hexadecimal format if severe errors are encountered and is unable to convert it to ASCII format.

BYTE and TEXT Information in RIS Files

When the information recorded in the RIS file includes BYTE or TEXT data, the RRD information is reported as shown in the following examples.

Example 1

```
"<1200, TEXT, PB 877(necromsv) 840338515(96/08/17 20:21:55)>
```

where

- 1200 is the size of the TEXT data
- TEXT is the data type (it is either BYTE or TEXT)
- PB is the storage type (PB for storage in the tablespace, BB for blobspace storage)
- The next two fields are the server identifier and the time stamp for the column if conflict-resolution rule is defined for this replicate and the column is stored in a tablespace.

Example 2

```
"<500 (NoChange), TEXT, PB 877(necromsv) 840338478(96/08/17 20:21:18)>
```

where (NoChange) after 500 indicates the TEXT data has size of 500 but the data has not been changed on the source server. Therefore the data is not sent from the source server.

The following output provides some examples of RIS files:

```
TXH Source ID:390 / Name:sls4 / CommitTime:96-05-29 23:26:24
TXH Target ID:392 / Name:pur1 / ReceiveTime:96-05-29 23:27:16
----------
RRH Row:1 / Collection Id: 24707077 / Replicate Id: 24707077 / Table:
stores@cdr.item / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS pur1(392)|96/05/29 23:27:20(833437640)
LRD 20|VGA Card 439|A|Acer|Chevrolet|VGA Card|439|422.00|538.00|994.00|13
.00|13.00|8.00|Normal Demand Moderate Value Item
RRS sls4(390)|96/05/29 23:26:18(833437578)
RRD 20|VGA Card 439|A|Acer|Chevrolet|VGA Card|439|422.00|538.00|788.00|13
.00|12.00|8.00|Normal Demand Moderate Value Item
----------
RRH Row:2 / Collection Id: 24707077 / Replicate Id: 24707077 / Table:
stores@cdr.item / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS pur1(392)|96/05/29 23:26:52(833437612)
LRD 10|Digitizer 319|A|Apple|Toyota|Digitizer|319|29.00|36.00|552.00|28.0
0|28.00|2.00|Normal Demand Moderate Value Item
RRS sls4(390)|96/05/29 23:26:20(833437580)
RRD 10|Digitizer 319|A|Apple|Toyota|Digitizer|319|29.00|36.00|581.00|28.0
0|2.00|2.00|Normal Demand Moderate Value Item
----------
RRH Row:3 / Collection Id: 24707080 / Replicate Id: 24707080 / Table:
stores@cdr.customer / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS ctr0(377)|96/05/29 23:26:35(833437595)
LRD 18|Larraine Hustin|2021 South Blvd.|I|8|(399)-2423745|8469.00|0.00|0.
00|0.00|83429.00|Untouched|This Customer belongs to the Nation of Kamas
RRS sls4(390)|96/05/29 23:26:22(833437582)
RRD 18|Larraine Hustin|2021 South Blvd.|A|8|(399)-2423745|0.00|0.00|0.00|
0.00|83429.00|Untouched|This Customer belongs to the Nation of Kamas
----------
```

```
RRH Row:4 / Collection Id: 24707077 / Replicate Id: 24707077 / Table:
stores@cdr.item / DbOp:Update
RRH CDR:14 (Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS pur1(392)|96/05/29 23:26:48(833437608)
LRD 9|Keyboard 173|A|ALR|Datsun|Keyboard|173|978.00|1399.00|912.00|23.00|
23.00|12.00|Slow Moving Moderate Value Item
RRS sls4(390)|96/05/29 23:26:23(833437583)
RRD 9|Keyboard 173|A|ALR|Datsun|Keyboard|173|978.00|1399.00|866.00|23.00|
18.00|12.00|Slow Moving Moderate Value Item
==========
TXH Transaction aborted

TXH ATS file:/tmp/p1s1/ats.pur1.sls4.D_2.960529_23:27:16.6 has also been created
for this transaction
```

The following RIS output is for a committed transaction with an aborted row and a replication order exception.

```
TXH Source ID:398 / Name:srv3 / CommitTime:96-05-30 08:43:41
TXH Target ID:395 / Name:srv0 / ReceiveTime:96-05-30 08:43:42
----------
RRH Row:2 / Collection Id: 25886728 / Replicate Id: 25886728 / Table: stor
es@cdr.stock / DbOp:Update
RRH CDR:3 (Update exception, row does not exist in target table, converte
d to insert) / SQL:0 / ISAM:0
LRH Local Row in delete:Y
LRS srv0(395)|96/05/30 08:43:39(833471019)
LRD 19|10|A|234.00|0.00|0.00|0.00|Delivery in 3 days|0.00|0.00|0.00|0|0|9
0|1996-05-30 08:42:13
RRS srv3(398)|96/05/30 08:43:41(833471021)
RRD 19|10|A|234.00|327.00|0.00|0.00|Delivery in 3 days|0.00|0.00|0.00|3|2
|19|1996-05-30 08:43:35
----------
RRH Row:1 / Collection Id: 25886737 / Replicate Id: 25886737 / Table: stor
es@cdr.orddetail / DbOp:Delete
RRH CDR:0 / SQL:0 / ISAM:143
LRS srv4(399)|96/05/30 08:43:30(833471010)
LRD 4|5|4|A|5|6.00|4590.00|0.00|0.00|02/04/1998|Issue from Site 2 Only|By
Road|CusCode=15, RecQty=9, OrdQty=33|0.00|0.00|0.00|0.00|4|1|3|1996-05-3
0 12:44:21
RRS srv3(398)|96/05/30 12:43:41(833471021)
RRD 4|5|4|A|5|3.00|2295.00|0.00|0.00|02/06/1998|Please Issue ASAP|By Roa
d|CusCode=15, RecQty=3, OrdQty=39|0.00|0.00|0.00|0.00|4|0|5|1996-05-30 1
2:43:27
==========
TXH Transaction committed
TXH Total number of rows in transaction:5
```

Replication Event Monitor Messages

The Replication Event Monitor displays messages in the following format:

```
Context : Message.
```

The context part of `Context:Message` includes an operation, such as `define server` or `suspend group`, and the name of the object of the operation. The following lines show three examples of context information:

```
define replicate 'repl_1'
start group 'accounts'
suspend server 'lake'
```

The message part of `Context:Message` consists of a message that describes the problem. The messages are documented in Appendix A.

Three `Context:Message` examples follow:

- `define replicate 'repl_1' : database does not exist`
- `start group 'accounts' : illegal group state change`
- `suspend server 'lake' : insufficient memory to perform operation`

Chapter

11

Command-Line Utility

The command-line utility (CLU) lets you configure and control Enterprise Replication from the command line on your UNIX or Windows NT operating system.

This chapter has the following sections:

- Definitions
  - List of commands
  - Command-line abbreviations
  - Option abbreviations
  - Backslash
  - Connect clause
  - Participant
  - Participant modifier
  - Server group
  - Return codes
  - Time formats
- Example of a command-line script
- Descriptions of each individual command-line command

Tip: *Some of the* ***cdr*** *options described in this chapter are not available in the current release. If you try to use an option that is not available, the server responds with an error message, "Feature not supported."*

Definitions

This section defines the terminology and conventions used in the descriptions of the command-line utility.

The CLU has many variations, such as **cdr define group** and **cdr resume replication**. Although technically ER has only *one* command-line utility, **cdr**, this chapter treats each variation of the CLU as if it were a separate command.

Each variation of the CLU follows the same approximate format, with the following components:

- Command and its variation

  The command specifies the action that should be taken.
- Options

  The options modify the action of the command. Each option starts with a minus (-) or a double-minus (--).
- Target

  The target specifies the Enterprise Replication object which should be acted upon.

The following table shows a few examples of CLU commands.

Command & Variation	Options	Target
cdr define group	--connect ohio	accounts replicate1 replicate2
cdr modify group	--sequential	accounts
cdr sta gro	-c ohio	accounts g_server1 g_server2
cdr def ser	-I -L -S	corporate store1
cdr list replicate		

Important: *You must be user* ***informix*** *to execute any of the CLU variations except the* ***cdr list*** *options.*

List of Commands

The following table shows the CLU command variations and the page where the command is documented.

Command	Page
cdr change group	11-18
cdr change replicate	11-20
cdr define group	11-22
cdr define replicate	11-25
cdr define server	11-30
cdr delete group	11-34
cdr delete replicate	11-35
cdr delete server	11-36
cdr disconnect server	11-38
cdr error	11-39
cdr list group	11-42
cdr list replicate	11-43
cdr list server	11-45
cdr modify group	11-46
cdr modify replicate	11-48
cdr modify server	11-51
cdr resume group	11-53
cdr resume replicate	11-54
cdr resume server	11-55
cdr start	11-56
cdr start group	11-57

(1 of 2)

Command	Page
cdr start replicate	11-58
cdr stop	11-59
cdr stop group	11-61
cdr stop replicate	11-62
cdr suspend group	11-63
cdr suspend replicate	11-64
cdr suspend server	11-65

(2 of 2)

Command-Line Abbreviations

Each of the words that make up a **cdr** command variation can be abbreviated to three or more characters. For example, the following commands are all equivalent:

```
cdr define replicate
cdr define repl
cdr def rep
```

With the exception of the keyword `transaction`, all words (or letter combinations) that are part of the command *options* must be written as shown in the syntax diagrams.

Option Abbreviations

Each option for a command has a long form and a short form. You can use either form, and you can mix long and short forms within a single command. For example, using long forms you can write:

```
cdr define server --connect=ohio --idle=500 --send=dbs1 \
--recv=dbs2 --ats=/cdr/ats --initial utah
```

Using short forms, you can write the example above as follows:

```
cdr def ser -c ohio -i 500 -s dbs1 -r dbs2 -A /cdr/ats -I utah
```

The long form is always preceded by a double minus (--). The short form is preceded by a single minus (-) and is the first letter of the long form. However, sometimes the short form is capitalized and sometimes not. To find the correct syntax for the short form, check the table that accompanies each command variation.

Tip: *Informix suggests that you use the long forms of options when you are preparing a script that might be used by other people.*

Order of the Options

You can specify the options of the CLU commands in any order. Some of the syntax diagrams in this chapter show the options in a specific order because it makes the diagram easier to read. You can choose any order for the options that you wish.

Tip: *Informix suggests that, for ease of maintenance, you always use the same order for your options.*

Backslash

Many CLU commands become quite long when all of the options are included. The examples in this chapter use a backslash (\) to indicate that a command continues on the next line. The following two commands are equivalent. The first command is too long to fit on a single line, so it is continued on the next line. The second example, which uses short forms for the options, all fits on one line.

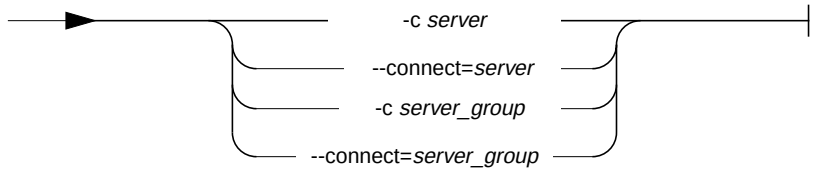
```
cdr define server --connect=ohio --idle=500 --send=dbs1 \
--recv=dbs2 --ats=/cdr/ats

cdr define server -c=ohio -i=500 -s=dbs1 -r=dbs2 -A=/cdr/ats
```

Check the documentation of your operating system to find out how to manage long lines at your command prompt.

Connect Option

The Connect option causes the command to use the global catalog that is on the specified database server instead of the global catalog that is on the server specified by **$INFORMIXSERVER**.

If this option is not invoked, the connection defaults to the database server specified by the **$INFORMIXSERVER** environment variable.



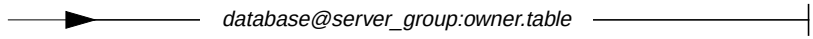
Element	Purpose	Restrictions	Syntax
server	Name of the database server to connect to	Must be the name of a database server that is registered with ER.	n/a
server_group	Name of the server group that includes the database server to connect to	Must be the name of an existing database server group.	n/a

You might use this clause if the database server to which you would normally attach is unavailable or, in some limited cases, to improve efficiency.

For more information about the global catalog, refer to "Global Catalog" on page 3-9.

Participant

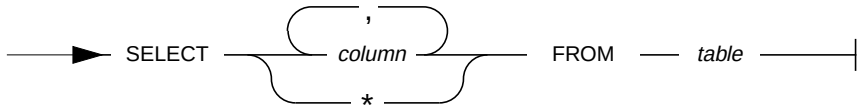
The participant identifies a table that should be replicated. You must include the server group, database, table, and table owner.



Element	Purpose	Restrictions	Syntax
database	Name of the database that includes the table to be replicated	The database server must be registered with ER.	
owner	User ID of the owner of the table to be replicated	None	
server_group	Name of the server group that includes the server to connect to	Must be the name of an existing server group.	
table	Name of the table to be replicated	The table must exist.	

Participant Modifier

The participant *modifier* is a restricted SELECT statement. The participant modifier specifies the rows that will be replicated.



Element	Purpose	Restrictions	Syntax
column	Name of a column in the table specified by the participant	The column must exist	
table	The table specified by the participant	Must be only a table name. You cannot specify an owner or a database server.	

The following restrictions apply to a SELECT statement that is used as a participant modifier:

- The FROM clause of the SELECT statement can reference only a single table.
- The table in the FROM clause must be the table specified by the participant.
- The columns selected must include a primary key
- The statement cannot include a join or a subquery.

For detailed information about the SELECT statement, refer to *Informix Guide to SQL: Syntax*.

Database-Server Group

Except in the Connect option, the CLU uses the *database-server group* name instead of the more familiar *database server* name (that is, the name specified in the **INFORMIXSERVER** environment variable) for all references to database servers. This manual also refers to a database-server group as a *server* or *server group*.

This manual uses the convention that the name of a database-server group starts with **g_**. This is only a convention; **g_** is not required syntax.

If you prefer, you can give your database-server group the same name as the first database-server name in the group. In that case, the term *server* would refer to either a database-server group or a database server.

For more information about database-server groups, refer to the *Administrator's Guide*.

UNIX

Server Groups on UNIX

On UNIX, a server group is defined as part of the **sqlhosts** file. In the following example, line 2 describes the database server **aserver**. Lines 3 and 4 define two more database servers that are aliases for **aserver**. Line 1 defines a server group, **g_aserver**, that includes **aserver**, **bserver**, and **aserver_shm**.

```
1  g_aserver      group       -          -           i=27
2  aserver        ontlitcp    myhost     svcname2    g=g_aserver
3  bserver        ontlitcp    myhost     svcname9    g=g_aserver
4  aserver_shm    onipcshm    myhost     xyz         g=g_aserver
```

All of the database server names (dbservername and dbserver aliases) must be grouped together after the line that defines the database-server group.

WIN NT

Server Groups on Windows NT

On Windows NT, use the Add Database Server wizard to add database servers and database-server groups to the IECC.

Return Codes

If the CLU command has an error, the server returns an error message and a return-code value. The message briefly describes the error. The return-code value corresponds to the return-code values listed in Appendix A.

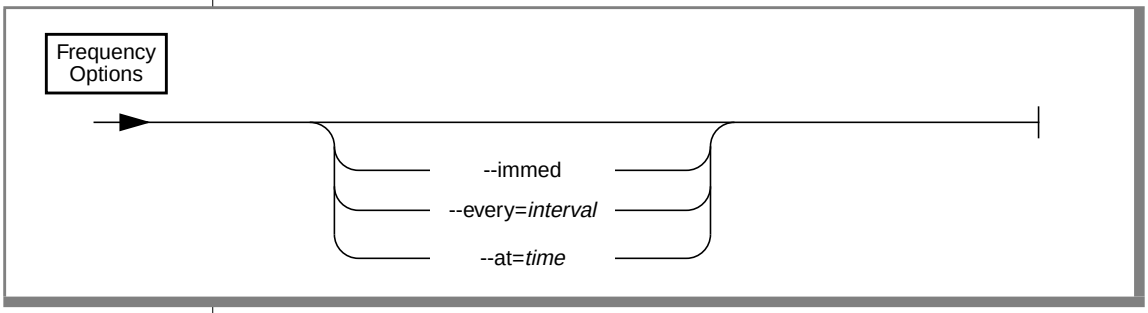
For example, the following command is incomplete because it does not include a conflict-resolution rule. The (58) is the error number as listed in Appendix A.

```
> cdr def rep r12 \
"host1@g_svr1:informix.customer" "select * from customer"
"host2@g_svr2:informix.customer" "select * from customer"

command failed -- No conflict resolution rule specified  (58)
```

Frequency Options

Several commands have options that allow you to specify the interval between replications or the time of day when an action should occur.

If no time option is specified, the action occurs immediately.



Element	Purpose	Restrictions	Syntax
interval	Time interval for replication	The smallest interval is minutes.	Intervals, p. 11-14
time	Specific time for replication	Time is given with respect to a 24-hour clock.	Time of Day, p. 11-13

The time options have the meanings shown in the following table.

Time Options		
Long Form	**Short Form**	**Meaning**
--immed	-i	Action occurs immediately.
--every	-e	Action occurs immediately and repeats at the frequency specified by interval.
--at	-a	Action occurs at the specified day and time.

Time of Day

The time of day is always given with respect to a 24-hour clock. For example, 19:30 is 7:30 P.M.

The `-a time` option lets you specify the day on which an action should occur. The string *time* can have the following formats:

- Day of week

  To specify a specific day of the week, give the either the long or short form of the day, followed by a period and then the time. For example, `--atime Sunday.18:40` or `-a Sun.18:40`, specifies that the action should occur every Sunday at 6:40 P.M.

  The day of the week is given in the locale of the client. For example, with a French locale, you might have `--atime Lunedi.3:30` or `-a Lun.3:30`. The time and day are in the time zone of the server.

- Day of month

  To specify a specific date in the month, give the date, followed by a period, and then the time. For example, `1.3:00` specifies that the action should occur at 3 A.M. on the first day of every month.

  The special character `L` represents the last day of the month. For example, `L.17:00` is 5 P.M on the last day of the month.

- Daily

  To specify that replication should occur each day, give only the time. For example, 4:40 specifies that the action should occur every day at 4:40 A.M.

Intervals

The `-e length` option lets you specify the interval between actions. The *length* of time between replications can be either of the following formats:

- The number of minutes

  To specify the number of minutes, specify an integer value. For example, `-e 60` indicates 60 minutes between replications.

- The number of hours and minutes

  To specify hours and minutes, give the number of hours, followed by a colon, and then the number of minutes. For example, `-e 5:45` indicates 5 hours and 45 minutes between replications.

Example Using the Command-Line Utility

This section contains a very simple example of replication that uses the command-line utility.

Prepare the Environment

To run this replication example, you need two Informix database servers. The database servers must be members of a database-server group. You *can* run two database server instances on one computer, but it is more realistic to use database servers on different computers. The example uses the following environment:

- Two computers are hosts for the database servers: **urban** and **rural**. All commands in this example, except for creation of the sample database **lake_db**, are issued by a person working on **urban**.
- The database servers are named **hill** and **lake**.

- Each database server is a member of a server group. The database servers **hill** and **lake** are member of the groups **g_hill** and **g_lake**, respectively.
- The databases for this example are identical **stores7** databases with logging:

  On the **hill** host (urban), create the **hill_db** database:

  ```
  urban> dbaccessdemo7 hill_db -log
  ```

  On the **lake** host (rural), create the **lake_db** database:

  ```
  rural> dbaccessdemo7 lake_db -log
  ```

Declare the Database Servers for Replication

Before replicating data, you must declare the database servers to Enterprise Replication.

1. Decide where the send queue should go.

   The send queue location is one of the options of **cdr define serve**r. If you do not specify a location for the send queue, it defaults to root dbspace. This example uses the default.

2. Declare the **hill** database server to ER.

   ```
   urban> cdr define server --init g_hill
   ```

   When you define a server for ER, you *must* use the name of the server group rather than the database server name.

3. Verify that the definition succeeded. You can use one of the following actions:

   - The first ER definition on a database server creates a **syscdr** database. Use DB-Access and choose **Database → Select** to verify that **syscdr** is present.
   - Use the CLU utility **cdr list server** to list existing ER servers.

4. Declare the second database server, **lake**, to ER.

```
urban> cdr def ser --connect lake --init --sync g_hill g_lake
```

   The `--connect` option allows you to declare **g_lake** to ER (and thus create the **syscdr** database) on the database server **lake** while working on the **urban** computer.

   The `--sync` option instructs the command to use the already-defined server (**g_hill**) as a pattern for the new definition.

5. Verify that the second definition succeeded. You can use one of the following actions:

   - Use DB-Access to check the ER servers in the **syscdr** database. The following two statements should show the same results:

```
SELECT servid, servname FROM syscdr@hill:servdef
SELECT servid, servname FROM syscdr@lake:servdef
```

   - Use the CLU utility **cdr list server**.

Define Replicates

The replicate definition has four basic parts: options, the replicate name, and two participants.

1. Define a replicate that records all changes to the **customer** table:

```
urban> cdr define repl -C ignore repl1 \
"hill_db@g_hill:informix.customer" "select * from customer" \
"lake_db@g_lake:informix.customer" "select * from customer"
```

2. Verify that the definition succeeded. You can use one of the following actions:

   - Use DB-Access to check the replicate definitions:

```
SELECT repid, repname, repstate FROM syscdr:repdef
```

   - Use the CLU utility **cdr list replicate**.

Start Replication

You have now *described* the replication environment, but no replication occurs until you instruct ER to start replicating. You can start and stop each replicate individually.

1. Start replication for the replicate that you just defined, **repl1**:

```
urban> cdr start repl -c g_hill repl1
```

2. Verify that the replication is active. Execute **cdr list replicate** again and verify that the STATE column has changed from INACTI to ACTIVE.

   At this point replication is active, but there has been no database activity.

3. Use DB-Access to make a change to the **hill** database:

```
INSERT INTO customer VALUES (0,'alice','adams',
'Aardvark', 121 First Street','','Atlanta','IN',
'47904', '317-463-5555')
```

4. Use DB-Access to verify that the change to **hill_db** database is reflected on the **lake_db** database:

```
SELECT * FROM lake_db@lake:customer
WHERE fname = 'alice'
```

Command-Line Utility Syntax

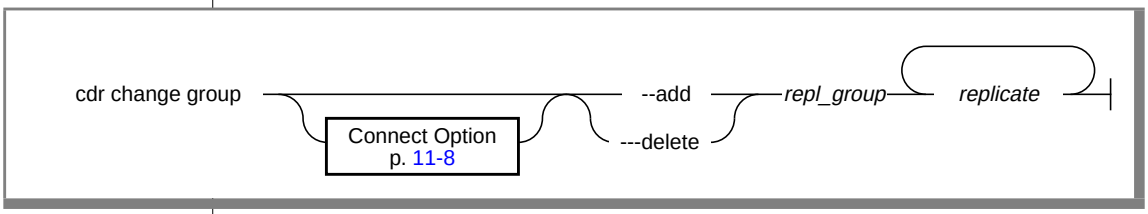
The next sections show the syntax for all of the variations of the **cdr** command-line utility.

The error messages that ER returns when a command fails are documented in Appendix A.

cdr change group

The **cdr change group** command changes a group by adding or deleting replicates.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of the replicate group to change	The replicate group must exist.	
replicate	Name of the replicate(s) to modify	The replicate(s) must exist.	

The options in **cdr change group** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--add	-a	Add replicate(s) to a group
--delete	-d	Remove replicate(s) from a group

Usage

Use this command to add replicates to a group or to remove replicates from a group.

Examples

The following example adds the replicates **house** and **barn** to group **newprod**.

```
cdr change group -add newprod house barn
```

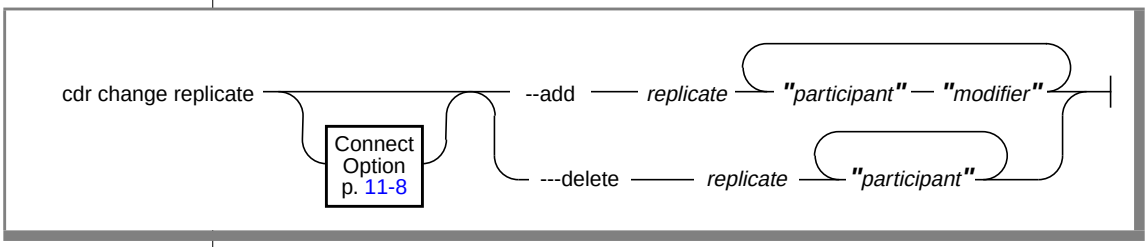
The following example removes the replicates **teepee** and **wigwam** from group **favorites**.

```
cdr cha gro -del favorites teepee wigwam
```

cdr change replicate

The **cdr change replicate** command allows you to modify an existing replicate by adding or deleting one or more participants.

Syntax



Element	Purpose	Restrictions	Syntax
modifier	Narrows the selection of rows for replication		Participant Modifier, p. 11-9
participant	Specifies the database server and table for replication		Participant, p. 11-9
replicate	Name of the replicate	The replicate name must be unique.	

The options in **cdr change replicate** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--add	-a	Add participant(s) to a replicate
--delete	-d	Remove participant(s) from a replicate

Usage

Use this command to add or delete a participant from a replicate. You can define a replicate that has zero or one participants, but to be useful, a replicate must have at least two participants.

Examples

The following example adds two participants to the replicate named **repl_1**: **db1@server1:antonio.table** with the modifier `select * from table1`, and **db2@server2:carlo.table2** with the modifier `select * from table2`.

```
cdr change repl -a repl_1 \
    “db1@server1:antonio.table1” “select * from table1” \
    “db2@server2:carlo.table2” “select * from table2”
```

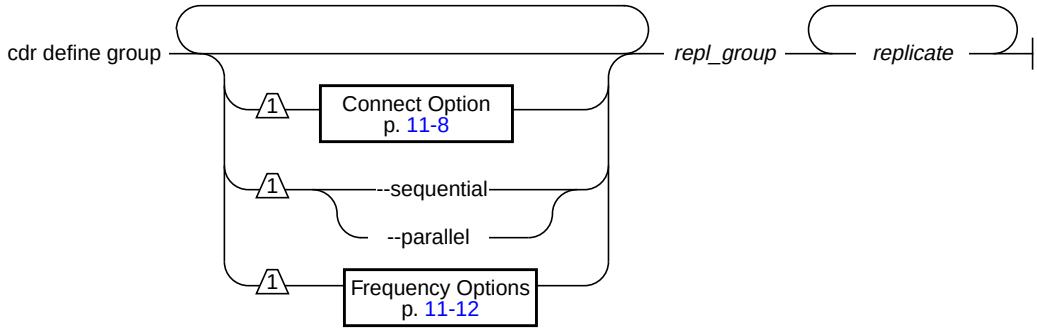
The following example removes the same two participants from replicate **repl_1**.

```
cdr change repl -d repl_1 \
    “db1@server1:antonio.table1” \
    “db2@server2:carlo.table2”
```

cdr define group

The **cdr define group** command defines a replicate group in the global catalog.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of group to create		
replicate	Name of a replicate to be included in the group		

The options in **cdr define group** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--parallel	-p	Process all groups in parallel. You can use this option only when replicates in the group have disjoint data domains. If this option is not set, the default is sequential processing of the groups.
--sequential	-s	Process all groups sequentially. The default is sequential processing.

Usage

The **cdr define group** command defines a replicate group. Any valid replicate can be defined as part of a replicate group. However, each replicate in the replicate group must meet the following restrictions:

- All of the replicates must be defined over the same set of database servers.
- All of the replicates must be in the same replication state.

  For information about replication states, refer to "Changing Replicate States" on page 7-26.

Once a replicate is defined as part of a replicate group, control and configuration is only allowed with replicate-group control commands. You cannot control a replicate individually until either the replicate is removed from the replicate group or the replicate group is deleted.

For a discussion of parallel versus sequential processing, refer to "Replicate Group" on page 3-7.

Example

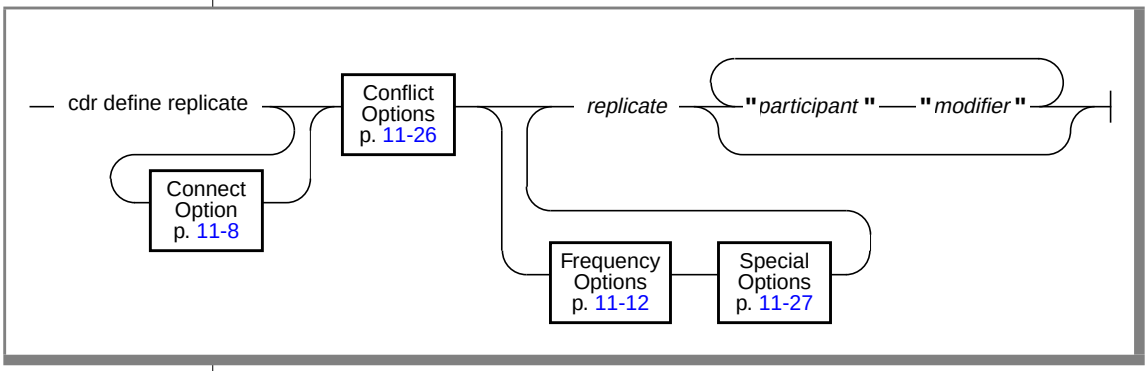
The following example connects to the default server and defines the replicate group **accounts** with replicate participants **repl1**, **repl2**, and **repl3**.

```
cdr define group accounts repl1 repl2 repl3
```

cdr define replicate

The **cdr define replicate** command defines a replicate in the global catalog.

Syntax



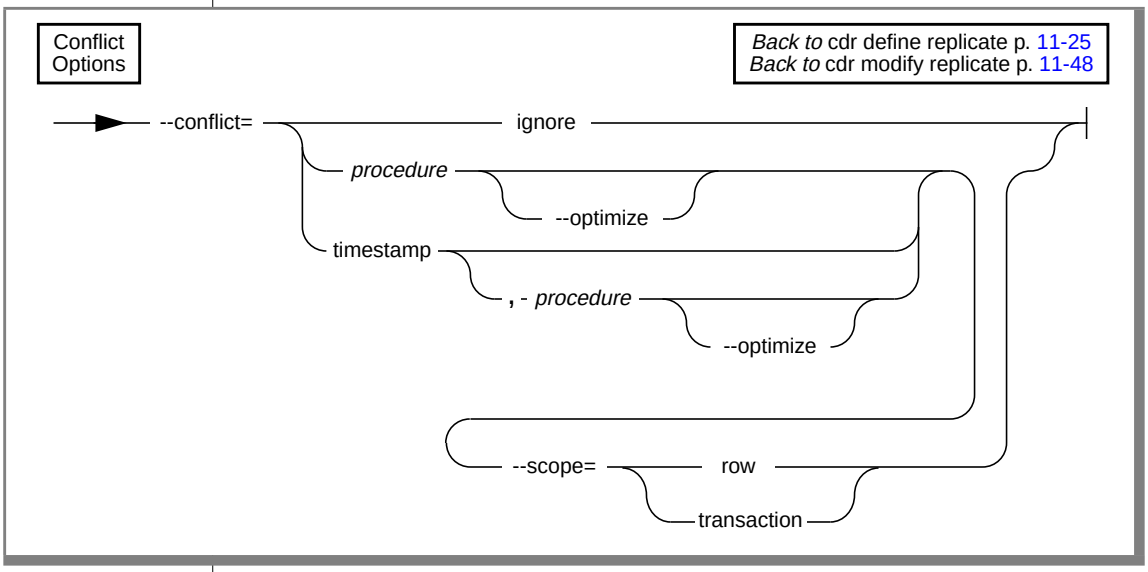
Element	Purpose	Restrictions	Syntax
modifier	Specifies columns to replicate	None	Participant Modifier, p. 11-9
participant	Name of a participant in the replication	The participant must exist at the time you execute the statement.	Participant Name, p. 11-9
replicate	Name of the new replicate	The replicate name must be unique.	

Usage

To be useful, a replicate must include at least two participants. You can define a replicate that has fewer than two participants, but before you can use that replicate, you must use **cdr modify replicate** to add more participants.

Conflict Options

The conflict options specify how ER should resolve conflicts among data that arrive at the database server. For information about the types of conflict resolution, refer to "Applying Data for Distribution" on page 3-23.



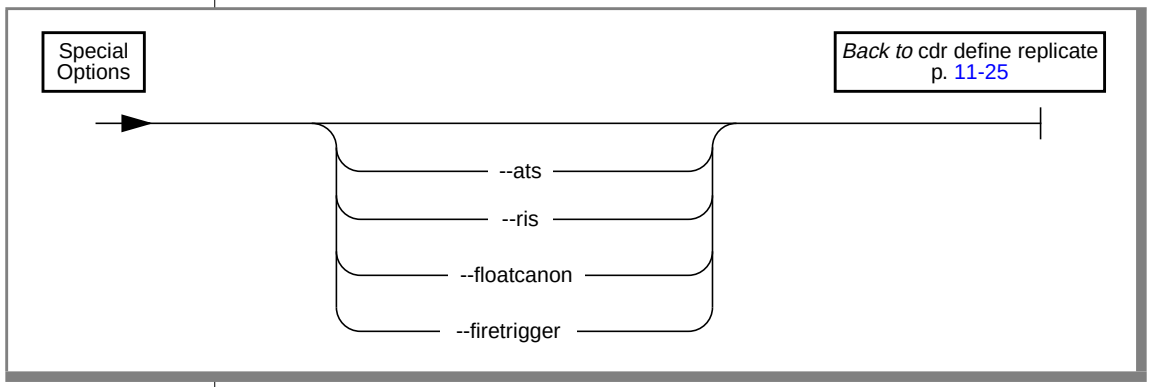
Element	Purpose	Restrictions	Syntax
procedure	Stored procedure for conflict resolution	The procedure must exist.	Procedure name

The conflict options have the meanings shown in the following table.

Conflict Options

Long form	Short form	Meaning
--conflict	-C	Specifies the rule that will be used for conflict resolution.
--optimize	-O	When a conflict occurs, and the row to be replicated meets the following conditions, the replicated row is accepted and the stored procedure is not called: ■ The row is from the same database server that last updated the local row on the target table. ■ The row has a time stamp greater than or equal to the local row.
--scope	-S	Specifies the scope for conflict resolution. If scope is not specified, the default scope is transaction. When specifying the scope, you can abbreviate *transaction* to fewer letters.

Special Options



The special options for **cdr define replicate** have the meanings shown in the following table.

Special Options		
Long form	**Short form**	**Meaning**
--ats	-A	Activate aborted-transaction spooling for replicate transactions that fail to be applied to the target database. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)
-ris	-R	Activates row-information spooling for replicate row data that fails conflict resolution or encounters replication order problems. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)
--floatcanon	-F	When floating-point numbers are replicated, they are transferred in canonical format. This format is portable between operating systems, but can lose accuracy.
--firetrigger	-T	The rows inserted by this replicate should fire triggers at the destination.

Examples

The following example illustrates the use of **cdr define replicate**.

Line 1 specifies a primary conflict-resolution rule of *time stamp*. If the primary rule fails, the stored procedure **sp1** will be invoked to resolve the conflict. Since no database server is specified here (or on any later line), the command connects to the server named in **$INFORMIXSERVER**.

Line 2 specifies that the replicate has a transaction scope for conflict-resolution scope and enables aborted-transaction spooling. The final item in the specifies the name of the replicate, **newrepl**.

Line 3 defines the first participant, "`db1@iowa:antonio.table1`" with the select statement "`select * from table1`".

Line 4 defines a second participant: "db2@utah:carlo.table2" with the select statement "select * from table2".

```
1 cdr define repl --conflict=timestamp,sp1 \
2 --scope=tran --ats newrepl \
3 "db1@iowa:antonio.table1" "select * from table1" \
4 "db2@utah:carlo.table2" "select * from table2"
```

The following example is the same as the example above with the following exceptions:

Line 1 instructs ER to use the global catalog on database server **ohio**.

Line 2 also specifies that the data should be replicated every five hours.

```
1 cdr def repl -c ohio -C timestamp,sp1 \
2 -S tran -A -e 5:00 newrepl \
3 "db1@iowa:antonio.table1" "select * from table1" \
4 "db2@utah:carlo.table2" "select * from table2"
```

The following example is the same as the example above except data will be replicated daily at 1:00 A.M.

```
cdr define repl -c ohio -C timestamp,sp1 \
     -s tran -A -a 1:00 replname \
      "db1@iowa:antonio.table1" "select * from table1" \
      "db2@utah:carlo.table2" "select * from table2"
```

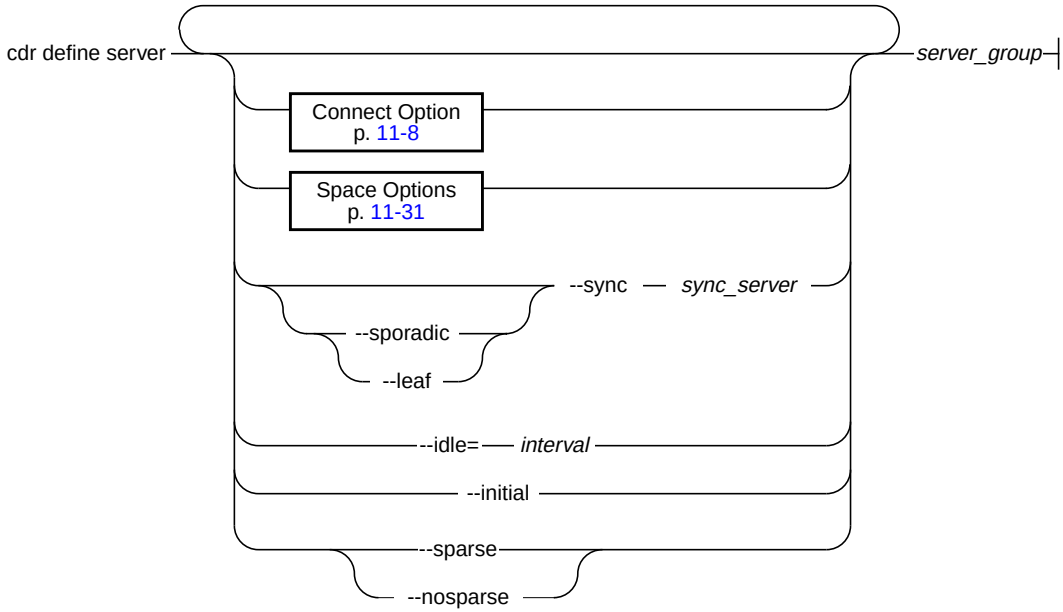
The following example is the same as the example above except data will be replicated on the last day of every month at 5:00 A.M.

```
cdr define repl -c ohio -C timestamp,sp1 \
-S tran -A -a L.5:00 replname \
"db1@iowa:owner.table" "select * from table" \
"db2@utah:table2" "select * from table2"
```

cdr define server

Use the **cdr define server** command to define a database server group and all of its members for Enterprise Replication.

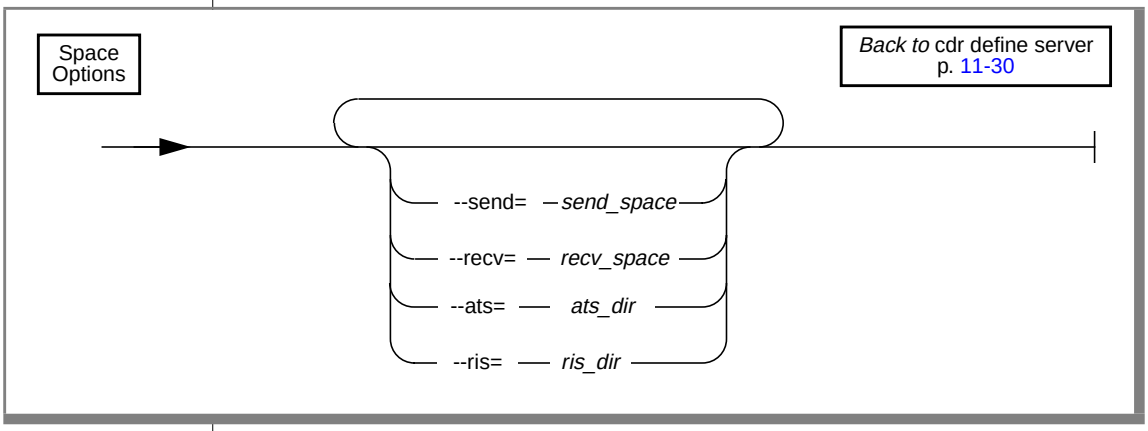
Syntax



Element	Purpose	Restrictions	Syntax
interval	Idle time-out for this server		Intervals, p. 11-14
server_group	Name of a server group to declare for ER	Must be the name of an existing server group.	
sync_server	Name of server to use for synchronization	Must be a server that is registered with ER. The server must be online.	

Space Options

The Space options specify storage locations for special options that the ER server can use.



Element	Purpose	Restrictions	Syntax
ats_dir	Name of the directory for aborted-transaction spooling	Must be a full pathname.	Directory name on your operating system.
recv_space	Name of the dbspace for the receive queue	The dbspace must already exist.	
ris_dir	Name of the directory for row-information spooling	Must be a full pathname.	Directory name on your operating system.
send_space	Name of the dbspace for the send queue	The dbspace must already exist.	

Usage

The **cdr define server** command enters a database server from the *server_group* into the ER global catalog.

Important: *A database server must be defined in the global catalog prior to participation in replication.*

The options for **cdr define server** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--ats	-A	Activates aborted-transaction spooling for replicate transactions that fail to be applied to the target database. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.")
--idle	-i	Causes an inactive connection to be terminated after *idle* time.
--initial	-I	Starts replication on *server_group*. The *server_group* must be the same as the connection server, which is either $INFORMIXDIR or a server specified by the Connect option.
--leaf	-L	Defines the server as a non-sporadic, non-root server with hub server *sync_server*.
--nosparse	-p	Maintains the full set of catalog information. This is the default for non-sporadic servers.
--receive	-r	Assigns the receive queue to the dbspace *recv_space*.
-ris	-R	Activates row-information spooling for replicate row data that fails conflict resolution or encounters replication-order problems. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)
--send	-s	Assigns the send queue to the dbspace *send_space*.
--sparse	-P	Maintains only a partial set of information in the global catalog. This is the default for sporadic servers.
--sporadic	-M	Defines the server as a sporadic, non-root server with hub server *sync_server*.
--sync	-S	Synchronizes the catalogs of *server_group* with the catalogs that are on the server *sync_server*.

Examples

The following example connects to the server **stan**, initializes Enterprise Replication, and defines the server **stan** with an idle time-out of 500 minutes. The ER send queue will be created in **dbspace1** and the receive queues will be created in **dbspace2**. Any files generated by ATS will go into directory **/cdr/ats**.

```
cdr define server --connect=stan --idle=500 --send=dbspace1 \
    --recv= dbspace2 -A /cdr/ats -I stan
```

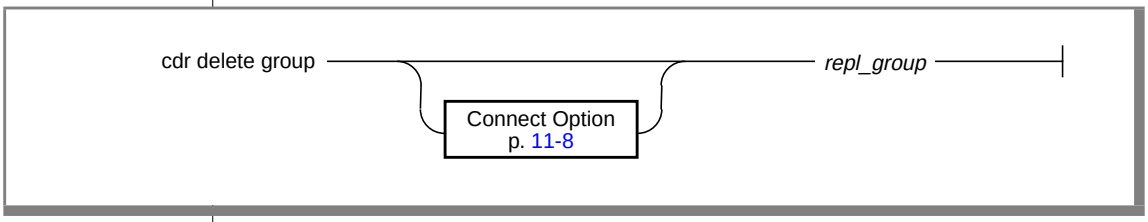
The following example connects to the server **oliver**, initializes Enterprise Replication, synchronizes its catalogs with the catalogs on server **stan**, and defines the server **oliver** with an idle time-out of 600 minutes. The send queue will be created in **dbspaceA** and the receive queues will be created in **dbspaceB**. Also, any files generated by ATS will go into directory **/cdr/ats**.

```
cdr define server -c oliver-i 600 -s dbspaceA -r dbspaceB \
     -A /cdr/ats -I -S stan oliver
```

cdr delete group

The **cdr delete group** command deletes a replicate group.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of group to delete	The group must exist.	

Usage

The **cdr delete group** command deletes the replicate group *repl_group* from the global catalog.

Tip: *The **cdr delete group** command does not affect the replicates or associated data. When a replicate group is deleted, the individual replicates within the group are unchanged.*

Example

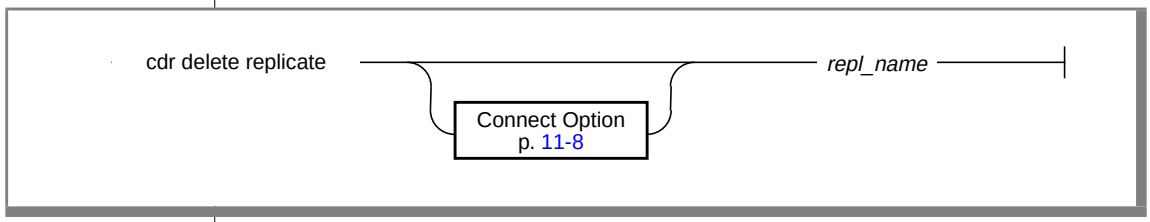
The following example connects to the default server and deletes the replicate group **accounts**:

```
cdr delete group accounts
```

cdr delete replicate

Use the **cdr delete replicate** command to delete a replicate from the global catalog.

Syntax



Element	Purpose	Restrictions	Syntax
repl_name	Name of the replicate to delete	The replicate must exist.	

Usage

The **cdr delete replicate** command deletes the replicate *repl_name* from the global catalog. All replication data for the replicate is purged from the send queue at all participating database servers. If *repl_name* is contained in a *replicate group* this command will fail. You must remove the replicate from the replicate group before you can delete it.

Example

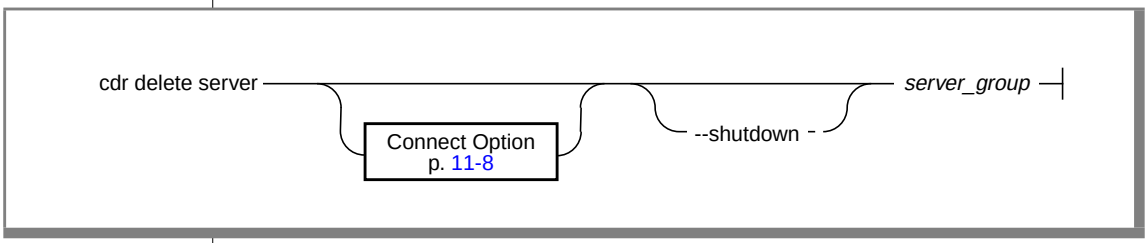
The following command connects to the default server and deletes the replicate **smile**:

```
cdr del rep smile
```

cdr delete server

Use the **cdr delete server** command to delete a database server from the global catalog.

Syntax



Element	Purpose	Restrictions	Syntax
server_group	Name of server group to remove from the global catalog.		

Usage

The specified database server is deleted from the global catalog. The database server is removed from all participating replicates. All replication data is purged from the send queues for the specified database server. Enterprise Replication is shut down and the catalogs are removed on the database server in *server_group*.

When you delete a server, you must issue the **cdr delete serve**r command *twice*. The first **cdr delete server** removes the server on the local host and removes the ER connection to other hosts. The second **cdr delete server** removes the server from all of the other ER hosts.

The options for **cdr delete server** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--shutdown	-X	Shut down ER and remove server information from the global catalog.

Shutdown Option

The shutdown option completely shuts down Enterprise Replication on the server *server_group* and purges all Enterprise Replication data and catalog information. Once this command is executed, you cannot run ER on *server_group* until you redefine the server with the **cdr define server** command. The shutdown option is *not* the default.

The shutdown option (`--shutdown` or `-X`) requires that *server_group* and the server specified by the Connect option be the same server. If the Connect option is omitted, then *server_group* must be the group that includes **$INFORMIXSERVER**.

Example

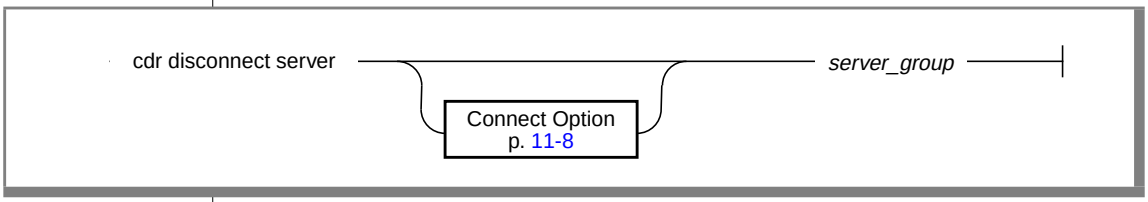
The following example connects to the server **tempsite**, shuts down, and removes replication:

```
cdr delete server tempsite
cdr delete server -c tempsite -X tempsite
```

cdr disconnect server

The **cdr disconnect server** command stops a server connection.

Syntax



Element	Purpose	Restrictions	Syntax
server_group	Name of the server group to disconnect	The server group must be currently active in ER.	Server Group, p. 11-10

Usage

The **cdr disconnect server** command drops the connection between *server_group* and the server specified in the connect option. If the Connect option is omitted, the command drops the connection between *server_group* and **$INFORMIXSERVER**.

Example

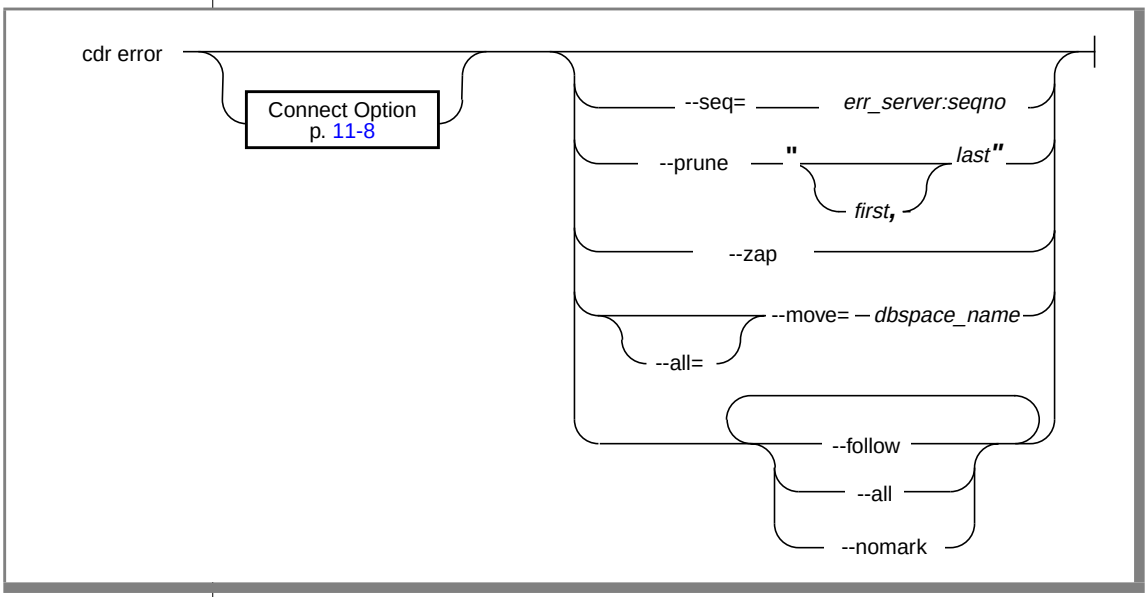
The following example drops the connection between the default server (**$INFORMIXSERVER**) and the serve group **g_store1**.

```
cdr disconnect server g_store1
```

cdr error

The **cdr error** command manages the error table and provides convenient displays of errors.

Syntax



Element	Purpose	Restrictions	Syntax
dbspace_name	The dbspace where the error table is stored.	The dbspace must exist.	
err_server	Name of server that holds the error table	The server must be registered for ER.	
first	Start date for a range	Must be a valid date.	Time of Day, p 11-13
last	Ending date for range	Must be a later date than *first*.	Time of Day, p 11-13
seqno	Sequence number of a specific error	Must be the number of an error in the error table.	

Usage

The options for **cdr define server** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
(no options)		Print the current list of errors. After the errors have been printed, mark them as *reviewed*.
--all	-a	When used with the -move option, move the error table to *dbspace* on all defined servers. Otherwise, print all errors, including those already reviewed.
--follow	-f	Continuously monitor the error table.
--move	-m	Move the error table to the dbspace *dbspace_name*.
--nomark	-n	Do not mark errors as *reviewed*.
--prune	-p	Prune the error table to those dates in the range from *first* to *last*.
--seq	-s	Remove the (single) error specified by *server:seqno* from the error table.
--zap	-z	Remove all errors from the error table.

Examples

The following command displays the current list of errors on database server **ernie**. After the errors are displayed, Enterprise Replication marks the errors as *reviewed*.

```
cdr error --connect=hill
```

The following command connects to the database server **lake** and removes from the error table all errors that occurred before the time when the command was issued.

```
cdr error -c lake --zap
```

The following command deletes all errors from the error table that occurred at or before 2:56 in the afternoon on May 1, 1996.

```
cdr error -p "1996-05-01 14:56:00"
```

The following command deletes all errors from the error table that occurred at or after noon on May 1, 1996 and before or at 2:56 in the afternoon on May 1, 1996.

```
cdr error -p "1996-05-01 14:56:00,1996-05-01 12:00:00"
```

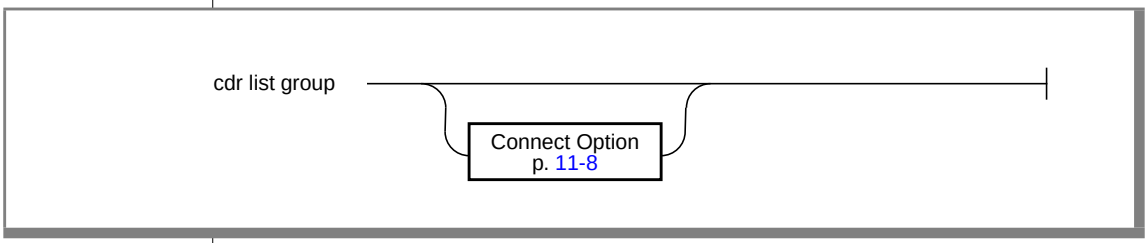
The following command moves the error table to **dbspace1** on all defined servers.

```
cdr error -a -m dbspace1
```

cdr list group

The **cdr list grou**p command displays information about the groups on the current server.

Syntax



Usage

The **cdr list group** command displays information about replication groups. You do not need to be user **informix** to use this command.

Example

The following example displays a list of the groups on the current server:

```
cdr list group
```

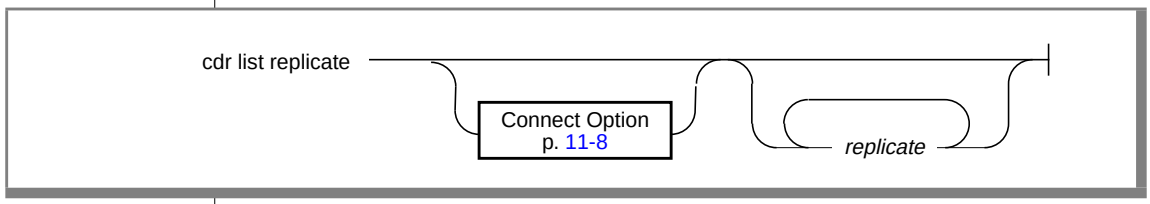
Sample output:

```
GROUP       SEQ  PARTICIPANTS
-------------------------------------------------------
weekly       Y   rep1, rep2, rep3
NewGroup     N   one, three, seven
```

cdr list replicate

The **cdr list replicate** command displays information about the replicates on the current server.

Syntax



Element	Purpose	Restrictions	Syntax
replicate	Name of the replicate(s)	The replicate must exist.	

Usage

The **cdr list replicate** command displays information about replicates. If no replicates are named, the command lists all replicates on the current server. If one or more replicates are named, the command displays detailed information about those replicates.

You do not need to be user **informix** to use this command.

Example

The following example displays a list of the replicates on the current server:

```
cdr list replicate
```

The output from the previous command might be the following:

```
REPLICATE  STATE      CONFLICT      FREQUENCY  OPTIONS
------------------------------------------------------------------------------
myrep      INACTI     ignore        immediate  transaction,ris,ats
rOne       ACTIVE     ignore        immediate  transaction,ris,ats
rTwo       ACTIVE     ignore        immediate  transaction,ris,ats
```

In addition to ACTIVE and INACTI (inactive), the **STATE** column can include DEFN FAI (definition failure).

The following example specifies the names of replicates:

```
cdr list repl rOne rTwo
```

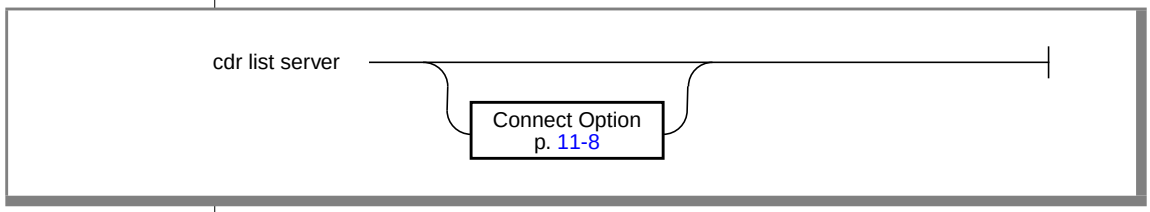
The following output might result from the previous command:

```
REPLICATE  SERVER   STATE   TABLE      SELECT
--------------------------------------------------------------------------------
rOne       g_srv1   INACT  customer    Select * from informix.state
rOne       g_srv2   INACT  customer    Select * from informix.state
rTwo       g_srv1   INACT  customer    Select address1, address2, city, company,
customer_num, fname, lname, phone, state, zipcode from informix.customer
rTwo g_srv2 INACT customer Select address1, address2, city, company,
customer_num, fname, lname, phone, state, zipcode from informix.customer
```

cdr list server

The **cdr list server** command displays a list of the Enterprise Replication servers that are defined on the current server.

Syntax



Usage

The **cdr list server** command display information about servers. You do not need to be user **informix** to use this command.

Example

The following example displays a list of the servers on the current server:

```
cdr list server
```

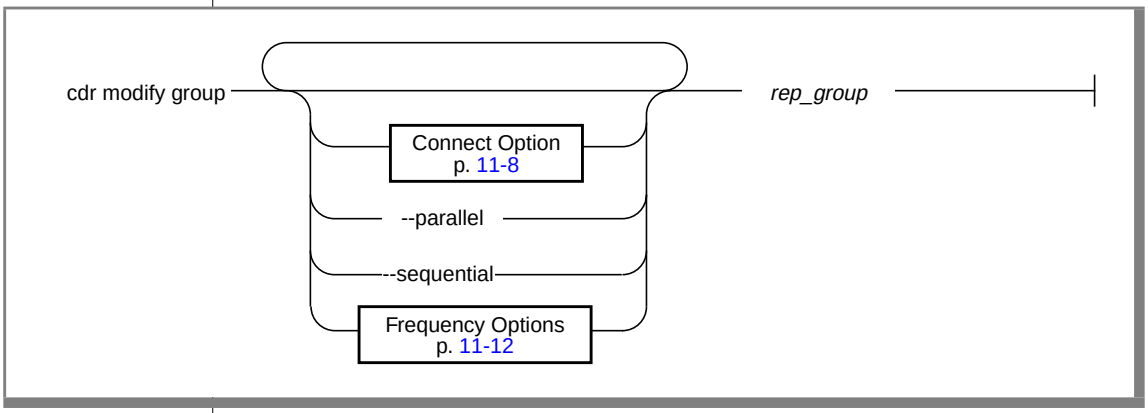
The previous command might give the following output:

```
SERVER    ID    STATE    STATUS    CONNECTION    CHANGED
------------------------------------------------------------
g_svr1    33    ACTIVE
g_svr2    34    ACTIVE
```

cdr modify group

The **cdr modify group** command modifies a replicate group.

Syntax



Element	Purpose	Restrictions	Syntax
rep_group	Name of group to modify		

Usage

The **cdr modify group** command modifies the attributes of the replicate group *rep_group*. Attributes of the group that are not specifically mentioned remain unchanged. To add or delete replicates to a group, use the **cdr change group** command.

The options in **cdr modify group** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--parallel	-p	Process members of the group in parallel.
--sequential	-s	Process members of the group sequentially.

Example

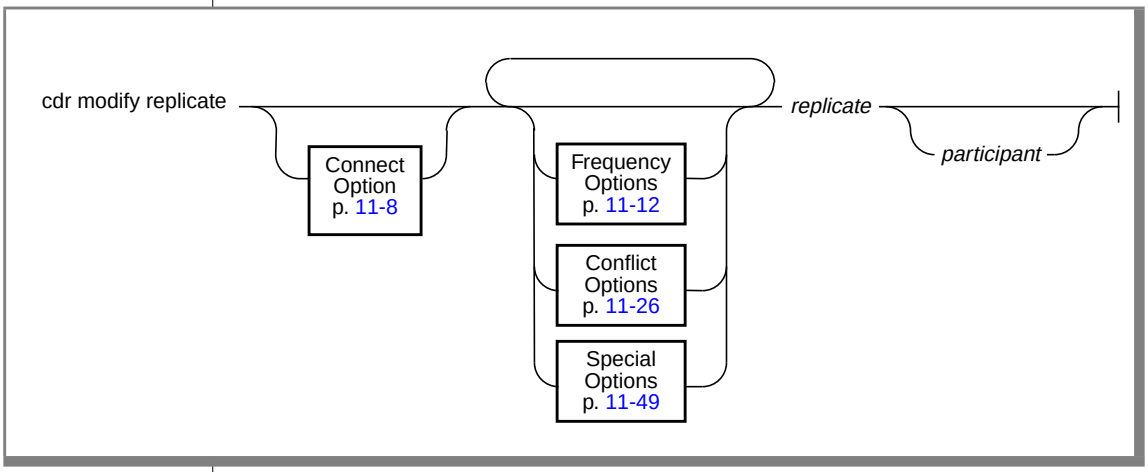
Connect to the default server (**$INFORMIXSERVER**) and modify the replicate group **accounts** to process replication data for the group sequentially.

```
cdr modify group --sequential accounts
```

cdr modify replicate

Use the **cdr modify replicate** command to modify replicate attributes.

Syntax



Element	Purpose	Restrictions	Syntax
participant	Name of a participant in the replication	The participant must exist at the time you execute the statement.	Participant Name, p. 11-9
replicate	Name of the replicate to modify	The replicate name must be unique.	Replicate Name, p. 11-25

Usage

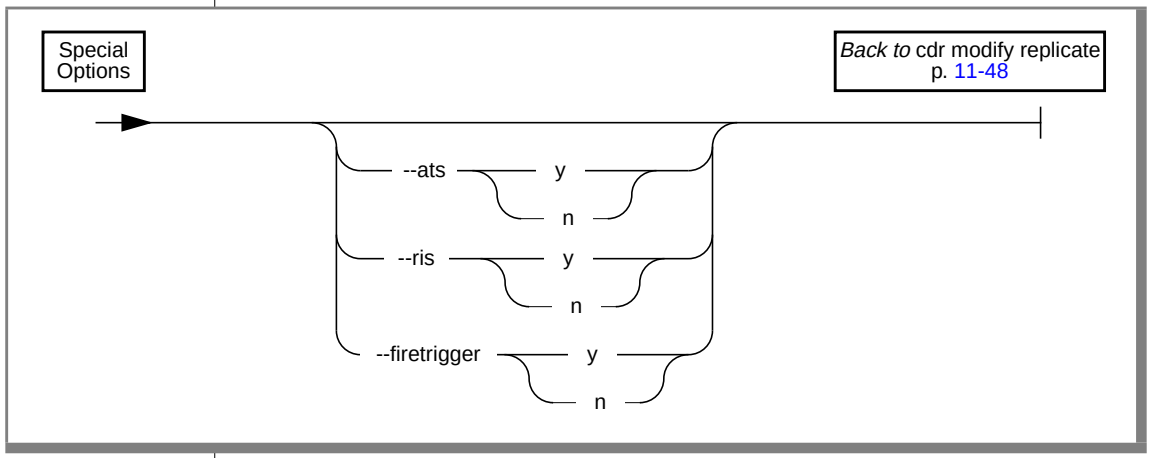
The **cdr modify replicate** command modifies a replicate. You can change the attributes of a participant.

Restrictions

The attributes for **cdr modify replicate** are the same as the attributes for **cdr define replicate**, with the following exceptions:

- You cannot change the canonical float format.
- You cannot change the conflict resolution from ignore to a non-ignore option. However, you can change from time-stamp resolution to procedure resolution.
- The ATS, RIS, and Fire Triggers options require a yes (y) or no (n) argument.

Special Options



The special options for **cdr define replicate** have the meanings shown in the following table.

Special Options		
Long form	**Short form**	**Meaning**
--ats	-A	Activate (y) or deactivate (n) aborted-transaction spooling for replicate transactions that fail to be applied to the target database. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)
--firetrigger	-T	Cause the rows inserted by this replicate to fire (y) or not-fire (n) triggers at the destination.
-ris	-R	Activate (y) or deactivate (n) row-information spooling for replicate row data that fails conflict resolution or encounters replication-order problems. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)

Examples

Modify the frequency attributes of replicate **smile** to replicate every 5 hours:

```
cdr modify repl -every=300 smile
```

Modify the frequency attributes of replicate **smile** to replicate daily at 1:00 A.M.:

```
cdr modify repl -a 01:00 smile
```

Modify the frequency attributes of replicate **smile** to replicate on the last day of every month at 5:00 A.M.:

```
cdr modify repl -a L.5:00 smile
```

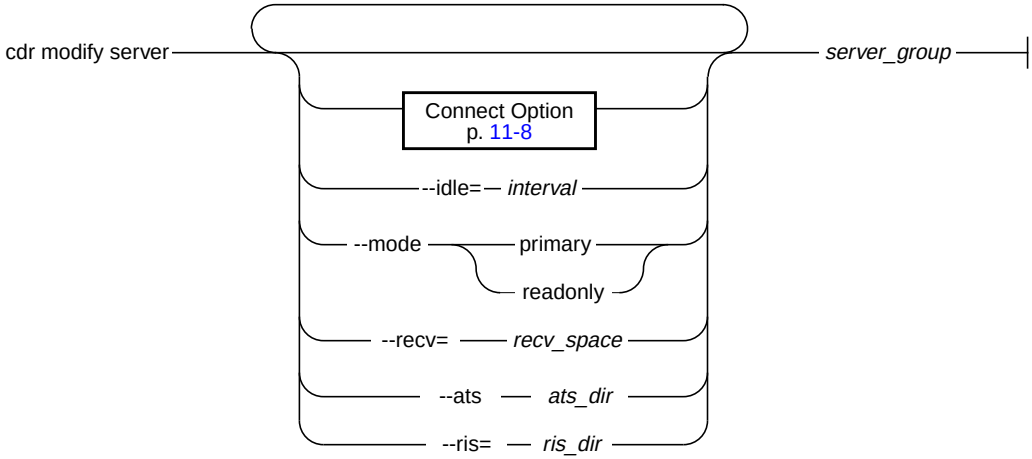
Change the mode of the first participant listed to read-only, and the mode of the second to primary. This example could be of a fail-over: setting the current master to a slave, and a former slave to the master. (This situation might occur if **server1** needed to be taken down for some reason.)

```
cdr mod repl smile "R db1@server1:antonio.table1" \
                   "P db2@server2:carlo.table2"
```

cdr modify server

Use the **cdr modify server** command to modify the ER attributes of a database server.

Syntax



Element	Purpose	Restrictions	Syntax
ats_dir	Name of aborted-transaction spooling directory	Must be a full pathname.	Directory name on your operating system.
interval	Idle time-out for this server		Intervals, p. 11-14
recv_space	Name of the dbspace for the receive queue	The dbspace must already exist.	
ris_dir	Name of the row-information spooling directory	Must be a full pathname.	Directory name on your operating system.
server_group	Name of a server group to define for ER	The server group must already exist.	

Usage

The **cdr modify server** command modifies the replication database server *server_group*.

The options for **cdr modify server** have the meanings shown in the following table.

Options		
Long form	**Short form**	**Meaning**
--ats	-A	Activate aborted-transaction spooling for replicate transactions that fail to be applied to the target database. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)
--idle	-i	Causes an inactive connection to be terminated after *idle* time.
--mode	-m	Change the mode of all replicates using this server to primary or to read-only. You can use the abbreviations `p` and `r`.
--receive	-r	Assigns the receive queue to the dbspace *recv_space*.
-ris	-R	Activates row-information spooling for replicate-row data that fails conflict resolution or encounters replication-order problems. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.".)

Examples

The following example connects to the server **paris**, and modifies the idle time-out of server group **g_rome** to 10 minutes. ATS files go into the directory **/cdr/atsdir**.

```
cdr modify server-c paris -i 10 -A /cdr/atsdir g_rome
```

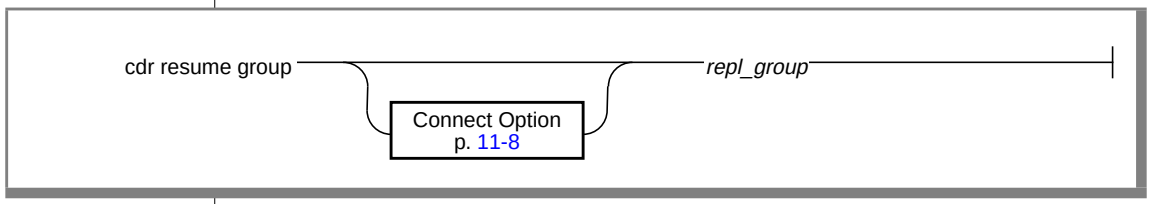
The following example connects to the default server (**$INFORMIXSERVER**) and set the modes of all participants on **g_geometrix** to primary.

```
cdr modify server -m primary g_geometrix
```

cdr resume group

Use the **cdr resume group** command to resume delivery of replication data for replicates defined in a replicate group.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of group to resume	The group must exist.	

Usage

The **cdr resume group** command causes all replicates contained in the replicate group *repl_group* to enter the active state (capture, send) for all participants.

For more information about replication states, refer to "Changing Replicate States" on page 7-26.

Example

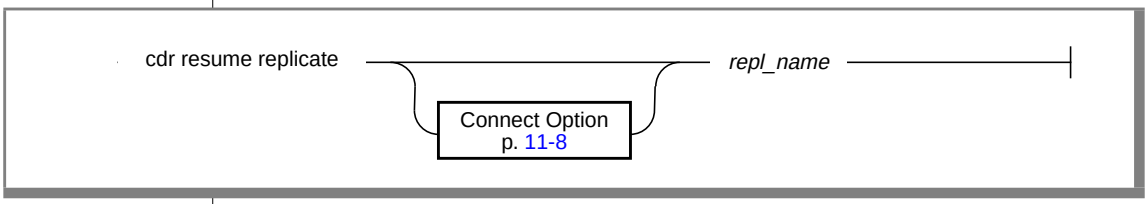
The following example connects to the default server and resumes the replicate group **accounts**:

```
cdr resume group accounts
```

cdr resume replicate

Use the **cdr resume replicate** command to resume delivery of replication data.

Syntax



Element	Purpose	Restrictions	Syntax
repl_name	Name of the replicate to change to active state.	The replicate must not be a member of a replicate group.	

Usage

The **cdr resume replicate** command causes all participants in the replicate *repl_name* to enter the active state.

If the replicate *repl_name* is a member of a replicate group, this command will fail. When *repl_name* is in a replicate group, use the **cdr resume group** command.

For more information on replicate states, refer to "Changing Replicate States" on page 7-26.

Example

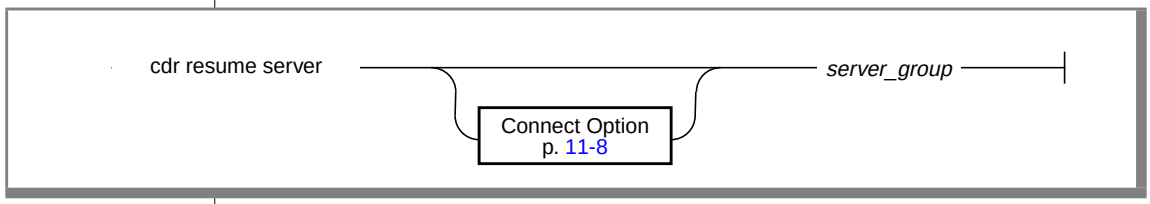
The following example connects to the default server (**$INFORMIXSERVER**) and resumes the replicate **smile**:

```
cdr resume repl smile
```

cdr resume server

Use the **cdr resume server** command to resume delivery of replication data to a database server.

Syntax



Element	Purpose	Restrictions	Syntax
server_group	Name of server group to resume	The server group must be defined in ER.	Server Group, p. 11-10

Usage

The **cdr resume server** command restarts suspended replication to the server *server_group*. The server must have been previously suspended with the **cdr suspen**d server command.

Example

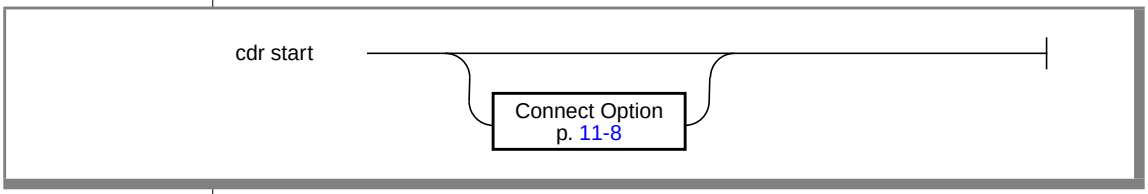
The following example resumes replication on the server groups **g_utah** and **g_iowa**:

```
cdr resume serv g_utah g_iowa
```

cdr start

The **cdr start** command starts Enterprise Replication processing.

Syntax



Usage

Use **cdr start** to restart Enterprise Replication after it is stopped with **cdr stop**. When you issue **cdr start**, Enterprise Replication threads start and continue from the point at which they stopped.

Enterprise Replication resumes evaluation of the logical log (if required for the instance of Enterprise Replication) at the *replay* position. The replay position is the position where Enterprise Replication stops evaluating the logical log when **cdr stop** is executed. If the evaluation process is running and the logical log ID for the replay position no longer exists when Enterprise Replication is started, then the restart partially fails (the database server log contains an error message stating that the replay position is invalid). If the restart partially fails, no database updates performed on the local database server are replicated.

Warning: *Informix recommends that you issue **cdr start** and **cdr stop** with extreme caution. Use these commands during a period of little or no database activity, and keep the downtime to a minimum. Transactions that occur during this period are not replicated and hence tables can become unsynchronized.*

Example

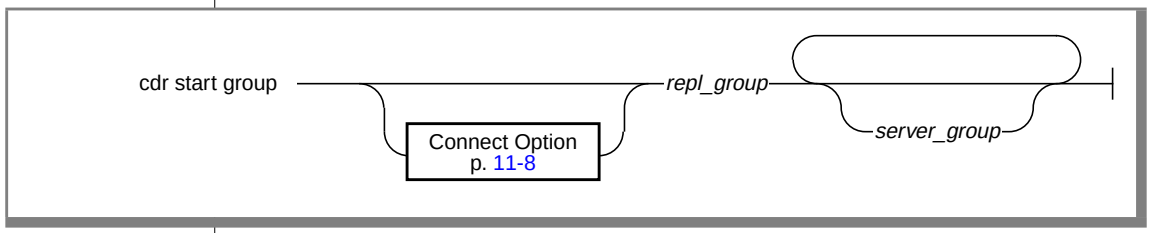
The following example restarts ER processing on database server **utah**:

```
cdr start -c utah
```

cdr start group

The **cdr start group** command starts the capture and transmittal of replication transactions for the replicates contained in a replicate group.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of replicate group to start	The group must exist	
server_group	Name(s) of server groups(s) on which to start the group		Server Group, p. 11-10

Usage

The replicates defined in the replicate group *repl_group* enter the active state (capture-send) on the database servers in *server_group*. If no server group is specified, all database servers in the replication system enter active state.

Example

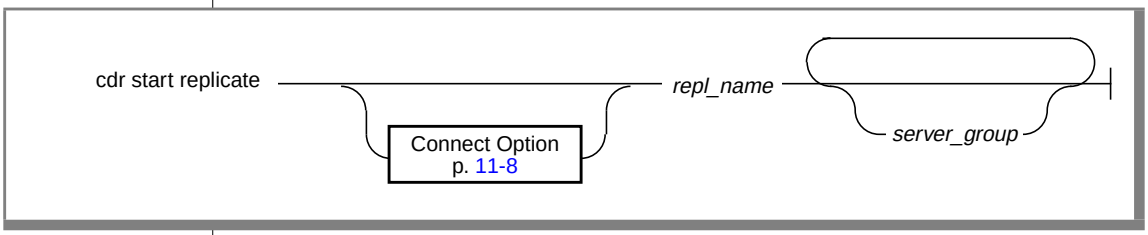
Connect to the default database server and start the replicate group **accounts** on the server groups **g_hill** and **g_lake**.

```
cdr start group accounts g_hill g_lake
```

cdr start replicate

Use the **cdr start replicate** command to begin to capture and transmit replication transactions.

Syntax



Element	Purpose	Restrictions	Syntax
repl_name	Name of the replicate to start	The replicate name must already be defined.	
server_group	Name of server group(s) on which to start the replicate	The server group(s) must already exist.	

Usage

The replicate *repl_name* enters the active state (data is captured and sent) on the server(s) *target*. If no *target* server is specified, the *repl_name* starts on all servers that are defined in the Enterprise Replication system.

If *repl_name* is a member of a replicate group, this command will fail. When *repl_name* is in a replicate group, use **cdr start grou**p.

Example

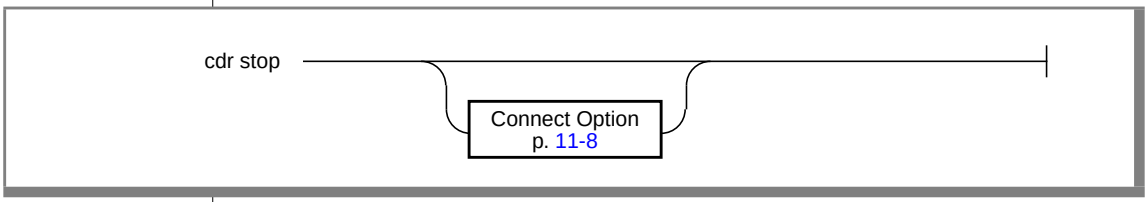
The following command connects to database server **lake** and starts the replicate **accounts** on the server groups **g_svr1** and **g_svr2**:

```
cdr start replicate --connect=lake accountsg_svr1 g_svr2
```

cdr stop

The **cdr stop** command stops Enterprise Replication processing.

Syntax



Usage

In most situations, ER starts when **cdr define server** is first executed. The replication threads remain running until the database server is shut down, or until the local database server is deleted with the **cdr delete server**. If the database server is shut down while ER is running, replication begins again when the database server is restarted.

Under rare conditions, users may want to temporarily stop the ER threads without stopping the database server. The **cdr stop** command shuts down all ER threads in an orderly manner. When the shutdown of ER is complete, the message `CDR: shutdown complete` appears in the database server log file.

After issuing the **cdr stop** command, replication threads remain stopped (even if the database server is stopped and restarted) until a **cdr start** command is issued.

Warning: *If you issue **cdr stop** and database activity continues, the database server from which the command is issued and the other database servers participating in replicates might become inconsistent. To ensure consistency, verify that no database update activity occurs while Enterprise Replication is stopped.*

Example

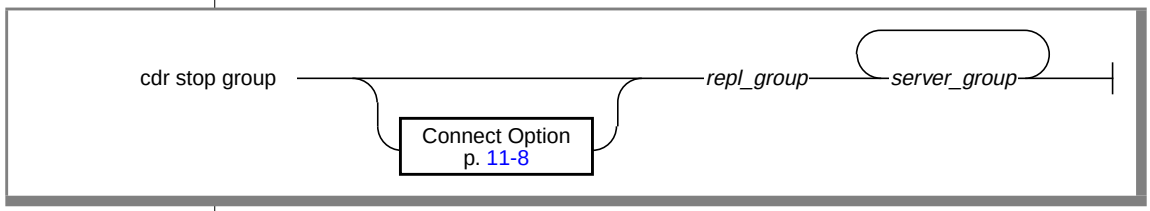
The following example stops ER processing on database server **paris**. Processing does not resume until a **cdr start** command restarts it.

```
cdr stop -c paris
```

cdr stop group

Use the **cdr stop group** command to stop capture and transmittal transactions for replicates contained in a replicate group.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of replicate group to stop	The group must be a currently active group.	
server_group	Name of server group(s) on which to stop the replicate group		Server Group, p. 11-10

Usage

All replicate data defined in the replicate group *repl_group* enters the *inactive* state *(*no capture, no send*)* on the database server(s) in *server_group*. If no server group(s) are specified, the replicate group *repl_group* enters the inactive state for all database servers participating in the replicate group.

Example

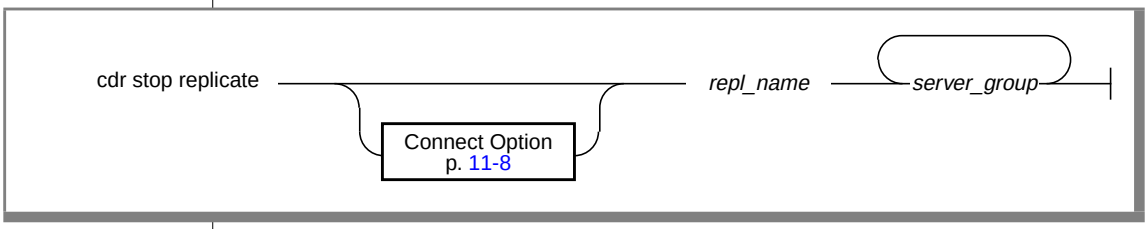
The following example connects to the database server **paris** and stops the group **accounts** on server groups **g_utah** and **g_iowa**.

```
cdr stop group --connect=paris accounts g_utah g_iowa
```

cdr stop replicate

The **cdr stop replicate** command stops the capture and transmittal of transactions for replication.

Syntax



Element	Purpose	Restrictions	Syntax
repl_name	Name of the new replicate	The replicate name must be unique.	
server_group	Name of server groups(s) on which to stop the replicate	The server group must already exist.	

Usage

The replicate *repl_name* enters the inactive state (no capture, no send) on the replicate database server(s) *server_group*. If no servers are specified, the replicate enters the inactive state on all database servers participating in the replicate.

If the replicate *repl_name* is contained in a replicate group, this command will fail. When *repl_name* is in a replicate group, use the **cdr stop group** command.

Example

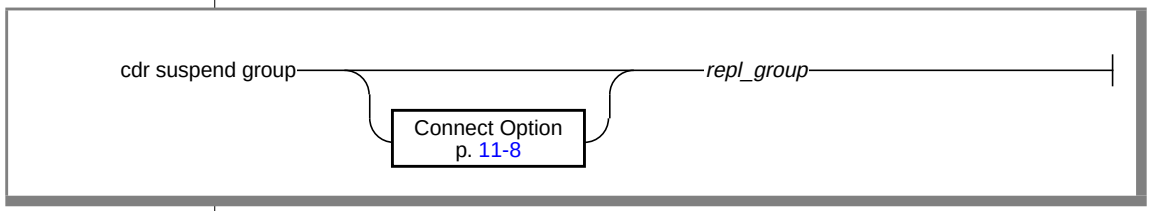
The following command connects to the database server **lake** and stops the replicate **aRepl** on server groups **g_server1** and **g_server2**:

```
cdr stop replicate -c lake aRepl g_server1 g_server2
```

cdr suspend group

The **cdr suspend group** command suspends delivery of replication data for replicates contained in a replicate group.

Syntax



Element	Purpose	Restrictions	Syntax
repl_group	Name of group to stop	The group must be a currently active group.	

Usage

The replicates contained in the replicate group *repl_group* enter the suspend state. No information is captured or sent for any replicate in the group.

Example

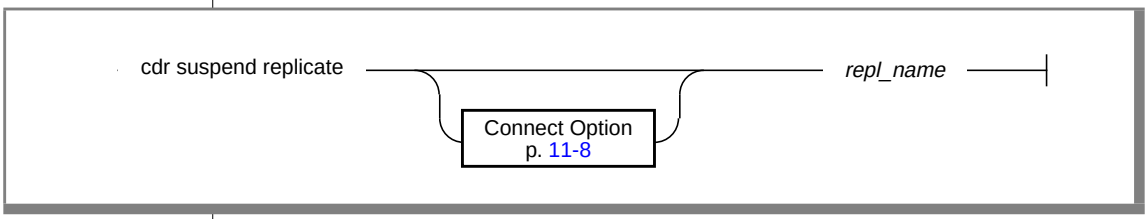
The following example connects to the default server (**$INFORMIXSERVER**) and suspends the replicate group **accounts**:

```
cdr suspend group accounts
```

cdr suspend replicate

Use the **cdr suspend replicate** command to suspend delivery of replication data.

Syntax



Element	Purpose	Restrictions	Syntax
repl_name	Name of the new replicate	The replicate name must be unique.	

Usage

The replicate *repl_name* enters the suspend state (capture, no send) for all participants.

If the replicate *repl_name* is contained in a replicate group, this command will fail. When *repl_name* is in a replicate group, use the **cdr suspend group** command.

Example

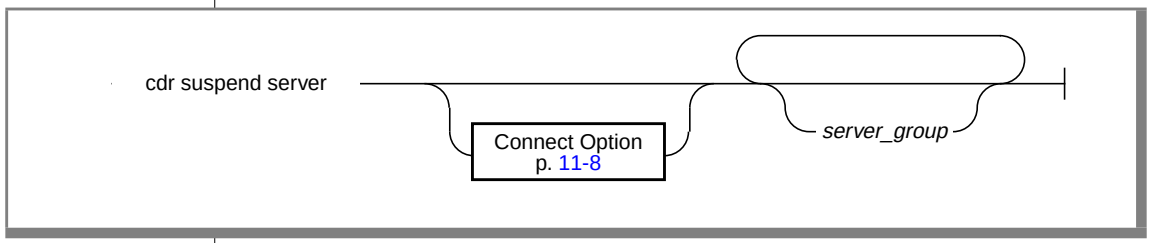
The following example connects to the server **stan** and suspends the replicate **house**:

```
cdr sus repl --connect stan house
```

cdr suspend server

Use the **cdr suspend server** command to suspend the delivery of replication data to a database server.

Syntax



Element	Purpose	Restrictions	Syntax
server_group	Name of server group(s) to suspend	The server group must be currently active in ER.	Server Group, p. 11-10

Usage

The **cdr suspend server** command suspends d11-9elivery of replication data to the database server(s) included in *server_group*. Replication data continues to all other database servers.

If no servers are specified, this command suspends all database servers participating in the Enterprise Replication system.

Tip: *Unlike **cdr suspend replicate**, the replicate and/or replicate group status is irrelevant.*

Example

The following example connects to the default server (**$INFORMIXSERVER**) and suspends the servers **g_iowa**, **g_ohio**, and **g_utah**.

```
cdr suspend serv g_iowa, g_ohio, g_utah
```

Chapter

12

API Commands

The application programming interface (API) calls for Enterprise Replication to provide a single control point to configure and control replication objects. Replication objects consist of database servers, replicates, and replicate groups.

Several API calls execute specific commands at a given time. All times are stored in the global catalog in Greenwich mean time (GMT).

Tip: *Some of the options described in this chapter are not available in the current release. If you try to use an option that is not available, the server returns message number 101, "Feature not supported."*

Structures Used by Enterprise Replication APIs

The API commands documented in this chapter use the following structures. These structures are also documented in **incl/esql/cdrapi.h** under the **$INFORMIXDIR** directory.

```
typedef struct plist
{
    char      serv[CDR_GL_NAMESIZE + 1];          /* server name */
    char      db[CDR_GL_NAMESIZE + 1];             /* database */
    char      table[CDR_GL_NAMESIZE + 1];          /* table */
    char      owner[CDR_GL_USERSIZE + 1];          /* owner */
    char      selectstmt[CDR_GL_SELECTSIZE + 1];   /* select stmt */
    uint4     partmode;                            /* participant mode */
    struct plist      *part_next;                  /* next participant */
} CDR_plist;
```

```
typedef struct freq
{
    uint2       hr;             /* hour 0-32767 */
    uint1       min;            /* minute 0-59 */
    uint1       type;           /* see frequency types below values */
    uint1       day;
    uint1       month;
} CDR_freq;


/* frequency types */
#define  FREQ_IMMED      1    /* Replicate all data immediately. This is the   */
                              /* default value.                                */
#define  FREQ_INTERVAL   2    /* Replicate data at the hour and minute         */
                              /* intervals specified by the hour and minute    */
                              /* members of the structure.                     */
#define  FREQ_TIME       3    /* Replicate data every day at the specified time */
#define  FREQ_DAYOFWEEK  4    /* Replicate data once a week at the time and day */
                              /* specified by the hour, minute, and day members */
                              /* of the structure, where 1=Monday, 2=Tuesday    */
                              /* 3=Wednesday, and so forth.                     */
#define  REQ_DAYOFMONTH  5    /* Replicate data once a month at the time and   */
                              /* day of the month specified by the hour,       */
                              /* minute, and day members of the structure,     */
                              /* where the day member contains a value between */
                              /* 1 and 31 or L designating the last day of the */
                              /* month                                        */


typedef struct conf
{
    uint1   method;                                 /* conflict method */
    char    storedproc[CDR_GL_SPSIZE + 1];          /* stored procedure */
    uint1   storedprocopt;                          /* stored procedure option */
} CDR_conf;


typedef struct repl_attrs
{
    uint4       flags;                  /* replicate attributes */
    time_t      begin;                  /* start time */
    CDR_freq*   freq;                   /* freq tx's sent */
    uint4       threshold;              /* threshold (in bytes) */
    CDR_conf*   primary;                /* primary method*/
    CDR_conf*   secondary;              /* secondary method */
} CDR_repl_attrs;


typedef struct idlist
{
    char    id[CDR_GL_NAMESIZE+1];      /* identifier */
    struct  idlist*  id_next;           /* next identifier */
} CDR_idlist;
```

```
typedef struct q_type
{
    uint1    qtype;                     /*
                                         * Unused field, only present
                                         * for backward compatability
                                         */
    char     dbspace[CDR_GL_NAMESIZE+1]; /* dbspace for Q_STABLE */
} CDR_qtype;


typedef struct serv_attrs
{
    uint2        sflags;                    /* definition flags */
    uint2        idle;                      /* connection time-out (minutes) */
    CDR_qtype*   sendq;                     /* send queue type */
    CDR_qtype*   recvq;                     /* receive queue type */
    char         atsDir[CDR_DIRNAMESIZE+1];    /*  ATS spool dir */
    char         risDir[CDR_DIRNAMESIZE+1];    /*  RIS spool dir */
    char         syncServ[CDR_GL_NAMESIZE+1];  /* catalog sync server name */
} CDR_serv_attrs;
```

Enterprise Replication APIs

The following table includes the API name and the corresponding page number.

(1 of 2)

API	Page
cdr_modify_repl()	12-22
cdr_modify_replmode()	12-23
cdr_modify_serv()	12-24
cdr_modify_servmode()	12-25
cdr_move_errortab()	12-26
cdr_participate()	12-28
cdr_participate_group()	12-27
cdr_participate_repl()	12-28
cdr_prune_errors()	12-29
cdr_prune_single_error()	12-30
cdr_resume_group()	12-31
cdr_resume_repl()	12-32
cdr_resume_serv()	12-33
cdr_start_cdr()	12-34
cdr_start_group()	12-35
cdr_start_repl()	12-36
cdr_stop_cdr()	12-37
cdr_stop_group()	12-38
cdr_stop_repl()	12-39
cdr_suspend_group()	12-40
cdr_suspend_repl()	12-41
cdr_suspend_serv()	12-42

(2 of 2)

cdr_connect()

The **cdr_connect()** function designates the origin for control and configuration.

Important: *If you plan to call programs in addition to Enterprise Replication, you must manage your own connections.*

Syntax

```
#include "cdrapi.h"

int
cdr_connect(server_name)
    char      *server;
```

server is the name of the replication database server.

Usage

Specify the replication database server *server_name* as the origin for all control and configuration calls. A connection is established to the replication database server *server_name* (if `null` defaults to **$INFORMIXSERVER)**. The database server name specified must be defined in the global catalog. For more information, see **cdr_define_serv()**.

Important: *If this call is not invoked prior to making an API call, the connection defaults to the database server specified by the* ***$INFORMIXSERVER*** *environment variable.*

cdr_define_group()

Use the **cdr_define_group()** function to define a replicate group in the global catalog.

Syntax

```
#include "cdrapi.h"

int
cdr_define_group(group, attrs, repls, freq, which)
    char         *group;
    uint4         attrs;
    CDR_idlist   *repls;
    CDR_freq     *freq;
    void         *which;
```

group is the replicate group name.

attrs are the group attributes.

repls is the list of replicate names.

freq are the frequency attributes.

which If the call fails, this pointer references the argument that caused the error (see Appendix A).

Usage

The **cdr_define_group** function defines the replicate group with the replicates *repls*. The replicates specified must be defined over the same set of replication database servers and must all be in the same *state*. Any valid replicate can be defined as part of a replicate group. All replicates must be in the same replication state.

The **attrs** argument specifies attributes for the replicate group. The following attributes specify how the system processes data for the replicate group. Attributes are specified by setting the appropriate flag bit in **attrs**.

Flag Value	Description
A_NONSEQUENTIAL	This option configures Enterprise Replication to process all non-sequential groups in parallel. Use this option only when groups have disjoint data domains. The default is sequential processing of groups.

The *freq* argument specifies the frequency with which data is replicated to database servers that participate in the replicate group. This defaults to the frequency of the participating replicates if the argument is null. The following frequency types can be specified by setting the type member of the frequency structure.

Flag Value	Description
FREQ_IMMED	Replicate all data immediately.
FREQ_INTERVAL	Replicate data at the hour and minute interval specified by the hour and minute members of the structure.
FREQ_TIME	Replicate data every day at the time specified by the hour and minute members of the structure.
FREQ_DAYOFWEEK	Replicate data once a week at the time and day specified by the hour, minute, and day members of a structure where the day member contains a value between 1 and 7 where 1 = Monday, 2 = Tuesday, 3 = Wednesday, and so on.
FREQ_DAYOFMONTH	Replicate data once a month at the time of day of the month specified by the hour, minute, and day members of the structure where the day member contains a value between 1 and 31, or L , designating the last day of the month.

If a replicate group is defined over a set of existing replicates whose frequency type is not immediate, data is replicated for the replicate group with respect to the earliest time that data was replicated for all replicates.

Tip: *Once a replicate is defined as part of a replicate group, control and configuration is only allowed with replicate-group control commands. You cannot control the replicate individually until either the replicate is removed from the replicate group or the replicate group is deleted.*

cdr_define_repl()

The **cdr_define_repl()** function defines a replicate in the global catalog.

Syntax

```
#include "cdrapi.h"

int
define_repl(repl_name, attrs, mflags, participants, which)
    char            *repl_name;
    CDR_repl_attrs  *attrs;
    uint4           mflags;
    CDR_plist       *participants;
    void            *which;
```

repl_name is the name of the replicate to define.

attrs are the replicate attributes.

mflags are the definition flags.

participants are the participating database servers and associated attributes.

which references the argument that caused the error, if the command failed. (See Appendix A.)

Usage

The **cdr_define_repl()** function defines the replicate *repl_name* with attributes *attrs* and participants *participants* in the global catalog.

Attribute Values for CDR_repl_attrs

The *attrs* argument is a pointer to the **CDR_repl_attrs** structure, which specifies attributes for the replicate. These attributes specify how the system manages replication data for the replicate. To specify attributes, set the appropriate flag bit in the **flags** field of the **CDR_repl_attrs** structure and, where applicable, set the appropriate related field.

The arguments of the **CDR_repl_attrs** structure specify the attributes of the replicate, as follows.

Type of Value	Value of flags Field	Description
Conflict resolution option	A_PRIMARY	The structure member ***attrs.primary*** specifies the primary resolution method for the replicate. The member ***attrs.primary.method*** specifies the conflict-resolution method.
Conflict resolution option	A_SECONDARY	The structure member ***attrs.secondary*** specifies the secondary resolution method for the replicate. The member ***attrs.secondary.method*** specifies the type. The member ***attrs.secondary*** is only valid if the primary method selected is time stamp. The only valid bit is STORED_PROCEDURE.
Conflict resolution scope	A_RSCOPE	Activates conflict resolution on a row-by-row basis. This is mutually exclusive with A_TSCOPE.
Conflict resolution scope	A_TSCOPE	Activates conflict resolution on a transaction basis. This is mutually exclusive with A_RSCOPE.
Diagnostic information	A_RIS	Activates row-information spooling for replicate row data that fails conflict resolution or encounters replication-order problems. (For more information, see "Row-Information Spooling" on page 10-9.)
Diagnostic information	A_ATS	Activates aborted-transaction spooling for replicate transactions that fail to be applied to the target database. (For more information, see "Aborted-Transaction Spooling" on page 10-3.)
Data format	A_CANNONICAL	All replication data for the replicate is delivered in machine-independent form. This machine-independent format is required for replicates defined over a heterogeneous collection of database servers where the replication data is of type *float* or *double*.

(1 of 2)

Type of Value	Value of flags Field	Description
Trigger activation	A_TRIGGER	Triggers should be fired for this replicate. The default fires no triggers for replicated updates.
	A_BEGIN	Activates the replicate at the time specified by ***attrs.begin***. The replicate becomes active on all database servers participating in the replicate. The time specified is in local time, specified by the **$TZ** environment variable. If the **$TZ** environment variable is not set, then GMT is assumed. All times are stored in the global catalog in GMT. If the A_BEGIN flag bit is not set the replicate must be activated by **cdr_start_repl()** or the begin time must be modified with **cdr_modify_repl()**.
	A_FREQ	Frequency replication data is distributed to active participating database servers as specified by **attrs.freq**. See the A_BEGIN flag bit for discussion of time zones. The default is to distribute replication data immediately.

(2 of 2)

If the **R_PARTICIPANTS** flag bit is set in **mflags**, the system expects the **participants** argument to be present. This argument specifies the *replicate participants* for the replicate. Participant information is stored in the **CDR_plist** structure. The **selectstmt** member of that structure specifies the criteria for replication. The following requirements are necessary for the statement:

- The SELECT clause can only reference columns from the table being replicated.
- The SELECT clause must contain at least the primary key.
- The SELECT clause cannot contain aggregates or constants.

Attribute Values for CDR_conf

The arguments of the **CDR_conf** structure specify the conflict resolution method, as follows.

Type of Value	Value of flags Field	Description
Resolution type	R_IGNORE	Updates are applied as received; no update collisions are detected or processed. Replicates using this type of resolution do not use any special Enterprise Replication columns. This value is mutually exclusive with R_TIMESTAMP and R_STOREDPROC.
Resolution type	R_TIMESTAMP	The replication time stamp determines which update will prevail. This is mutually exclusive with R_IGNORE and R_TIMESTAMP.
Resolution type	R_STOREDPROC	A stored procedure specified by the structure member ***attrs.primary.storedproc*** is used to resolve the conflict. If you set this flag, you must allocate a **CDR_conf** structure and set the **primary** field. This value is mutually exclusive with R_IGNORE and R_TIMESTAMP. If you set R_STOREDPROC, you can set the stored-procedure behavior to R_STOREDPROC_OPTM or R_STOREDPROC_EXEC.
Stored procedure behavior	R_STOREDPROC_OPTM	Optimized mode: the replicated row is accepted and the stored procedure is not calle, even when conflict is detected, if the replicated row is from the same database server that last updated the local row on the target table, and the replicated row has a time stamp greater than or equal to the local row. This value is mutually exclusive with R_STOREDPROC_EXEC.
Stored procedure behavior	R_STOREDPROC_EXEC	Execute mode: the stored procedure defined for the replicate is always called when conflict is detected. This value is mutually exclusive with R_STOREDPROC_OPTM.

cdr_define_serv()

Use the **cdr_define_serv()** function to define a database server in the global catalog.

Syntax

```
#include "cdrapi.h"

int
cdr_define_serv(server,attrs)
    char              *server;
    CDR_serv_attrs    *attrs;
```

server is the database server name.

attrs are the database server attributes.

Usage

The **cdr_define_serv** function defines the database server *server* in the global catalog with the attributes *attrs*. The database server name specified must be defined in the **$INFORMIXDIR/etc/sqlhosts** file or the SQLHOSTS registry key.

Important: *A database server must be defined in the global catalog prior to participation in replication.*

The *attrs* argument specifies attributes for the database server. To specify attributes, set the appropriate flag bit in **attrs.sflags** and use the supplied structure member:

A_INIT	Enterprise Replication is initialized on the database server that is defined.
A_SYNC	Use **attrs.syncServ** as the database server for catalog synchronization when the database server is initialized. This option is only available when the A_INIT flag has been set.
A_IDLE	All database servers maintain a separate connection to all other database servers participating in replication exclusively for the transmission of replication data. Because the connection consumes system resources, an idle time-out provides a way to optimize infrequent connections. The connection will be automatically re-established when data is queued for the database server. The idle time-out is specified in minutes by **attrs.timeout**.
A_SENDQ	Use **attrs.sendq** for the send-queue dbspace. The send queue is assigned to a particular dbspace.
A_RECVQ	Use **attrs.recvq** for the receive-queue dbspace. The receive queues is assigned to a particular dbspace.
A_ATS	Use **attrs.ats_dir** as the directory for ATS files. If ATS is specified for a replicate and this attribute was not set, the **/tmp** directory is used. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.")
A_RIS	Use **attrs.ris_dir** as the directory for RIS files. If RIS is specified for a replicate and this attribute was not set, the **/tmp** directory will be used. (For more information, see Chapter 10, "Diagnosing Enterprise Replication.")

cdr_delete_group()

Us the **cdr_delete_group()** function to delete a replicate group.

Syntax

```
#include "cdrapi.h"

int
cdr_delete_group(group, time)
    char     *group;
    time_t   time;
```

group is the replicate group name.

time is the time at which to delete the replicate group.

Usage

The **cdr_delete_group** function deletes the replicate group *group* from the global catalog at the time *time* (zero for immediately).

Tip: *The **cdr_delete_group** command does not affect the replicates or associated data. When a replicate group is deleted, the state of individual replicates within the group is unchanged.*

cdr_delete_repl()

Use the **cdr_delete_repl()** function to delete a replicate from the global catalog.

Syntax

```
#include "cdrapi.h"

int
cdr_delete_repl(repl, time)
    char      *repl;
    time_t    time;          /* unused */
```

repl is the name of the replicate to delete.

time is the optional time at which to delete the replicate.

Usage

The **cdr_delete_repl** function deletes the replicate *repl* from the global catalog at time *time* (zero for immediately). All replication data for the replicate is purged from the send queue at all participating database servers. If *repl* is contained in a replicate group the call will fail. You must remove the replicate from the group before it can be deleted.

cdr_delete_serv()

Use the **cdr_delete_serv()** function to delete a database server from the global catalog.

Syntax

```
#include "cdrapi.h"

int
cdr_delete_serv(server)
    char    *server;
```

server is the database server to delete.

Usage

The database server specified is deleted from the global catalog. The database server is removed from all participating replicates. All replication data is purged from the send queues for the database server specified. Enterprise Replication is shut down and the catalogs are removed on the replicating database server *server*.

When you delete a server, you must issue the **cdr_delete_serv()** command *twice*. The first **cdr_delete_serv()** removes the server on the local host and removes the ER connection to other hosts. The second **cdr_delete_serv()** removes the server from all of the other ER hosts.

cdr_error_reviewed()

Use the **cdr_error_reviewed()** function to mark a specified error as reviewed.

Syntax

```
#include "cdrapi.h"

int
cdr_error_reviewed (orig_server, remote_seqnum)
    char       *orig_server;
    long       remote_seqnum;
```

orig_server is the database server name from which the error originated.

remote_seqnum is the remote sequence-number value for the error that is to be updated. Combined with the originating database server name, *remote_seqnum* provides a unique key for every error.

Usage

The **cdr_error_reviewed** function marks an error in the severe-error table as having been *reviewed*. When an error is reviewed it means that someone has resolved the reason for a given error. The review state is used only to track and determine which errors to delete with **cdr_prune_errors()**. The originating database server name and the sequence number from that database server are used to uniquely specify the error to be marked. The specified error is updated on all database servers participating in replication.

cdr_modify_group()

Use the **cdr_modify_group()** function to modify group attributes.

Syntax

```
#include "cdrapi.h"

int
cdr_modify_group(group, attrs, freq)
    char        *group;
    uint4       attrs;
    CDR_freq    *freq;
```

group is the replicate group name.

attrs are the replicate group attributes.

freq are the frequency attributes.

Usage

The **cdr_modify_group** function modifies the group *group* with attributes *attrs*. The *attrs* argument specifies which attributes to modify. For a description of the attributes, see "cdr_define_group()" on page 12-8. The *freq* argument can be null; in such cases the frequency attributes of the replicate group are not modified.

cdr_modify_repl()

Use the **cdr_modify_repl()** function to modify replicate attributes.

Syntax

```
#include "cdrapi.h"

int
cdr_modify_repl(repl,freq)
    char              *repl;
    CDR_repl_attrs    *attrs;
```

repl is the replicate name.

freq are the frequency attributes.

Usage

The **cdr_modify_repl** function modifies the replicate *repl* with the frequency attributes *freq*. The *freq* argument specifies the new frequency for the replicate. For a description of attributes, see "cdr_delete_repl()" on page 12-18.

Attribute Values for CDR_repl_attrs

In addition to the attributes specified in cdr_define_rel, this API uses the following values.

Type of Value	Value of flags Field	Description
Transaction spooling	A_NRIS	Deactive row-information spooling for this replicate.
Transaction spooling	A_NATS	Deactivate aborted-transaction spooling pooling for this replicate.
Triggers	A_NTRIGGER	Triggers should not be fired for this replicate.

cdr_modify_replmode()

Use the **cdr_modify_replmode()** function to change the mode of participants in a replicate.

Syntax

```
#include "cdrapi.h"

int
cdr_modify_replmode(repl, participants, which)
    char        *repl;
    CDR_plist   *participants
    void        *which;
```

repl is the replicate name.

participants is the list of participants.

which If the call failed, this pointer references the argument that caused the error.

Usage

The **cdr_modify_replmode** function modifies the modes of replicate participant *participants* in replicate *repl*. The *partmode* field of each participant structure is used to set the mode of the participant.

If the value of *partmode* is A_RDONLYMODE, then the participant is set to a read-only state, where updates made to the replicated table on the database server for this participant are not replicated, and the snooping and grouping phases of Enterprise Replication are not notified of the participant.

If the value of *partmode* is A_PRIMARYMODE, then the participant is set to a primary state, where updates made to the replicated table on the participant database server are replicated. This is the default mode.

cdr_modify_serv()

Use the **cdr_modify_serv()** function to modify database server attributes.

Syntax

```
#include "cdrapi.h"

int
cdr_modify_serv(server,attrs)
    char      *server;
    CDR_serv_attrs  *attrs;
```

server is the database server name.

attrs are the database server attributes to modify.

Usage

The **cdr_modify_serv()** function modifies the replication database server *server* with attributes *attrs*. The same flag bits and options are supported as for **cdr_define_serv()**, except that A_SENDQ is not modifiable.

cdr_modify_servmode()

Use the **cdr_modify_servmode()** funtion to change the *mode* of participants for a database server.

Syntax

```
#include "cdrapi.h"

int
cdr_modify_servmode(server, mode)
    char      *server;
    uint4     mode;
```

server is the database server name.

mode is the mode to use for all of the participants for this database server.

Usage

The **cdr_modify_servmode()** funtion changes the modes of replicate participants for the given database server. The *mode* argument determines the mode for all participants of the database server. The A_RDONLYMODE mode sets the participants to read-only mode. The A_PRIMARYMODE mode sets the participants to primary mode.

Participants already in the designated mode are unaffected.

For A_RDONLYMODE, participants of replicates with conflict resolution that are not set to ignore are skipped. For detailed information on mode values, see "cdr_modify_replmode()" on page 12-23.

cdr_move_errortab()

Use the **cdr_move_errortab()** function to move the error table to another dbspace.

Syntax

```
#include "cdrapi.h"

int
cdr_move_errortab(dbspace, performonall)
    char      *dbspace;
    int       performonall;
```

dbspace is the name of the dbspace in which the error table is to be moved.

performonall is the flag that indicates whether this operation should be performed on all database servers or only the database server to which the user is connected.

Usage

The **cdr_move_errortab()** function moves the severe-error table to a new dbspace. All errors that are currently stored in the table are moved to the new dbspace. If the **performonall** flag is non-zero, this operation is performed on all database servers involved in replication. Otherwise, it only performs on the database server to which the user is connected.

cdr_participate_group()

Use the **cdr_participate_group()** function to add or delete replicates to or from an existing replicate group.

Syntax

```
#include "cdrapi.h"

int
cdr_participate_group(group, op, repls, which)
    char        *group;
    uint1       op;
    CDR_idlist  *repls;
    void        *which;
```

group is the name of the replicate group.

op is the operation A_ADD or A_REMOVE.

repls is the list of replicates.

which If the call fails, this pointer references the argument that caused the error (see Appendix A).

Usage

The **cdr_participate_group()** function adds or removes (determined by the operator *op*) replicates *repls* from replicate group *group*. If the last participant is deleted from the replicate *group*, the replicate group is removed from the global catalog. When a replicate is removed from a replicate group, the replicate state remains the same as the replicate group from which it was removed.

cdr_participate_repl()

Use the **cdr_participate_repl()** function to add or remove participants from a replicate.

Syntax

```
#include "cdrapi.h"

int
cdr_participate_repl(repl, participants, op, which)
    char          *repl;
    uint1         op;
    CDR_plist     *participants;
    void          *which;
```

repl is the replicate name.

participants is the list of participants.

op is the operation type: **A_ADD** or **A_REMOVE**.

which If the call failed this pointer references the argument that caused the error (see Appendix A).

Usage

The **cdr_participate_repl()** function adds or removes (determined by operator *op*) replicate participants *participants* from replicate *repl*.

If the operation is type **A_ADD**, participants *participants* are added to the replicate. The replication state is initialized to *inactive*.

If the operation is type **A_REMOVE,** participants *participants* are removed from the replicate. The *selectstmt* member of the **CDR_plist** structure is not required and will be ignored if it is present.

cdr_prune_errors()

Use the **cdr_prune_errors()** function to prune errors from the severe-error table based on time or time range.

Syntax

```
#include "cdrapi.h"
#include "datetime.h"
#include "decimal.h"

int
cdr_prune_errors(uppertime, lowertime, reviewed_only)
    dtime_t         *uppertime;
    dtime_t         *lowertime;
    int             reviewed_only;
```

uppertime is the upper (most recent) time limit for deletion of errors. This value *must* be specified.

lowertime is the lower (earliest) time limit for deletion of errors. If a null is passed, all errors earlier than *uppertime* are deleted.

reviewed_only is the flag to specify whether all errors should be deleted or only those errors that have been reviewed. Any non-zero value causes only errors that have been reviewed to be deleted.

Usage

The **cdr_prune_errors()** function deletes errors from the severe error table on all database servers involved in replication for a time range, or for all errors before a specified time. If only one time value is passed in, all errors that occurred at or before that time are deleted. If both time values are specified, then errors that occurred within the time range are deleted. If the **reviewed_only** flag is set, only errors that match the specified time criteria *and* that have been reviewed are deleted.

cdr_prune_single_error()

Use the **cdr_prune_single_error()** function to prune a single error from the severe-error table.

Syntax

```
#include "cdrapi.h"

int
cdr_prune_single_error(origserver, remote_seqnum)
    char   *origserver;
    long   remote_seqnum;
```

origserver is the database server name from which the error originated.

remote_seqnum is the remote sequence number value for the error that is to be deleted. Combined with the originating database server name, this provides a unique key for every error.

Usage

The **cdr_prune_single_error** function deletes a single error from the severe-error table. It uses the originating severe name and the sequence number from that database server to uniquely specify the error to be deleted. The specified error is deleted from all database servers that participate in replication.

cdr_resume_group()

Use the **cdr_resume_group()** function to resume delivery of replication data for replicates defined in a replicate group.

Syntax

```
#include "cdrapi.h"

int
cdr_resume_group(group,time)
    char     *group;
    time_t   time;       /* unused */
```

group is the replicate group name.

Usage

The replicates contained in the replicate group *group* enter the active state (capture, send) for all participants.

cdr_resume_repl()

Use the **cdr_resume_repl()** function to resume delivery of replication data.

Syntax

```
#include "cdrapi.h"

int
cdr_resume_repl(repl, time)
    char     *repl;
    time_t   time;        /* unused *
```

repl is the replicate name.

time is the optional time at which to resume.

Usage

The replicate *repl* enters the active state (capture, send) at the time *time* (zero for immediately) for all participants. If *repl* is contained in a *replicate group*, the call will fail. (See "cdr_resume_group()" on page 12-31.)

cdr_resume_serv()

Use the **cdr_resume_serv()** function to resume delivery of replication data to a database server.

Syntax

```
#include “cdrapi.h”

int
cdr_resume_serv(db_server, servers, time, which)
    char          *db_server;
    CDR_idlist    *servers;
    time_t        time;        /* unused *
    void          *which;
```

db_server is the database server name.

servers is the optional list of database servers.

time is the time to resume.

which If the call fails, this pointer references the argument that caused the error (see Appendix A).

Usage

Delivery of replication data resumes (including queued data) for the database server *db_server* on the replicating database servers *servers* (null for all database servers) at the time *time* (zero for immediately).

cdr_start_cdr()

Use the **cdr_start_cdr()** function to start Enterprise Replication processing.

Syntax

```
#include "cdrapi.h"

int
cdr_start_cdr()
```

Usage

Use the **cdr_start_cdr()** function to restart Enterprise Replication after it is stopped with **cdr_stop_cdr()**. When you issue **cdr_start_cdr()**, Enterprise Replication threads start and continue from the point at which they stopped.

Enterprise Replication resumes evaluation of the logical log (if required for the instance of Enterprise Replication) at the *replay* position. The replay position is the position where Enterprise Replication stops evaluating the logical log when **cdr_stop_cdr()** is executed. If the evaluation process is running and the logical log ID for the replay position no longer exists when Enterprise Replication is started, then the restart partially fails (the database server log contains an error message stating that the replay position is invalid). If the restart partially fails, no database updates performed on the local database server are replicated. For this reason, Informix recommends that you issue **cdr_start_cdr**() and **cdr_stop_cdr**() with extreme caution. Use these commands when little or no database activity occurs on the database server, and keep Enterprise Replication downtime to a minimum.

cdr_start_group()

Use the **cdr_start_group()** function to begin capture and transmittal of replication transactions for replicates contained in a replicate group.

Syntax

```
#include "cdrapi.h"

int
cdr_start_group(group, servers, time, which)
    char          *group;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

group is the replicate group name.

servers is the optional list of database servers.

time is the optional time at which to activate the replicate group.

which If the call fails this pointer references the argument that caused the error (see Appendix A).

Usage

The replicates defined in the group *group* enter the active state (capture-send) on the database servers *servers* or (if database servers is null) all database servers at the time *time* (immediately if time is zero).

cdr_start_repl()

Use the **cdr_start_repl()** function to begin to capture and transmit replication transactions.

Syntax

```
#include "cdrapi.h"

int
cdr_start_repl(repl, servers, time, which)
    char          *repl;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

repl is the replicate name.

servers is the optional list of database servers.

time is the optional time at which to activate the replicate.

which If the call fails, this pointer references the argument that caused the error (see Appendix A).

Usage

The replicate *repl* enters the active state (data is captured and sent) on the replicate database servers *servers* or (if *servers* is null) all database servers at the time specified (immediately if time is **zero**). To enter the active state:

- synchronize active participants.
- begin to capture data for replication on the participants specified.

If **repl** is contained in a *replicate group*, the call will fail. For operation on groups, see "cdr_start_group()" on page 12-35.

cdr_stop_cdr()

Use the **cdr_stop_cdr()** function to stop Enterprise Replication processing.

Syntax

```
#include "cdrapi.h"

int
cdr_stop_cdr()
```

Usage

In most situations, Enterprise Replication starts when **cdr_define_serv()** is first executed from a database server. The Enterprise Replication threads remain running until the database server is shut down, or until the local database server is deleted with the **cdr_delete_serv()** call. If the database server is shut down with Enterprise Replication running, Enterprise Replication restarts when the database server is restarted.

Under rare conditions, users may want to temporarily stop the Enterprise Replication threads without stopping the database server. If **cdr_stop_cdr()** is called, all Enterprise Replication threads shut down in an orderly manner. When the shutdown of Enterprise Replication is complete, you see the message `Enterprise Replication: shutdown complete` in the database server log file. After issuing the stop Enterprise Replication request, Enterprise Replication threads remain stopped (even if the database server is stopped and restarted) until a start Enterprise Replication request (**cdr_start_cdr()**) is issued.

Warning: *If you issue **cdr_stop_cdr()** and database activity continues, the database server from which the command is issued and the other database servers participating in replicates might become inconsistent. To ensure consistency, verify that no database update activity occurs while Enterprise Replication is stopped.*

cdr_stop_group()

Use the **cdr_stop_group()** function to stop capture and transmittal transactions for replicates contained in a replicate group.

Syntax

```
#include "cdrapi.h"

int
cdr_stop_group(group, servers, time, which)
    char          *group;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

group is the replicate group name.

servers is the optional list of database servers.

time is the time to stop.

which If the call fails, this pointer references the argument that caused the error (see Appendix A).

Usage

All replicate data defined in the group *group* enters the *inactive* state *(*no capture, no send*)* on the database servers *servers* (null for all database servers participating in the replicate group).

cdr_stop_repl()

Use the **cdr_stop_repl()** function to stop the capture and transmittal of transactions for replication.

Syntax

```
#include "cdrapi.h"

int
cdr_stop_repl(repl, servers, time, which)
    char          *repl;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

repl is the replicate name.

servers is the optional list of database servers.

time is the time to stop.

which If the call fails, this pointer references the argument that caused the error (see Appendix A).

Usage

The replicate *repl* enters the inactive state (no capture, no send) on the replicate database servers *servers* (null for all database servers) at the time *time* (zero for immediately).

To enter the inactive state (on the database servers specified) stop the capture of replication data.

If **repl** is contained in a *replicate group*, the call will fail, (see "cdr_stop_group()" on page 12-38).

cdr_suspend_group()

Use the **cdr_suspend_group()** function to suspend delivery of replication data for replicates contained in a replicate group.

Syntax

```
#include "cdrapi.h"

int
cdr_suspend_group(group)
    char     *group;
    time_t   time;
```

group is the replicate group name.

Usage

The replicates contained in the replicate group *group* enter the *suspend* state (capture, no send) for all participants.

cdr_suspend_repl()

Use the **cdr_suspend_repl()** function to suspend delivery of replication data.

Syntax

```
#include "cdrapi.h"

int
cdr_suspend_repl(repl, time)
    char      *repl;
    time_t    time;
```

repl is the replicate name.

time is the optional time at which to suspend.

Usage

The replicate *repl* enters the *suspend* state (capture, no send) at the time *time* (zero for immediately) for all participants. If *repl* is contained in a *replicate group* the call will fail. (See "cdr_suspend_group()" on page 12-40.)

cdr_suspend_serv()

Use the **cdr_suspend_serv()** function to suspend the delivery of replication data to a database server.

Syntax

```
#include "cdrapi.h"

int
cdr_suspend_serv(server, servers, time, which)
    char          *server;
    CDR_idlist    *servers;
    time_t        time;
    void          *which;
```

server is the database server name to suspend.

servers is the optional list of database servers.

time is the optional time to suspend.

which If the call failed this pointer references the argument that caused the error (see Appendix A).

Usage

The **cdr_suspend_serv** function suspends delivery of replication data to the replicating database server *server* on database servers *servers* (null for all database servers) at the time *time* (zero for immediately). Replication data continues to all other database servers.

Tip: *Unlike **cdr_suspend_repl()**, the replicate and/or replicate group status is irrelevant.*

Appendixes

Section

Appendix

A

Return Codes

Definition of Return Codes

The following list shows the return messages and codes that are returned by the command-line utility and by the API functions that are documented in Chapters 11 and 12.

These are also documented in the **cdrerr.h** file in **$INFORMIXDIR/incl/esql** directory.

List of Return Values

Mnemonic	Numeric Value	Description
CDR_SUCCESS	0	Command was successful
CDR_ENOCONNECT	1	No connection for the specified server
CDR_EBUF	2	(Reserved for future use)
CDR_ECOLUNDEF	3	Table column undefined
CDR_ECOMPAT	4	Incompatible server version
CDR_ECONNECT	5	Unable to connect to server specified
CDR_EDBDNE	6	Database does not exist The **which** argument contains a pointer to a **CDR_plist** structure specifying the database server and database.
CDR_EDBLOG	7	Database not logged The **which** argument contains a pointer to a **CDR_plist** structure specifying the database server and database.
CDR_EFREQ	8	Bad or mismatched frequency attributes. The frequency type specified is incorrect or the values are out of range for the frequency type specified.
CDR_ECONNECTED	9	Already connected to specified server
CDR_EGRPSTATE	10	Illegal group state change
CDR_EGRPUNDEF	11	Undefined group
CDR_EGRPUNIQ	12	Group name already in use
CDR_EIDLE	13	Invalid idle time
CDR_EINVOP	14	Invalid operator or specifier
CDR_ELEN	15	(Reserved for future use)
CDR_ELOGFILE	16	(Reserved for future use)

(1 of 6)

Mnemonic	Numeric Value	Description
CDR_ENOPART	17	Participants required for operation specified
CDR_ENOPKEY	18	Table does not contain primary key
CDR_EOWNER	19	Table does not exist No namespace in database for owner specified. The **which** argument contains a pointer to a **CDR_plist** structure specifying the database
CDR_EPACTIVE	20	Server already participating in replicate
CDR_EPDEF	21	(Reserved for future use)
CDR_EPKEYSELCT	22	Primary key not contained in select clause
CDR_EPRIKEY	23	(Reserved for future use)
CDR_ERECVQ	24	Dbspace for receive queue does not exist
CDR_EREPACTIVE	25	Replicate already participating in a group
CDR_EREPDEF	26	Group operation not permitted on replicate
CDR_EREPLSYN	27	(Reserved for future use)
CDR_EREPLUNIQ	28	Replicate name already in use
CDR_ETBLDNE	29	Table does not exist Table does not exist. The **which** argument contains a pointer to a **CDR_plist** structure specifying the database server and database.
CDR_EREPSTATE	30	Illegal replicate state change
CDR_EREPUNDEF	31	Undefined replicate
CDR_ESENDQ	32	dbspace specified for the send queue does not exist
CDR_ESERVDEF	33	Server not participant in replicate / group

Mnemonic	Numeric Value	Description
CDR_ESERVMAX	34	Maximum number servers exceeded
CDR_ESERVRESOLV	35	Server not defined in sqlhosts
CDR_ESERVSET	36	Disjoint servers for replicates
CDR_ESERVUNDEF	37	Undefined server The **which** argument contains a pointer to a **CDR_plist** structure specifying the undefined database server.
CDR_ESPDNE	38	Stored procedure does not exist Undefined stored procedure. The **which** argument contains a pointer to a **CDR_conf** structure that specifies the conflict-resolution method and stored procedure name.
CDR_ESQLSYN	39	Illegal select syntax SELECT statement uses illegal syntax. The **which** argument contains a pointer to a **CDR_plist** structure specifying the database server and SELECT statement.
CDR_ESQLUNSUP	40	Unsupported SQL syntax (join, and so on)
CDR_ETHOLD	41	(Reserved for future use)
CDR_ETIME	42	Invalid time
CDR_EVALID	43	Participants required for specified operation The **R_PARTICIPANT** flag bit is set in **mflags** but the **participants** argument is null.

(3 of 6)

Mnemonic	Numeric Value	Description
CDR_ENAMERR	44	Illegal name syntax Invalid replicate or database server name. In the case of an invalid database server name the **which** argument will contain a pointer to a type **CDR_plist** specifying the database server.
CDR_EPART	45	Invalid participant
CDR_EREPL	46	(Reserved for future use)
CDR_ESERV	47	Invalid server name
CDR_ENOMEM	48	Out of memory
CDR_EREPMAX	49	Maximum number of replicates exceeded
CDR_EPARTMAX	50	Maximum participants
CDR_ERUMOR	51	Attempt to delete server remotely
CDR_ESERVUNIQ	52	Server name already in use
CDR_EDUPL	53	Duplicate server or replicate
CDR_EBADCRULE	54	Bad conflict rule specified
CDR_ENOSCOPE	55	Resolution scope not specified
CDR_ESCOLSDNE	56	Shadow columns do not exist for table Shadow columns are not present or have invalid data types. If non-ignore conflict resolution is chosen, shadow columns must exist for replicated table.
CDR_ECRDELTAB	57	Error creating delete table
CDR_ENOCRULE	58	No conflict resolution rule specified
CDR_EBADSCOLS	59	Table has bad type for shadow columns or has shadow columns at wrong place
CDR_EGRPPART	60	Illegal operation on group participant

(4 of 6)

Mnemonic	Numeric Value	Description
CDR_ENOPERM	61	User doesn't have permission to issue command
CDR_ENOCDR	62	CDR not active
CDR_ECDR	63	CDR already active
CDR_ENOSYNC	64	Remote / cyclic synchronization not allowed
CDR_ESERVID	65	Server identifier already in use
CDR_ENOTIME	66	No upper time for prune error
CDR_ERRNOTFOUND	67	Error not found for delete or update
CDR_EPARTMODE	68	Illegal participant mode
CDR_ECONFLICT	69	Conflict mode for replicate not ignore
CDR_ECONSAME	70	Connect / disconnect to / from same server
CDR_EROOT	71	Conflicting root server flags
CDR_ESPORADIC	72	Conflicting sporadic server flags
CDR_ESPOROOT	73	Sporadic-root configuration not allowed
CDR_EPARENT	74	Cannot delete server with children
CDR_ESPAROOT	75	Sparse-root configuration not allowed
CDR_ESPARSE	76	Conflicting sparse server flags
CDR_ELIMITED	77	Request denied on limited server
CDR_EMSGFORMAT	78	Unsupported message format
CDR_EDROPDB	79	Couldn't drop syscdr database (Error -425)
CDR_EATSDIR	80	ATS directory does not exist
CDR_ERISDIR	81	RIS directory does not exist

(5 of 6)

Mnemonic	Numeric Value	Description
CDR_ECRCHANGE	82	Illegal conflict-resolution change
CDR_USAGE	99	Incorrect use of command-line utility
CDR_ESERVERR	100	Fatal server error
CDR_ENOSUPPORT	101	Feature not available

(6 of 6)

Appendix

B

Fine-Tuning Enterprise Replication

This appendix provides reference material and several examples to help you fine-tune Enterprise Replication.

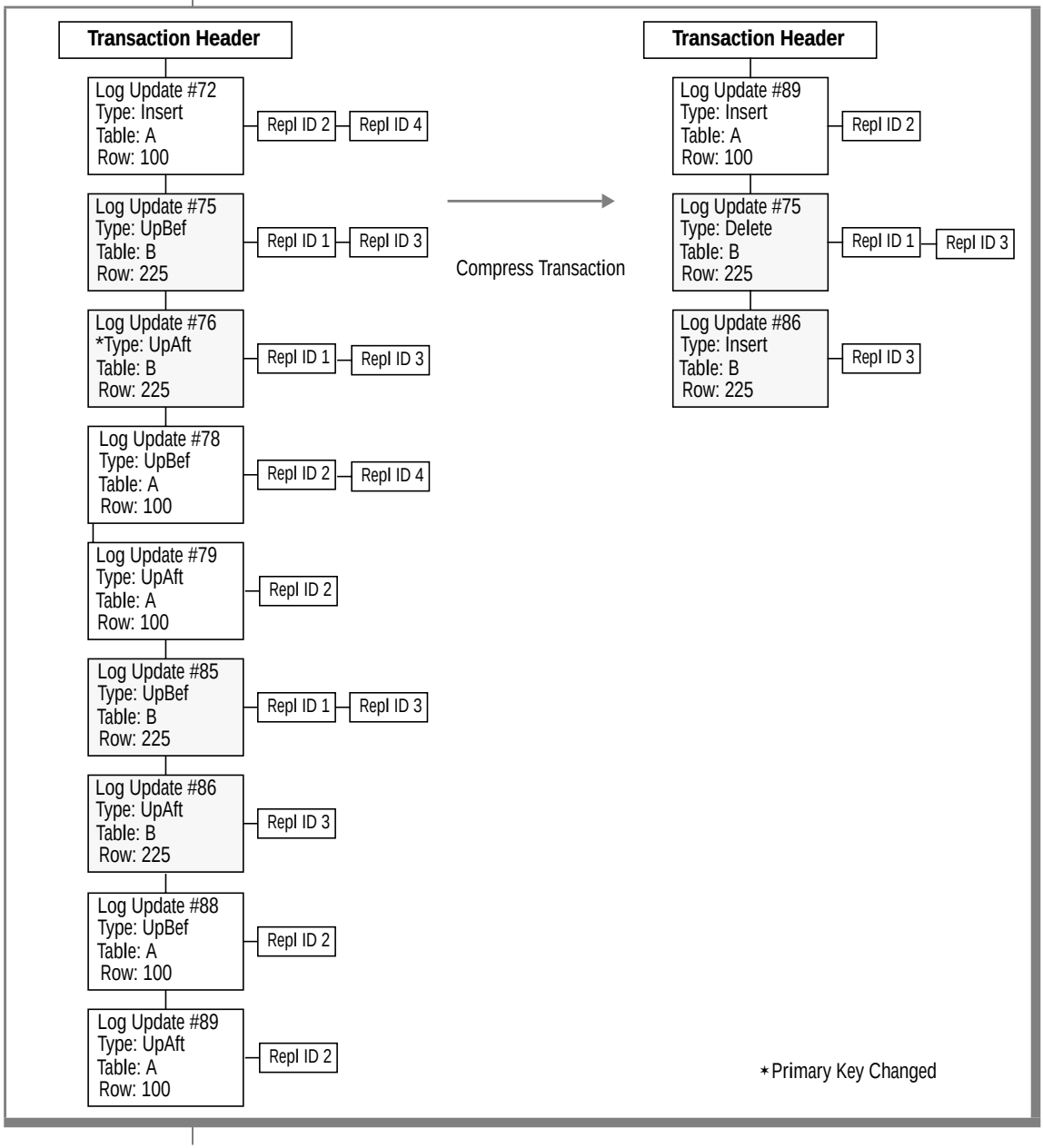
Evaluating Images in a Complex Transaction

Figure B-1 on page B-2 illustrates the evaluation compression of a single transaction with the following activities:

- One row is inserted and then updated several times.
- Another row in a different table is updated twice and the primary key changes in the second update.

Each row is evaluated against the appropriate replicates, and a list of replicates is assembled where the row has been evaluated as a yes.

Figure B-1
How Enterprise Replication Evaluates Images in a Complex Transaction



A Stored Procedure Example

This example illustrates the function of conflict resolution by stored procedure. The stored procedure will resolve a conflict for rows in a table defined as:

```
CREATE TABLE employees
        (
        emp_num int,
        empname CHAR(15),
        salary INT,
        PRIMARY KEY (emp_num)
       ) WITH CRCOLS LOCK MODE ROW;
```

The replicate is defined as `SELECT * FROM employees`. The `emp_num` is the primary key. The primary key is passed along with the other replicated row data and the local target row data.

The procedure determines the type of collision. If a replication order error is detected, the replicated row is not applied, but is logged. Otherwise, if the replicated salary field is not equal to the local target, the replicated row is applied, but is not logged.

```
create procedure updateSalary
(localServer CHAR(18), localTS DATETIME YEAR TO SECOND,
localDelete CHAR(1),
 replServer CHAR(18), replTS DATETIME YEAR TO SECOND, replDbop
CHAR(1),
 localEmpNum INT, localEmpName CHAR(15), localSalary INT,
 replEmpNum  INT, replEmpName  CHAR(15), replSalary  INT)
returning CHAR(1), INT, INT, CHAR(15), INT;

{-------------------------------------------------------}
{ Do nothing if replicated row is out of logical order:
     (1) if local row is found in the "deleted" row table
                 and
                 the replicated operation is update or delete
     (2) if local row is found in target table and the
                 replicated operation is insert.
        Enter the row to log.
}

if (localDelete = 'Y') then
    if (replDbop = 'U') or (replDbop ='D')
        then
            return 'O', 100, null, null, null;
    end if
if (replDbop = 'I') then
    return 'O', 101, null, null, null;
end if

{-------------------------------------------------------}
{ If the replicated row is from server "corporate" and the
  the local row's salary field is not equal to the
    replicated salary field, then update local row.
}

if (replServer = 'corporate' and localSalary != replSalary)
then
        return replDbop, 200, replEmpNum, replEmpName,
replSalary;
else
        return replDbop, 201, replEmpNum, replEmpName,
            localSalary;
end if;

end procedure
```

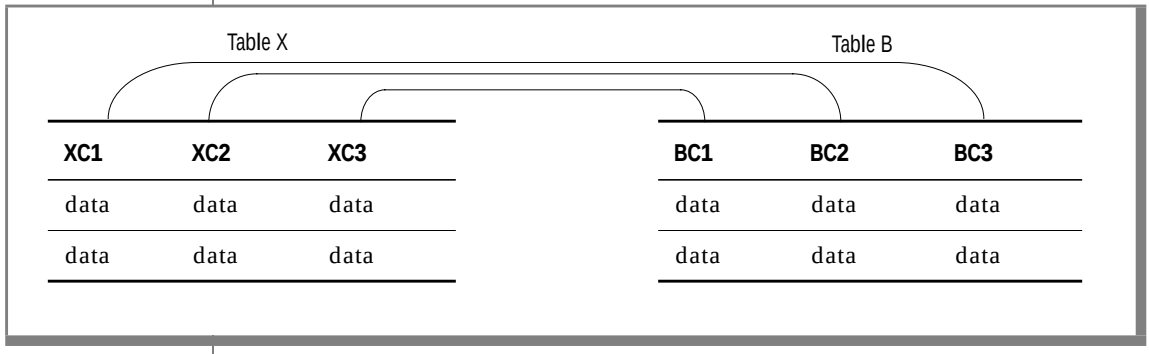
Column Mapping

You can reorder columns between replicated tables. Each participant in the replication system must define the identical number of columns (n to n mappings only). These columns must define the same domains. No data conversions are performed by column mapping.

Column mapping is defined by the enumerations of columns in the select statement that accompanies each participant in a replication system.

For example in Figure B-2, **Table X** with columns **XC1**, **XC2**, and **XC3** maps to columns **BC1**, **BC2**, and **BC3** in **Table B**.

Figure B-2
Column Mapping



The columns map as follows: **XC1** to **BC3**, **XC2** to **BC2**, and **XC3** to **BC1**. The SELECT statements would then read:

```
SELECT XC1, XC2, XC3 FROM Table X
SELECT BC3, BC2, BC1 FROM Table B.
```

Step-by-Step Procedure

Replication is defined and controlled with a few simple steps. The following example shows three sites in the replication system:

- Bangor, Maine
- Ames, Iowa
- Milpitas, California

Each of the sites have a database named **general**. Each site has equivalent **sqlhosts** files (UNIX) or SQLHOSTS registry key entries (Windows NT) that identify each of the database servers and server groups. The table **employees** is to be replicated between all three sites.

The database **general** has the following schema:

```
CREATE DATABASE general WITH LOG; {or OPEN database general,
if already created}
CREATE TABLE employees
        (
        emp_num     INT,
        empname     CHAR(15),
        company     CHAR(20),
        address1    CHAR(20),
        address2    CHAR(20),
        city        CHAR(15),
        state       CHAR(2),
        zipcode     CHAR(5),
        phone       CHAR(18),
        PRIMARY KEY (emp_num)
      ) WITH CRCOLS LOCK MODE ROW;
```

Step 1: Define Database Schema for Maine Site

On the database server in Bangor, Maine, create database **general**.

Step 2: Define Database Schema for Iowa Site

On the database server in Ames, Iowa, create database **general**.

Step 3: Define Database Schema for California Site

On the database server in Milpitas, California, create database **general**.

Step 4: Declare Database Servers to Enterprise Replication

Declare all three database servers (**bangor**, **ames**, and **milpitas**) to Enterprise Replication. This is required because none of the database servers have been previously declared. You can declare the database servers using any of the following methods:

- Using the API call **cdr_define_serv()**
- Using the CLU command **cdr define server**
- Using the IECC graphical user interface

After the database servers are declared to Enterprise Replication, each database server has a database named **syscdr** that contains the global catalog.

Step 5: Define a Replicate Called employeeinfo

Next the replication system is defined using the API call **cdr_define_repl()**. The replication system will be defined with the name **employeeinfo** with participant database servers **bangor**, **ames**, and **milpitas** as follows:

```
general@bangor:employees "SELECT * FROM employees"
general@milpitas:employees "SELECT * FROM employees"
general@ames:employees "SELECT * FROM employees"
with resolution rules set to timestamp primary
```

The replication system is defined in the global catalog with an initial state of *inactive*. This means that no replication data is captured or transmitted.

Step 6: Prepare the Tables for Replication

The database administrator must ensure that the data in the tables is consistent across all sites before starting replication.

This example assumes that **bangor** contains the complete correct version of the table. The following list shows actions necessary to prepare for replication:

- Lock the tables on **bangor**, **ames**, **milpitas** with a shared lock.
- Unload the authoritative copy of the table from **bangor**.
- Load new copies into **ames** and **milpitas**.

Step 7: Start Replication

The API call **cdr_start_repl()** is invoked with the name of the replicate. This command starts data capture and queueing at all participating database servers.

Unlock the tables on **bangor**, **ames**, and **milpitas.**

After the completion of the start, the replication system is active. Data is being captured and propagated.

Step 8: Suspending or Removing Database Servers

Some months later the database administrator is informed that the computer **bangor** must be taken out of service for maintenance. The administrator has several choices:

- Suspend the **bangor** database server for both **milpitas** and **ames.**
- Remove **bangor** as a participant from **employeeinfo.**
- Delete the database server **bangor** from the global catalog.

Method 1 Uses the cdr_suspend_serv() Call

If the database administrator monitors the queues on **bangor** after the suspends to **ames** and **milpitas** are issued and ensures that the queues have emptied properly, no data intended for **bangor** is lost. All updates for **bangor** are queued at the respective origins (either **ames** or **milpitas**). When **bangor** returns to operation and is resumed at **ames** and **milpitas**, all updates that occurred during the **bangor** outage are propagated to **bangor**. No resynchronization is necessary. The database server is suspended on both of the other database servers (**ames** and **milpitas**). The replication is still in the active state.

Method 2 or 3 Assumes bangor Is Resynchronized

These methods assume that **bangor** is resynchronized before it is re-added to the **employeeinfo** replicate on **ames** and **milpitas**. Method 2 uses **cdr_participate_repl()** to remove **bangor** as a participant in the **employeeinfo** replicate. Method 3 uses **cdr_delete_serv()** to remove **bangor** as a participant in any replication. In both cases replication messages intended for **bangor** might be discarded. The replication is still active on **ames** and **milpitas**.

Step 9: Stopping Replication, cdr_stop_repl()

When the replication is no longer required, the administrator can issue the **cdr_stop_repl()** call. This API stops data capture at **bangor**, **ames**, and **milpitas**. After all table changes have propagated (queues have emptied), the tables will be consistent. During the period that the queues are emptying, the replication is quiescent; no data is being captured but data is still being propagated for the replication. When the queues have drained on all the sites, the replication returns to the inactive state.

Appendix

C

Configuration Parameters

The database server configuration file ($ONCONFIG) includes four configuration parameters that affect the behavior of Enterprise Replication:

- CDR_LOGBUFFERS
- CDR_EVALTHREADS
- CDR_DSLOCKWAIT
- CDR_QUEUEMEM

CDR_LOGBUFFERS

value	2048
units	kilobytes
range of values	128 to 32768
takes effect	when shared memory is initialized

The CDR_LOGBUFFERS configuration parameter specifies the amount of buffer space reserved by ER for the evaluation of transactions from the logical log. If the amount of buffer space reserved is too small, performance might be degraded.

Tip: *Logical logging can be buffered or unbuffered. Buffered logging introduces recovery issues and also presents potential latency problems because of the delays in writing logical-log buffers to disk. Informix recommends that you use unbuffered logging with Enterprise Replication.*

CDR_EVALTHREADS

value	1,2
units	evaluators
range of values	1 to 129
takes effect	when shared memory is initialized

The CDR_EVALTHREADS configuration parameter specifies the number of evaluator threads to create when ER is started and enables parallelism when ER evaluates transactions from the logical log. The format is:

```
(per-cpu-vp,additional)
```

The following table provides four examples of CDR_EVALTHREADS.

Number of Threads	Explanation	Example
1,2	1 evaluator thread per CPU VP, plus 2	For a 3 CPU VP server: (3 * 1) + 2 = 5
2	2 evaluator threads per CPU VP	For a 3 CPU VP server: (3 * 2) = 6
2,0	2 evaluator threads per CPU VP	For a 3 CPU VP server: (3* 2) = 6
0,4	4 evaluator threads for any database server	

CDR_DSLOCKWAIT

value 5

units seconds

takes effect when shared memory is initialized

The CDR_DSLOCKWAIT configuration parameter specifies the number of seconds the data synchronization component waits for a dslockwait time-out. The CDR_DSLOCKWAIT configuration parameter behaves similarly to the SET LOCK MODE statement. This parameter is useful in update-anywhere replication scenarios. Database servers that receive updates from other database servers in the replicate can abort because of locking problems. (When a database server inserts data into a table, it locks the index, which could potentially result in a collision with another database server.)

CDR_QUEUEMEM

value	4096
units	kilobytes
range of values	> = 500
takes effect	when shared memory is initialized

The CDR_QUEUEMEM configuration parameter specifies the maximum amount of memory that is used for the send and receive queues.

Tip: *Enterprise Replication holds receive-queue replication information in memory until memory is depleted. When memory is depleted, the information is written to the receive queue.*

The maximum memory used for queued elements on a specific Enterprise Replication database server is the total number of database servers involved in replication, multiplied by the value of CDR_QUEUEMEM.

When you increase the value of CDR_QUEUEMEM, you reduce the number of elements that must be written to disk, which can eliminate I/O overhead. Therefore, if elements are frequently stored on disk, increase the value of CDR_QUEUEMEM. Tune the value of CDR_QUEUEMEM for the amount of memory available on a computer.

Appendix

D

Status Information

The system-monitoring interface (SMI) tables in the **sysmaster** database provide information about the state of the database server. Enterprise Replication uses the following SMI tables to access information about database servers declared to Enterprise Replication.

Table	Description	Reference
syscdrerror	Error information	page D-2
syscdrgrp	Replicate group information	page D-2
syscdrgrppart	Detailed information about replicate groups	page D-3
syscdrpart	Participant information	page D-3
syscdrq	Queued data information	page D-4
syscdrqueued	Queued data information	page D-5
syscdrrecv	Receive queue information	page D-5
syscdrrepl	Replicate information	page D-6
syscdrs	Server information	page D-7
syscdrserver	Database server information	page D-9
syscdrtx	Transaction information	page D-10
syscdrtxproc	Transaction processes information	page D-11

syscdrerror

The **syscdrerror** table contains information about errors that Enterprise Replication has encountered.

Column	Type	Description
errornum	integer	Error number
errorserv	char(18)	Database server name where error occurred
errorseqnum	integer	Sequence number that can be used to prune single-error table
errorstmnt	char(128)	Error description
error time	datetime	Time error occurred
sendserv	char(18)	Database server name, if applicable, that initiated error behavior
reviewed	char(1)	Y if reviewed and set by DBA N if not reviewed

syscdrgrp

The **syscdrgrp** table contains replicate group information.

Column	Type	Description
grpname	char(18)	Replicate group name
isseq	char(1)	Y if sequential N if non-sequential

syscdrgrppart

The **syscdrgrppart** table contains replicate group information.

Column	Type	Description
grpname	char(18)	Replicate group name
server	char(18)	Database server name
state	integer	The following are state values when the OR operator is applied: 0x0000 = Defined in the catalog 0x0001 = Peer request failed; definition persists 0x0002 = Inactive state 0x0004 = Active state 0x0008 = Database server is suspended 0x00010 = Marked for deletion and waiting for cleanup to occur 0x00020 = Temporary state (in transition)

syscdrpart

The **syscdrspart** table contains participant information.

Column	Type	Description
replname	char(18)	Replicate name
servername	char(18)	Database server name
partname	char(1)	P = primary database server (read / write) R = target database server (read only)
partstate	char(18)	Participant state = active, inactive
dbsname	char(18)	Database name

(1 of 2)

Column	Type	Description
owner	char(8)	Owner name
tabname	char(18)	Table name
selectstmnt	char(255)	Select statement

(2 of 2)

syscdrq

The **syscdrq** table contains information about Enterprise Replication queues.

Column	Type	Description
servid	integer	The itentifier number of the database server
repid	integer	The identifier number of the replicate
srvname	char(18)	The name of the database server
name	char(18)	Replicate group or replicate name
bytesqueued	integer	Number of bytes queued and ready
lstcmmttime	integer	Commit time (relative to source) of last transaction committed for this replicate group or replicate on database server *servername*
lstquetime	integer	Commit time (relative to source) of last transaction queued for this replicate group or replicate for database server *servername*

syscdrqueued

The **syscdrqueued** table contains data-queued information.

Column	Type	Description
servername	char(18)	Sending to database server name
name	char(18)	Replicate group or replicate name
bytesqueued	integer	Number of bytes queued and ready
lastcommittime	integer	Commit time (relative to source) of last transaction committed for this replicate group or replicate on database server *servername*
lastqueuedtime	integer	Commit time (relative to source) of last transaction queued for this replicate group or replicate for database server *servername*

syscdrrecv

The **syscdrrecv** table contains receive information.

Column	Type	Description
servername	char(18)	Database server name receiving information
name	char(18)	Replicate group or replicate name
srccommittime	integer	Time last transaction processed on local database server committed on database server *servername*

syscdrrepl

The **syscdrrepl** table contains replicate information.

Column	Type	Description
replname	char(18)	Replicate name
grpname	char(18)	Replicate group name (if replicate is part of a replicate group) Null if replicate is not part of a replicate group
replstate	char(18)	
freqtype	char(1)	C if continuous I if interval T if time based M if day of month W if day of week
freqmin	integer	Minute refresh should occur Null if continuous
freqhour	integer	Hour refresh should occur Null if continuous
freqday	integer	Day of week or month refresh should occur
scope	char(1)	T for transaction R for row-by-row
invokerowspool	char(1)	Y if row spooling is invoked N if no row spooling is invoked
invoketranspool	char(1)	Y if transaction spooling is invoked N if no transaction spooling is invoked
primresolution	char(1)	I for ignore T for time stamp S for stored procedure

(1 of 2)

Column	Type	Description
secresolution	char(1)	S for stored procedure Null if not configured
storedprocname	char(18)	Stored-procedure name Null if not defined.
istriggerfire	char(1)	Y if triggers are invoked N if triggers are not invoked
iscanonical	char(1)	Y if conversion to canonical form is required N if no canonical form used

(2 of 2)

syscdrs

The **syscdrs** table contains information about database servers that have been declared to Enterprise Replication.

Column	Type	Description
servid	integer	Server identifier
servname	char(18)	Database server name
cnnstate	char(1)	C = Connected D = Connection disconnected (will be retried) T = Idle time-out caused connection to terminate. X = Connection closed by user command. Connection unavailable until reset by user
cnnstatechg	integer	Time that connection state was last changed
servstate	char(1)	A = Active S = Suspended H = Holding Q = Quiescent (initial sync state only)
ishub	char(1)	Y if server is a hub, N if not

(1 of 2)

Column	Type	Description
issporadic	char(1)	Y if the server connection is sporadic, N if not
issparse	char(1)	Y if server uses a sparse catalog, N if not
rootserverid	integer	The identifier of the root server
forwardnodeid	integer	The identifier of the parent server
timeout	integer	Idle timeout
sendq	char(18)	Dbspace name indicating where the send queue is located
recvq	char(18)	Dbspace name indicating where the receive queue is located
atsdir	char(128)	ATS directory spooling name
risdir	char(128)	RIS directory spooling name

(2 of 2)

syscdrserver

The **syscdrserver** table contains information about the database server that have been declared to Enterprise Replication.

Column	Type	Description
servername	char(18)	Database server name
connstate	char(1)	C = Connected D = Connection disconnected (will be retried) T = Idle time-out caused connection to terminate. X = Connection closed by user command. Connection unavailable until reset by user
connstatechange	integer	Time that connection state was last changed
servstate	char(1)	A = Active S = Suspended H = Holding Q = Quiescent (initial sync state only)
ishub	char(1)	Y if server is a hub, N if not
issporadic	char(1)	Y if the server connection is sporadic, N if not
issparse	char(1)	Y if server uses a sparse catalog, N if not
rootserverid	integer	The identifier of the root server
forwardnodeid	integer	The identifier of the parent server
idletimeout	integer	Idle time out
sendqueueloc	char(18)	Dbspace name indicating where the send queue is located
rcvqueueloc	char(18)	Dbspace name indicating where the receive queue is located
atsdir	char(128)	ATS directory spooling name
risdir	char(128)	RIS directory spooling name

syscdrtx

The **syscdrtx** table contains information about Enterprise Replication transactions.

Column	Type	Description
srvid	integer	Server ID
srvname	char(18)	Receiving from database server name
txprocssd	integer	Transaction processed from database server *servername*
txcmmtd	integer	Transaction committed from database server *servername*
txabrtd	integer	Transaction aborted from database server *servername*
rowscmmtd	integer	Rows committed from database server *servername*
rowsabrtd	integer	Rows aborted from database server *servername*
avgrcvlat	float	Average latency between commit time (on source) and receive time (on target)
avgcmtlat	float	Average latency between commit time (on source) and commit time (on target)
txbadcnt	integer	Number of transactions with source commit time (on database server *servername*) greater than target commit time

syscdrtxproc

The **syscdrtxproc** table contains information about an Enterprise Replication transaction process.

Column	Type	Description
servername	char(18)	Receiving from database server name
txprocessed	integer	Transaction processed from database server *servername*
txcommitted	integer	Transaction committed from database server *servername*
txaborted	integer	Transaction aborted from database server *servername*
rowscommitted	integer	Rows committed from database server *servername*
rowsaborted	integer	Rows aborted from database server *servername*
avgrcvlatency	float	Average latency between commit time (on source) and receive time (on target)
avgcmtlatency	float	Average latency between commit time (on source) and commit time (on target)
txbadcnt	integer	Number of transactions with source commit time (on database server *servername*) greater than target commit time

Index

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Symbols