

# Ingres<sup>®</sup> 9.3

---

## Replicator User Guide

INGRES

ING-93-RPU-03

This Documentation is for the end user's informational purposes only and may be subject to change or withdrawal by Ingres Corporation ("Ingres") at any time. This Documentation is the proprietary information of Ingres and is protected by the copyright laws of the United States and international treaties. It is not distributed under a GPL license. You may make printed or electronic copies of this Documentation provided that such copies are for your own internal use and all Ingres copyright notices and legends are affixed to each reproduced copy.

You may publish or distribute this document, in whole or in part, so long as the document remains unchanged and is disseminated with the applicable Ingres software. Any such publication or distribution must be in the same manner and medium as that used by Ingres, e.g., electronic download via website with the software or on a CD-ROM. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Ingres.

To the extent permitted by applicable law, INGRES PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL INGRES BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USER OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF INGRES IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The manufacturer of this Documentation is Ingres Corporation.

For government users, the Documentation is delivered with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013 or applicable successor provisions.

Copyright © 2009 Ingres Corporation. All Rights Reserved.

Ingres, OpenROAD, and EDBC are registered trademarks of Ingres Corporation. All other trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

# Contents

---

## Chapter 1: Introducing This Guide 9

In This Guide .....	9
Audience .....	9
System-specific Text in This Guide .....	10
Terminology Used in This Guide .....	10
Path Notation in This Guide .....	10
Query Languages .....	11
ISO Entry SQL92 .....	11
Syntax Conventions Used in This Guide .....	11

## Chapter 2: Understanding the Replicator 13

Replication .....	13
Ingres Replicator .....	14
Advantages of Replication .....	15
Replicator Data Control .....	16
Replicator Versus Ingres Star .....	17
Security for Replicator .....	17
Product Requirements for the Replicator .....	18
Replicator and Other Products .....	19
Replicator Concepts .....	20
Replicator Components .....	22
Change Recorder .....	22
Distribution Threads .....	24
Replicator Server .....	25
How Replication of Data Occurs .....	26
Replicator Management and Monitoring Tools .....	29
Replicator Processing Tables .....	30
Base Table .....	30
Shadow Table .....	30
Archive Table .....	30
Input Queue Table .....	30
Distribution Queue Table .....	31
Replicator Design Concepts .....	31
Consistent Distributed Data Set (CDDS) .....	32
CDDS Example: Subset of Tables .....	33
CDDS Example: Vertical Partitioning .....	33
CDDS Example: Horizontal Partitioning .....	34
Data Propagation Paths in the CDDS .....	34

---

CDDS Target Types .....	36
Sample Configurations .....	38
Peer-to-Peer Configuration .....	39
Central-to-Backup Configuration .....	40
Cascade Configuration .....	41
Central-to-Branch Configuration .....	42
Hub-and-Spoke Configuration .....	43
Combination Configurations .....	44
Responsibilities of the Distributed Database Administrator .....	45

## **Chapter 3: Planning Your Replication Scheme 47**

Replication Planning .....	47
Application Design Issues .....	47
Derived Information .....	48
Data Ownership .....	51
Data Aggregation .....	52
Replication Scheme Design Issues .....	52
Collision Design .....	52
How Errors Are Handled .....	56
Replicator Server Assignment .....	58
Storage Structures .....	60
Using Hash Structures to Avoid Deadlock .....	61
How You Implement Replicator .....	62
How You Prepare for Replication .....	63
Rules for Database Objects .....	63
Database Administrator Privileges .....	64
Ingres Net Entries .....	65
Replication Scheme .....	66
Replication Scheme Planning Worksheets .....	66
Database Worksheet .....	68
Identifying the CDDS .....	70
Table Worksheet .....	77
Replication Scheme Examples .....	78
Example: R.E.P.'s CDDS 0 .....	78
Example: R.E.P.'s CDDS 2 .....	80
Example: R.E.P.'s Server Assignments .....	80

## **Chapter 4: Configuring Replication Using Visual DBA 83**

How You Configure Replicator Using VDBA .....	84
Installation of Replication Objects .....	85
Replication Branch in DOM Window .....	85

---

How You Work with Replication Databases .....	86
How You Create a CDDS.....	87
CDDS Definition Dialog.....	88
View Table Registration Information .....	90
Creation of Replication Keys.....	91
How Configuration Information Is Propagated.....	92
How Replication Is Activated .....	93
How You Deactivate Replication .....	93
Error Mail Destinations .....	93

## **Chapter 5: Configuring Replication Using Replicator Manager 95**

Replicator Manager.....	95
repmgr Command—Start the Replicator Manager.....	96
Navigation and Operation of Replicator Manager.....	97
How You Configure Replicator Using Replicator Manager.....	101
repcat Command—Create Replicator Catalogs.....	102
Run the Replicator Manager the First Time .....	103
Configuration Menu .....	104
Configuration Menu Map .....	106
Editing Operations .....	108
Database Summary .....	108
Add a Database to the Database Summary List .....	109
Edit a Database .....	109
Delete a Database.....	110
CDDS Summary .....	110
Add a CDDS .....	111
Edit a CDDS .....	111
Delete a CDDS .....	112
CDDS Database and Servers Window .....	112
Propagation Path Definition Window.....	114
View Table Registration Information .....	116
Table Registration Summary .....	117
Register Tables.....	118
Deregister Tables.....	118
Table Registration Details Window .....	119
Create Support Objects.....	122
Activate or Deactivate a Table .....	123
CreateKeys Options.....	124
Move Configuration Data Window .....	126
Move Configuration Data.....	127
How the MoveConfig Operation Works.....	127
Activate Change Recording Window.....	128

---

Activate or Deactivate a CDDS or Database .....	129
Mail Notification List Window .....	129
Create a Mail Notification List.....	130
Delete Mail Notification Entries.....	130

## **Chapter 6: Using the Replicator Server 131**

Tools for Performing Replicator Server Tasks .....	131
Tools for Monitoring Replicator .....	131
Install Replicator Service on Windows.....	132
How You Start Replicator Server .....	133
How Server Behavior Is Controlled .....	134
Server Monitoring Configurations .....	134
Error Handling .....	139
Server Processing Activity .....	140
Server Parameters.....	144
How the Replication Cycle Works.....	144
Server Flags.....	145
Server Startup Settings Worksheet.....	152
Replication Monitoring Using Visual Performance Monitor or Visual DBA .....	156
Monitor Replication Activity .....	156
View Server Statuses .....	157
Raise Events at the Database Level .....	157
Raise Events at the Server Level .....	157
View the Queue Collision Report.....	157
Create and View the Table Integrity Report.....	157
View the Distributed Configuration Checker Report .....	158
Replication Monitoring Using Replication Manager.....	158
Replication Monitor Window .....	158
Advanced Options .....	168
How You Assign Server Numbers Greater Than Ten.....	169
Scheduling Servers .....	170
rpserver Command—Start a Replicator Server.....	171
Database Events.....	172

## **Chapter 7: Maintaining the Replicator 175**

Maintenance of Replicated Tables and Databases .....	175
How You Stabilize the Replication System.....	175
Copying Data into a Replicated Table .....	176
Using SQL Statements on Replicated Tables.....	176
Reconfiguration of the Replicator .....	177
How You Add a Database to the Replicated Configuration.....	178

---

How You Copy Replicated Tables Between Replicated Databases .....	179
How You Change the Table Schema .....	181
Optimizing Replicator Catalogs and Support Tables .....	182
Collision Resolution.....	182
Collision Detection .....	182
Methods to Handle Collisions .....	183
Disaster Recovery .....	185
Recovering Transactions from a Peer Database Using the Reconciler.....	185
Resolver Reports Menu (Replicator Manager).....	186
Queue Collision Report .....	188
Table Integrity Report Window.....	189
Distributed Configuration Checker Report .....	192
Replicated Databases Report .....	194
CDDS Data Propagation Paths Report.....	195
Registered Tables Report .....	197
How to Deactivate Replication .....	199
Deactivate the Change Recorder for a CDDS or Database .....	199
Removing Replication Objects.....	200

## **Chapter 8: Using Advanced Features 201**

Lookup Tables.....	201
How You Set Up Horizontal Partitioning .....	201
Priority Collision Resolution .....	208
Strategies for Avoiding Deadlock .....	210
Row Locking Option.....	211
rep_xx_lockmode Parameter—Override Default Locking.....	212
Maxlocks Values .....	214
Deadlock in the Distribution Threads.....	214

## **Appendix A: Data Dictionary Tables 215**

System Catalogs .....	215
dd_cdds Table .....	215
dd_databases Table .....	216
dd_db_cdds Table.....	217
dd_distrib_queue Table.....	218
dd_input_queue Table .....	219
dd_mail_alert Table .....	220
dd_paths Table.....	220
dd_regist_columns Table .....	221
dd_regist_tables Table.....	221
dd_servers Table .....	222

---

Base Table .....	223
Archive Table.....	223
Shadow Table .....	224
Internal Tables.....	225
 <b>Appendix B: Cluster Support</b>	 <b>227</b>
Replicator in a Cluster Environment .....	227
-TPC Flag—Control Use of Two-phase Commit .....	227
 <b>Index</b>	 <b>229</b>



# Chapter 1: Introducing This Guide

---

This section contains the following topics:

[In This Guide](#) (see page 9)

[Audience](#) (see page 9)

[System-specific Text in This Guide](#) (see page 10)

[Terminology Used in This Guide](#) (see page 10)

[Path Notation in This Guide](#) (see page 10)

[Query Languages](#) (see page 11)

[ISO Entry SQL92](#) (see page 11)

[Syntax Conventions Used in This Guide](#) (see page 11)

## In This Guide

The *Replicator User* Guide gives you a comprehensive look at Ingres® Replicator from the perspective of the database administrator. Specifically, this guide includes the following topics:

- An overview of the Ingres Replicator and its associated concepts
- Guidelines for designing a scheme to meet your replication needs
- Instructions on how to use and maintain Ingres Replicator
- Advanced Replicators, including creating lookup tables using Visual DBA or Replicator Manager

## Audience

This guide is designed for the distributed database administrator who is responsible for configuring and maintaining Ingres Replicator.

The guide assumes that the reader is familiar with using Ingres, Ingres Net, and Visual DBA.

## System-specific Text in This Guide

This guide provides information that is specific to your operating system, as in these examples:

**Windows:** This information is specific to the Windows operation system.

**UNIX:** This information is specific to the UNIX operation system.

**VMS:** This information is specific to VMS operating system.

When necessary for clarity, the symbol ■ is used to indicate the end of the system-specific text.

For sections that pertain to one system only, the system is indicated in the section title.

## Terminology Used in This Guide

This guide uses the following terminology:

- A *command* is an operation that you execute at the operating system level. An extended operation invoked by a command is often referred to as a *utility*.
- A *statement* is an operation that you embed within a program or execute interactively from a terminal monitor.

**Note:** A statement can be written in Ingres 4GL, a host programming language (such as C), or a database query language (SQL or QUEL).

## Path Notation in This Guide

The directory structure of an Ingres installation is the same regardless of operating system. Rather than showing path examples for all environments, this guide uses UNIX notation only.

For example: When describing the location of the collation sequence file, the guide shows: \$II\_SYSTEM/ingres/files/collation/collation\_name.

On Windows, the location is:

%II\_SYSTEM%\ingres\files\collation\collation\_name

On VMS, the location is:

II\_SYSTEM:[INGRES.FILES.COLLATION]collation\_name

## Query Languages

The industry standard query language, SQL, is used as the standard query language throughout the body of this guide.

QUEL commands and system parameters, such as environment variables/logicals, are also included in this guide. For a complete description of available QUEL statements, see the *QUEL Reference Guide*.

## ISO Entry SQL92

Ingres is compliant with ISO Entry SQL92. In addition, numerous vendor extensions are included. For details about the settings required to operate in compliance with ISO Entry SQL92, see the *SQL Reference Guide*.

## Syntax Conventions Used in This Guide

This guide uses the following conventions to describe syntax:

Convention	Usage
Monospace	Indicates key words, symbols, or punctuation that you must enter as shown
Italics	Represent a variable name for which you must supply an actual value
[ ] (brackets)	Indicate an optional item
{ } (braces)	Indicate an optional item that you can repeat as many times as appropriate
(vertical bar)	Separates items in a list and indicates that you must choose one item



# Chapter 2: Understanding the Replicator

---

This section contains the following topics:

[Replication](#) (see page 13)

[Ingres Replicator](#) (see page 14)

[Product Requirements for the Replicator](#) (see page 18)

[Replicator and Other Products](#) (see page 19)

[Replicator Concepts](#) (see page 20)

[Replicator Components](#) (see page 22)

[Replicator Processing Tables](#) (see page 30)

[Replicator Design Concepts](#) (see page 31)

[Sample Configurations](#) (see page 38)

[Responsibilities of the Distributed Database Administrator](#) (see page 45)

## Replication

*Replication* is the act of maintaining identical copies of a defined set of data in more than one database. Replication allows continuous access to your databases, protects against catastrophic failures, improves performance, and is an excellent means of maintaining data across different database systems.

## Ingres Replicator

Ingres Replicator is a database connectivity system that manages replication. It provides transparent replication of data across databases on local or remote sites.

You can replicate between different databases on the same machine or on different machines on the other side of the world, as long as they are in a distributed network. So instead of having all your users throughout the world access the same database in one location, you can have a replicated database in every region.

Data from a transaction are not available for replication until the transaction has committed locally.

Ingres Replicator works behind the scenes to replicate data asynchronously by transaction. Designed as a layer that sits between the database and an application, Ingres Replicator is transparent to users.

Ingres Replicator queues each replicated transaction and uses two-phase commit to move the transaction from the Ingres Replicator queues to the target database. In the event of a source, target, or network failure, data integrity is enforced through this two-phase commit protocol by ensuring that either the whole transaction is replicated, or none of it is.

If the replication fails during two-phase commit, it remains in the queue and is tried again. Therefore, if one replicated database goes down, all the transactions that must be replicated to it remain queued and are sent again in the correct order as soon as it becomes available. For more information, see [How Two-Phase Commit Works](#) (see page 28).

**Note:** Two-phase commit is not available for installations running the Ingres Cluster Solution. For information on how to turn off two-phase commit and use Ingres Replicator with clusters, see the appendix “Cluster Support.”

Ingres Replicator also allows you to monitor your replicated transactions and run reports on your replication system.

## Advantages of Replication

Specific advantages of using Ingres Replicator include:

- Improved performance

When multiple sites need access to the same data, providing replicated copies of the data at each local site reduces network traffic and improves response times. Also, by distributing the users over more than one machine, the load on any single machine is reduced, overcoming system bottlenecks.

- Fault tolerance

If a node in the network becomes unavailable, replicated changes to be sent to that node are queued in order and are sent when the node comes back online. You can also configure Ingres Replicator so that if a particular machine fails, users can switch to another machine and continue working. If two databases do become inconsistent, Ingres Replicator has procedures for reconciling them that work in conjunction with the normal recovery procedures. Because your data exists in more than one location, loss of data on one machine is not catastrophic.

- Flexibility

Ingres Replicator offers existing and future database connectivity solutions in an ever-changing computing environment. If your system operates over a number of different databases, you can use Ingres Replicator to connect them. Ingres Replicator interoperates with non-Ingres databases through Enterprise Access products. You can maintain your legacy systems, or use Ingres Replicator to complete a controlled migration to an Ingres database system. Thus users of the legacy system can access data created on desktop or other heterogeneous environments.

## Replicator Data Control

Ingres Replicator gives you significant control over the flow of data. You can configure Ingres Replicator to fit your specific needs. Because Ingres Replicator has great flexibility, the key to achieving the results you want is planning your replication scheme.

You can control:

- Which data is replicated

You can replicate any combination of the following sets of data:

- Whole database
- Subset of tables
- Subset of columns
- Subset of rows (based on specified values)

- Where the replicated data goes

You can configure Ingres Replicator so that data is replicated in:

- One direction with a single target
- One direction with multiple targets
- Both directions between two databases
- Many directions between many databases
- Combination of any of these options

- What route the data takes

You must specify the route the replicated data takes to get to all its targets.

- When data is replicated

You can specify whether you want Ingres Replicator to propagate changes continuously, once a day, or on demand.

- What kind of databases you can use

You can use Ingres databases in any capacity with Ingres Replicator. Certain Enterprise Access databases can be used as both source and target databases; others, however, can be used only as a target database in a read-only capacity.



## Replicator Versus Ingres Star

Like Ingres Replicator, Ingres Star maintains data across two or more distributed databases.

However, Ingres Star uses two-phase commit between the local and remote databases. During an update by a local user, Ingres Star takes locks on the local database and the remote databases, and does not release the locks until the two-phase commit procedure has been successfully completed. Holding locks can cause delays for the local user.

Unlike Ingres Star, Ingres Replicator updates the local database asynchronously, and takes locks on the Ingres Replicator queues and each remote database in turn for the duration of the two-phase commit procedure. Thus, using Ingres Replicator instead of Ingres Star results in significant performance gains.

With Ingres Replicator, you can customize your replication environment in ways that are not available with Ingres Star.

With Ingres Star, either the transaction is completed in all databases or in none of them. With Ingres Replicator, however, two users can update the same record in different replicated databases. This can result in a collision. For more information, see Collision Design (see page 52).

## Security for Replicator

Security for Ingres Replicator is provided through Ingres Net, a database connection program that uses industry-standard networking protocols. Ingres Net provides security by requiring authorization before a user can access the system. Through Ingres Net, security is controlled by the operating system's password or by using Ingres Net passwords. For more information, see the *Connectivity Guide*.

## Product Requirements for the Replicator

To use Ingres Replicator, each location that requires data replication must have the following Ingres products:

- Ingres DBMS Server

A local DBMS server is required to support the local database. The server interprets client component requests and interacts or manipulates the database accordingly.

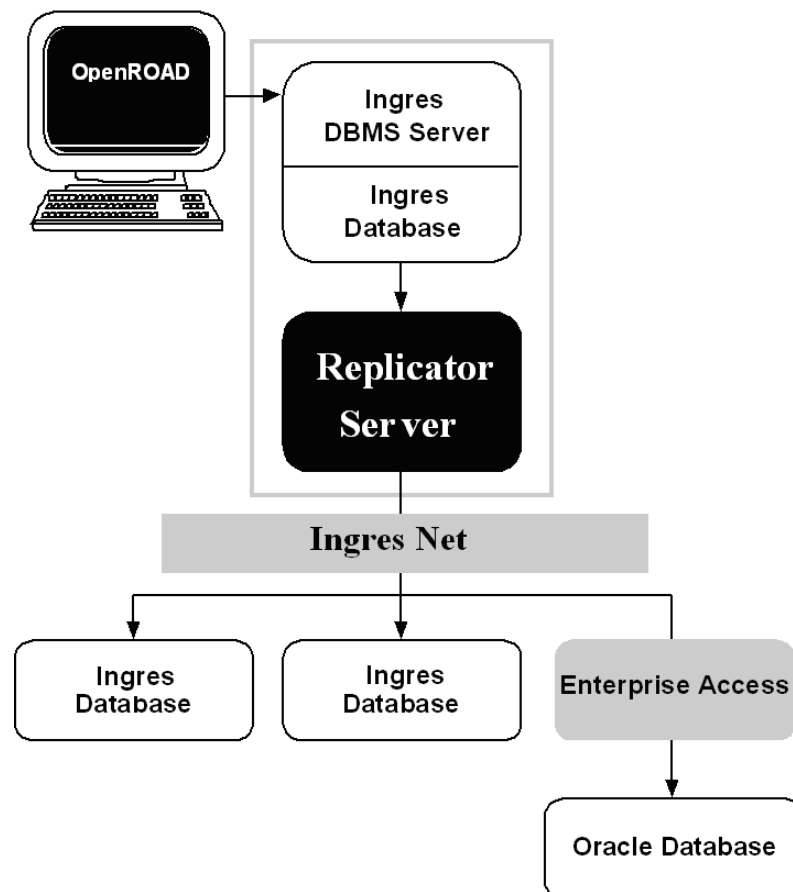
- Ingres Net

Ingres Net is required to access data on remote nodes on your network. It is an interface to your existing network setup and protocols. The remote transmission of replicated data among source and target databases is handled completely by Ingres Net and is transparent to the user.

## Replicator and Other Products

Ingres Replicator can also be used with Enterprise Access products, including Oracle, MS SQL, DB2 UDB, RMS, RDB, or EDBC products, including CA-Datcom/DB, IBM DB2, and IMS. Ingres Replicator is compatible with all Enterprise Access products, EDBC products, and the abstract data type, providing the database product that underlies your Enterprise Access Server or EDBC Server *and* your Enterprise Access product or EDBC product supports it. If your legacy system relies on data from a heterogeneous system, you can designate replicated data to be sent through the gateway.

The following figure illustrates the components of a sample Ingres Replicator environment consisting of a development tool, an Ingres DBMS Server and database, the Ingres Replicator Server, Ingres Net, two additional Ingres databases, and an Enterprise Access database:



## Replicator Concepts

To create a successful replication scheme, it is important to understand concepts and terms used in Replicator.

Key Ingres Replicator concepts are as follows:

### **Collision**

A collision is an event that occurs when more than one user updates the same replicated row in different databases. Ingres Replicator can preserve the history of changes made to each replicated row. If the same rows in two databases do not match in the last transaction, Ingres Replicator detects a collision.

For more information, see Collision Design (see page 52).

### **Consistent Distributed Data Set (CDDS)**

A CDDS defines what data is replicated and where it resides.

The CDDS is the unit of organization in Ingres Replicator. Various processes, such as collision and error handling, are specified at the CDDS level.

For more information, see Consistent Distributed Data Set (CDDS) (see page 32).

### **CDDS Target Type**

A target type defines which database within a CDDS is allowed to alter the data and how Ingres Replicator acts on each database within the CDDS.

The target types are:

- Full peer—generates changes to be replicated, accepts replicated changes and can detect collisions
- Protected read-only—accepts replicated changes and can detect collisions
- Unprotected read-only—accepts replicated changes but cannot detect collisions

A CDDS has a target type defined in every database that is contained in the CDDS. You must define target types for each CDDS. For more information, see CDDS Target Types (see page 36).

### **Data Propagation Path**

A data propagation path defines what route the data takes to reach target databases within a CDDS.

For more information, see Data Propagation Paths in the CDDS (see page 34).

**Replicated transaction key (replication key)**

To identify information specific to Ingres Replicator during processing of a replicated row, the system generates a key for each such row.

The key consists of:

- A database number that identifies the replicated database that manipulated the row
- The ID of the DBMS Server transaction that manipulated the row
- The sequence number within the transaction

For more information, see Change Recorder (see page 22).

**Other concepts...**

The section Replicator Components (see page 22) defines these concepts:

- Change Recorder
- distribution threads
- Replicator Server

The section Replicator Processing Tables (see page 30) defines these concepts:

- base table
- shadow table
- archive table
- input queue
- distribution queue

## Replicator Components

This section provides an overview of how Ingres Replicator components work together to process a replication from one database to another. This description is intended to provide the background necessary to maintain and troubleshoot Ingres Replicator. It discusses full peer to full peer or full peer to protected read-only configurations, and assumes that there are no collisions.

Ingres Replicator consists of three distinct components that participate in the actual replication of data:

- **Change Recorder**

The Change Recorder is a DBMS Server component that captures information about each update to a replicated table as part of the user transaction. The Change Recorder inserts a row in the shadow table and, if necessary, in the archive table, for each base table row manipulated by the user. It also adds an entry to the input queue for each such row.

- **Distribution threads**

When the user commits the transaction, a distribution thread in the DBMS Server reads the input queue and for each input row and destination, it inserts an entry into the distribution queue. The input rows are deleted and the Replicator Server is alerted.

- **Replicator Server**

The Replicator Server is a separate process that reads the distribution queue and propagates each distribution queue row to the corresponding target. At the end of each set of rows comprising a user transaction, the distribution rows are deleted and two phase commit is used to secure the transmitted transaction.

In addition, Ingres Replicator includes a management component—either Visual DBA or the Replicator Manager, plus additional utilities that provide operational support.

### Change Recorder

The Change Recorder is responsible for recording all changes made to base tables registered for replication. The recording process is accomplished internally in the DBMS, which automatically performs work on behalf of the user. The DBMS manipulates support tables to maintain a history of replicated transactions and a queue of transactions that need to be distributed by the distribution threads. This queue is known as the input queue. The data contained in the support tables is also used for collision resolution.

## How the Change Recorder Works

When a user makes a change in a table that has been registered for replication, the Change Recorder function is activated and the following process begins:

1. A replicated transaction key for that row is generated internally.
2. The Change Recorder inserts a new row into the shadow table with the replicated transaction key and the key columns of the base table.

If the change is an insert, the shadow record for this new replicated transaction key contains zeros for the previous replicated transaction key. If the change is an update or delete, the shadow record for this new replicated transaction key contains the replicated transaction key for the previous replication transaction for this row.

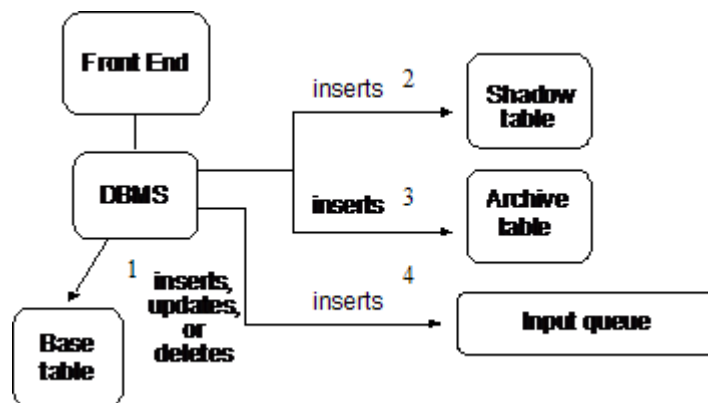
3. In full peer situations, if the change is an update or delete, the Change Recorder inserts the previous replicated transaction key and an image of the row associated with this ID into a new row in the archive table. An indicator is set to reflect that the data for the old replication is now in the archive table instead of the base table.

In this way the shadow and archive tables provide the sequential history of every replicated row.

4. The Change Recorder inserts the replicated transaction key into the input queue.

The input queue contains the information necessary to complete the replication, except the destinations.

The following figure illustrates the tasks of the Change Recorder:



## Distribution Threads

The distribution threads put the destination address on the replication that needs to be propagated. The distribution threads work in parallel to clear the input queue.

The distribution threads work within the DBMS Server to take the information from the input queue and expand it to fully specify what data needs to be replicated to which targets. The expanded information is stored in the distribution queue. The distribution threads do not perform any replication; they merely prepare data for replication by the Replicator Server. However, no data is expanded to the distribution queue if the threads are not configured. There can be a number of threads in each DBMS Server.

### How the Distribution Thread Works

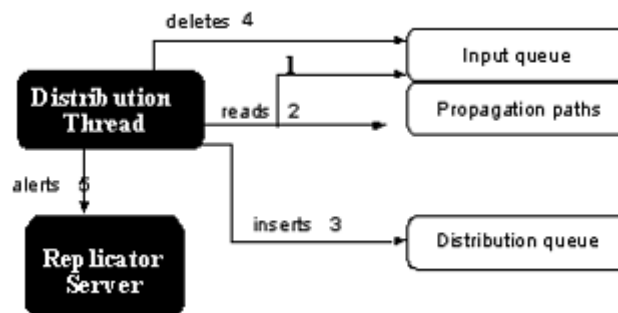
The distribution thread performs the following tasks:

1. Reads a record from the input queue to obtain the replication information.
2. Obtains destination information from the propagation paths table.
3. Inserts a row into the distribution queue for every destination of the replication.

The number of rows a distribution thread inserts into the distribution queue per replication is equal to the number of targets that you have defined for that replication's Consistent Distributed Data Set (CDDS). Each row contains the information from the input queue plus the destination information.

4. Deletes the replication's rows from the input queue.
5. Alerts the Replicator Server associated with the replication's CDDS.

The following figure illustrates the tasks of the distribution thread:





## Replicator Server

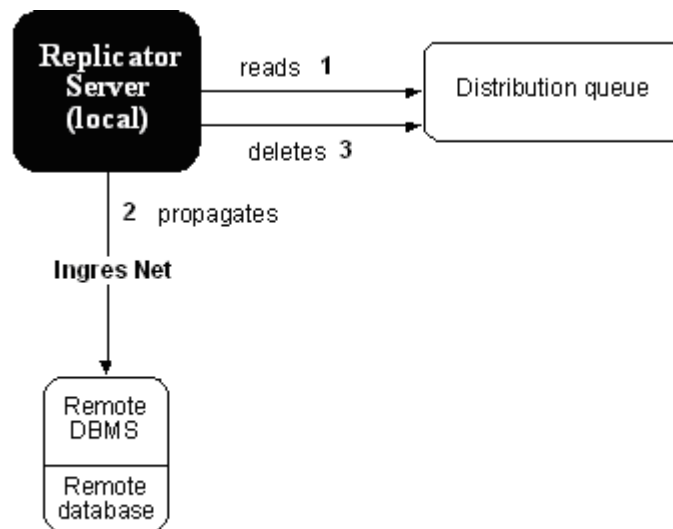
The Replicator Server is a stand-alone process that sends changes prepared by the distribution threads to their respective targets. Once the transaction has been transmitted, the replications are secured on the target database and are removed from the distribution queue. The Replicator Server also detects and resolves collisions.

### How the Replicator Server Process Works

After the distribution threads alert it, the Replicator Server completes the following tasks:

1. The Replicator Server reads the distribution queue.
2. The Replicator Server propagates each replicated row to the target database using direct SQL or database procedures over Ingres Net.
3. After propagating a set of rows in an original user transaction by CDDS, the server deletes the corresponding rows from the distribution queue.
4. The Replicator Server uses two-phase commit to update two databases. For more information, see Two-Phase Commit (see page 28).

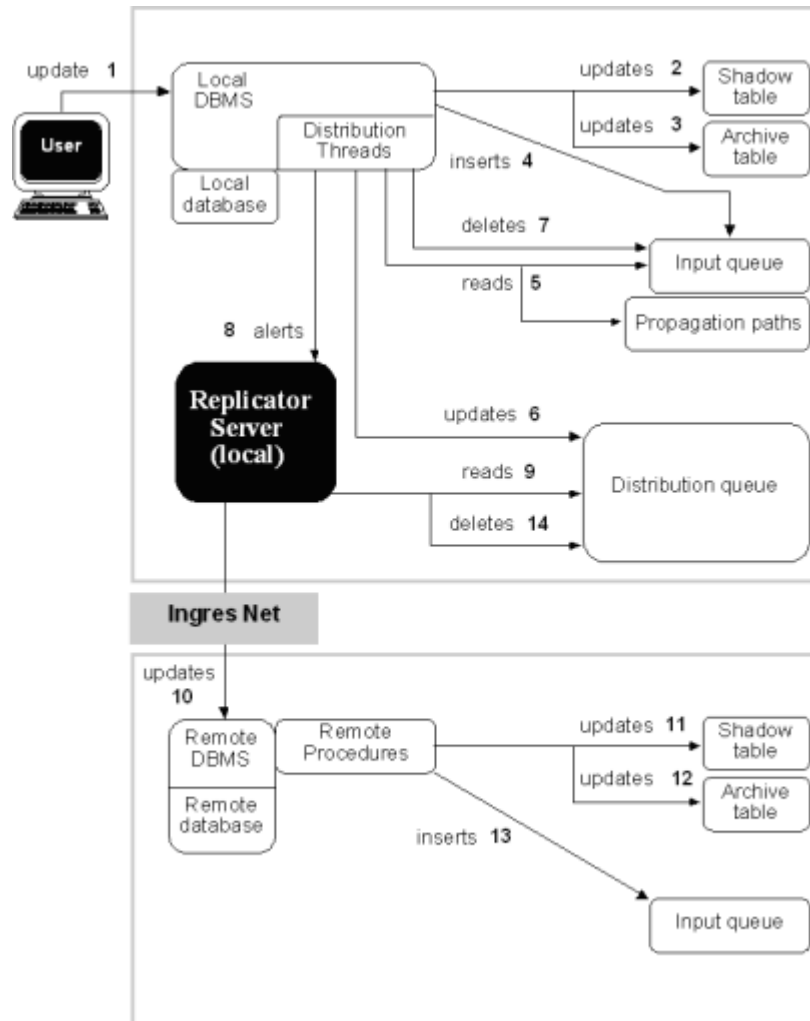
The Replicator Server also performs tasks such as checking the status of its Ingres Net connections. The following figure illustrates the tasks of the Replicator Server:



## How Replication of Data Occurs

Replication of data is carried out by the sequence of tasks performed by the Change Recorder, Distribution Threads, and Replicator Server.

Whenever a database table registered for replication is manipulated, the replication of that table is executed as shown in the following diagram, starting with #1:



The steps in the previous diagram are as follows:

1. A user updates the local database through the local DBMS Server.
2. The Change Recorder updates the shadow table.
3. The Change Recorder updates the archive table.
4. The Change Recorder adds a row to the input queue.
5. After a commit, the distribution threads read the input queue and the propagation paths table to determine whether and to where the replication needs distributing.
6. The distribution threads update the distribution queue with the replication and its destination information.
7. The distribution threads delete the replication from the input queue.
8. The distribution threads alert an active Replicator Server.
9. Replicator Server reads the distribution queue.
10. Replicator Server updates the remote database using remote SQL or database procedures.
11. If the remote database is a full peer or protected read-only target, the Replicator Server updates the corresponding shadow table.
12. If the remote database is a full peer target, the Replicator Server updates the corresponding archive table.
13. If the remote database is a full per target, a row is added to the remote input queue.
14. The appropriate row is deleted from the local database's distribution queue.  
  
The Replicator Server deletes the corresponding rows from the local distribution queue.
15. The changes at the remote and local databases are secured using two-phase commit.

## How Two-Phase Commit Works

Ingres Replicator uses a two-phase commit protocol to ensure the integrity of the distributed, replicated data. The term *two-phase* commit refers to the two phases required to commit a distributed transaction where two databases are updated. Unlike Ingres Ingres Star, Ingres Replicator updates only two databases at a time.

Two-phase commit occurs during the last step in the diagram in How Replication of Data Occurs (see page 26).

**Note:** The two-phase commit protocols are not used on the target connection; therefore, the transaction is not left in its installation log file.

Two-phase commit during replication involves the following steps:

1. The Replicator Server sends the replication to a target DBMS server.
2. The Replicator Server prepares to commit the transaction between the local and the target databases.
3. If Step 2 is successful, the Replicator Server completes committing the transaction at the target and local databases.

If an interruption occurs during the two-phase commit (for example, a machine or database shutdown), it is possible that a willing-to-commit transaction can be left on the Transaction Log File of the source database. To clear the willing-to-commit transaction, you can restart the Replicator Servers. If this fails to remove the willing-to-commit transaction from the Transaction Log File, you need to use the lartool utility. For more information, see the *Command Reference Guide*.

**Note:** Two-phase commit is not available for installations running the Ingres Cluster Solution. For more information, see the appendix "Cluster Support."

## Replicator Management and Monitoring Tools

Management and configuration tasks for Ingres Replicator can be performed using either Visual DBA or the Replicator Manager.

Monitoring tasks for Ingres Replicator can be performed using either Visual DBA or Visual Performance Monitor. As the smaller and simpler tool, Visual Performance Monitor is preferred.

### Windows and UNIX:

- Visual DBA

Visual DBA provides a visual user interface for performing most Ingres Replicator management functions, including creating system catalogs and configuring a replication scheme—defining the replication scenario, registering tables for replication, and moving the replication configuration to the target databases. Visual DBA provides several reports about your replication scheme, and it allows you to monitor and control server activity.

- Visual Performance Monitor

Visual Performance Monitor provides a visual user interface for monitoring most replication aspects including replication activity, Replicator Server status, raising database events for all Replicator Servers or an individual server, replication collisions, table integrity and replication configuration consistency between databases. 📄

- Replicator Manager

The Replicator Manager is a forms-based management tool used to configure your replication scheme, monitor replication activity, and report on the replication configuration. Configuring a replication scheme includes defining the replication scenario, registering tables for replication, and moving the replication configuration to the target databases. Replicator Manager provides several reports that can assist you to monitor and control server activity.

- Supporting utilities

Ingres Replicator provides additional utilities that assist in the management of the replicated databases. These utilities are described in the *Command Reference Guide*.

## Replicator Processing Tables

When a row is manipulated in a table registered for replication, up to four other tables can be used in processing a transaction for replication. A brief description of the role of each processing table that Ingres Replicator uses to replicate data follows. For a full definition of each table, see the “Data Dictionary Tables” appendix.

### Base Table

The base table is the table manipulated by the user.

### Shadow Table

The shadow table maintains a history of previous transactions. It contains a row for each row in the base table that has been manipulated. Shadow table rows include the system-generated replicated transaction key for that manipulation, plus the previous replicated transaction key, along with the base table’s designated unique key.

**Note:** Shadow tables are used only in full peer or protected read-only targets.

### Archive Table

The archive table contains the “before-images” of rows that have been updated or deleted. Archive table rows include all the replicated columns of the base table, along with the replication key of the manipulation that altered that row. Each row in the archive table corresponds to a row in the shadow table.

**Note:** Archive tables are used only in full peer targets.

### Input Queue Table

The input queue table contains a transient entry for each row manipulated in a replicated table.

## Distribution Queue Table

The distribution queue table contains a transient entry for each replicated row that is to be transmitted to another database. It contains the identifier of the target database.

The shadow and archive tables, together with the database procedures are generally referred to as a replicated table's *support objects*. For more information, see Full Peer Target (see page 37) and Protected Read-Only Target (see page 38).

## Replicator Design Concepts

The following design concepts organize replicated data:

- Consistent Distributed Data Set (CDDS)
- Data propagation paths
- Target types

## Consistent Distributed Data Set (CDDS)

A Consistent Distributed Data Set is a set of data that is kept consistent (identical) across two or more databases. A CDDS provides a method of defining and grouping data so that an entire database or parts of databases can be replicated to different target sites.

The CDDS gives the distributed DBA the ability to replicate information according to the needs of the business. Data can be replicated precisely when, where, and how the company requires. Changes to data in a CDDS must only be permitted to take place in full peer targets. This is not strictly enforced by Ingres Replicator, but if not adhered to, can result in collisions or inconsistencies.

A CDDS can consist of:

### **Whole database**

The whole database is replicated. For example, where two databases are geographically remote, each office accesses the local replicated database and propagates changes to the other database.

### **Subset of tables**

Only certain tables in the database are replicated. For example, branch offices do not need all the information that the head office has. For a detailed example, see CDDS Example: Subset of Tables (see page 33).

### **Subset of columns (vertical partitioning)**

Only selected columns in selected tables are replicated. For a detailed example, see CDDS Example: Vertical Partitioning (see page 33).

### **Subset of rows based on values (horizontal partitioning)**

Only certain rows from selected tables are replicated based on the values of one or more columns. For example, the branch office in London only gets those rows with London in the location column. Horizontal partitioning can only be done for specific values; data ranges are not allowed. For a detailed example, see CDDS Example: Horizontal Partitioning (see page 34). For information about setting up horizontal replication, see How You Set Up Horizontal Partitioning (see page 201).

When assigning tables to a CDDS, be sure to consider logical groupings of data. Keep in mind that if a user transaction crosses two or more CDDSs, for example, update a table in one CDDS and insert into a table in another CDDS, the Replicator Server(s) processes it as two or more separate transactions. For more information, see How Errors Are Handled (see page 56).



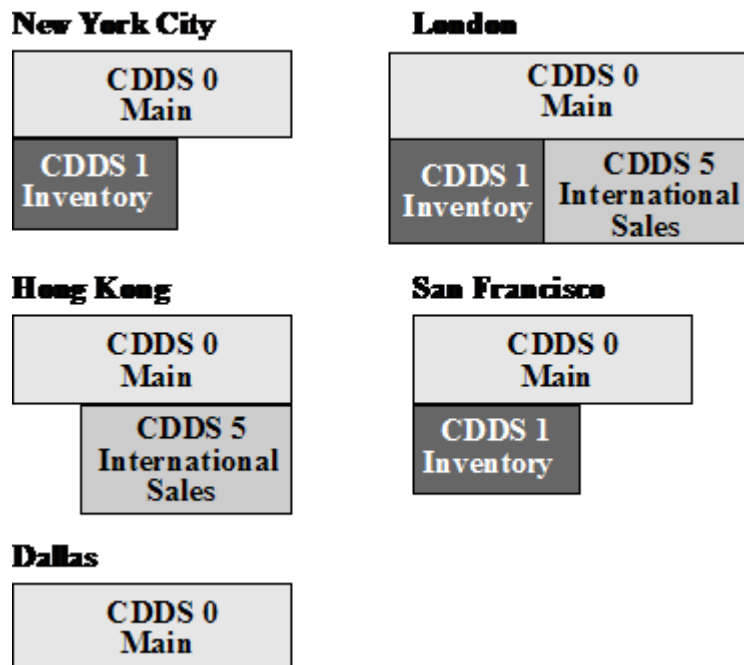
## CDDS Example: Subset of Tables

To illustrate CDDS planning, consider the example of 'Round the Earth Publishing (R.E.P.). R.E.P. has a replicated database that is split up into several CDDSs. R.E.P. has its headquarters in New York City and has offices in San Francisco, Dallas, London, and Hong Kong. Most of R.E.P.'s data is in one main CDDS (CDDS 0), which is shared by all locations.

R.E.P.'s inventory information is located in New York, San Francisco, and London, so these locations share an inventory CDDS (CDDS 1).

London and Hong Kong also share an international sales CDDS (CDDS 5).

The following figure illustrates the location of each CDDS:



## CDDS Example: Vertical Partitioning

R.E.P.'s CDDS 0 has a table that contains employee information. The employee table has a salary column. Top Management decides that the salary column must only be accessible to users in New York. Therefore, while the employee table is replicated, its salary column is not. For more information, see the column item in CDDS Definition Dialog (see page 88).

## CDDS Example: Horizontal Partitioning

The inventory CDDS contains information about R.E.P.'s products, but not actual inventory quantities. R.E.P. decides to split up its quantity on hand information into different warehouses.

The New York version has all the information, but the San Francisco and London versions have only the information for their respective locations. To do this, R.E.P. creates three new CDDSs, one for New York, one for San Francisco, and one for London. One of the replicated tables in the three warehouse CDDSs is the warehouse inventory table; it contains a column that specifies the location of the warehouse. The rows belonging in which CDDS is determined by the value in the location column of the warehouse inventory table.

To assign a CDDS by value, you create a lookup table containing the column for CDDS values and its corresponding attribute values. Using Visual DBA or the Replicator Manager, you can associate the lookup table with a base replicated table in your system. For detailed procedures and examples for implementing horizontal partitioning, see *How You Set Up Horizontal Partitioning* (see page 201).

## Data Propagation Paths in the CDDS

Replicator moves information from one database to another according to the data propagation paths for that CDDS. In a data propagation path, a database can be one of the following three types:

### **Originator**

The database where a user made a change to a replicated table.

### **Local**

The database that propagates the change to the target.

This is the last database that the change goes through before reaching the target. The local database can be the same as the originator database. A local database must be a target database in another path (it must receive the data as a target) before it can propagate the data as a local database.

### **Target**

The database that receives the change.

Every database in the CDDS must be a target (except the originator in that particular path). The target and the originator database cannot be the same in a given path.

## Example: Data Propagation Path

In the R.E.P. scenario described in CDDS Example: Subset of Tables (see page 33), several databases are allowed to manipulate data. Paths must be defined for each originating database. To do this, a separate data propagation path is defined for each CDDS that resides on an originating database. The path itself consists of separate entries for each target of the originating database.

Here is the example for CDDS 1 in San Francisco, which is set up to transmit inventory information to New York and London.

Data Propagation Paths for CDDS 1 in San Francisco:

Originator DB	Local DB	Target DB
SFO	SFO	NYC
SFO	NYC	LON

Note that the local database is part of all path definitions. It is used to differentiate between a direct and indirect path to a target. In the first entry of the example, the originator and local databases are both SFO; this refers to the start of the replication path. In the second entry, the originator and local databases differ; this indicates that the local database, NYC, is responsible for cascading replication to the target, LON. As a result of cascades, a data propagation path can run through many databases.

Data Propagation Paths for CDDS 1 in New York:

Originator DB	Local DB	Target DB
NYC	NYC	SFO
NYC	NYC	LON

Note that in this example, New York sends inventory data directly to San Francisco and London. The choice between the cascading paths in the SFO example and the direct paths in the NYC example can be influenced by various factors, such as communications infrastructure, transaction volumes at each site, and processor capacities.

Remember that a CDDS is simply a piece of a larger database. These individual pieces can reside at any site in an enterprise. In the R.E.P. scenario, international sales information (CDDS 5) from Hong Kong does not get copied to the master database in New York. Thus, no path definition for Hong Kong to New York for CDDS 5 is required. The New York inventory CDDS (CDDS 1) maintains the same standard, sharing only with the necessary remote sites.

## CDDS Target Types

CDDS target type refers to the function a database performs within a CDDS. The target type determines whether changes to databases can be replicated and whether collisions are detected. Target types are defined separately for each CDDS; therefore, a database can have more than one target type if it is contained in more than one CDDS.

Ingres Replicator currently supports the following types of CDDS target types to carry out replication:

- Full peer
- Protected read-only
- Unprotected read-only

A database provides services of a particular target type based on the CDDSs it contains. To plan database replication, you need to decide what role you want each database to play within each CDDS and assign one of the three target types. Your decision is based on the type of replication behavior you want the database to perform and the computing resources available in your system. By carefully analyzing how various work groups use data, you can determine which targets require full peer or read-only capabilities.

## Full Peer Target

A full peer designation is required for production CDDs where interactive users or other processes first manipulate data. Full peer CDD targets make full use of the Ingres Replicator processing capabilities and maintain shadow tables, archive tables, as well as the input and distribution queues.

In a full peer CDD target, each replicated table has a shadow table. Each change to a row designated for replication is tracked through a shadow record entry in the shadow table. Before-images of updates and deletes are maintained in the archive table. The shadow table itself contains the information necessary to allow the propagation of row changes to other targets.

If a row change is an update or a delete, the old image of the row is stored in an archive table. The archive table makes it possible to re-run the transactions that occurred in a database, in the proper order with the proper data, even if the replicated row has been changed many times. For each change in a replicated table of a full peer target, an entry is also made in the input queue table.

If a full peer source sends information to a full peer CDD target (in a peer-to-peer configuration), database procedures are used to transmit the data, while the DBMS captures local changes to the database. As a result, all replicated tables in this CDD must be registered with Ingres Replicator, not only to transmit information to other replicated CDD targets, but to receive information as well.

As the workhorse of the Ingres Replicator system, a full peer target consumes more disk space than any other target type; it makes use of all support tables and many of the CPU cycles available in the Ingres Replicator system to process its replication.

## Protected Read-Only Target

A protected read-only target is used to receive information from a full peer source. The term *protected* means that this target contains sufficient information to detect collisions against the source copy of the data. Users must not make changes to the data in a protected read-only target, although Ingres Replicator does not prevent this. The ability to detect and resolve collisions provides a protection against any change that is made to a replicated table from *another* source. Note that collisions resulting from updating data directly at the protected read-only target are generally not detectable.

A protected mode is important because more than one source can be transmitting to a single protected read-only target. If the target is not protected, transactions can arrive out of order. A protected read-only target maintains a shadow table for each replicated table but does not maintain archive tables or input and distribution queues. This allows the target to only accept replicated rows in order, thus ensuring that consistent data is available to users. Other sources replicate data to this target by database procedures; so although this target does not send data, all replicated tables in the protected read-only target must be registered with Ingres Replicator, and most Ingres Replicator support objects must be present.

## Unprotected Read-Only Target

In unprotected mode, there is no safeguard against any change that is made to a replicated table from another source (or locally); hence, use unprotected read-only target only if one full peer source sends information to the read-only target. Because the unprotected read-only target does not maintain any support objects within the database, no table registration is required for the target's tables. Collisions cannot be detected because of the absence of the shadow table.

All Enterprise Access and EDBC databases can only be designated as unprotected read-only.

## Sample Configurations

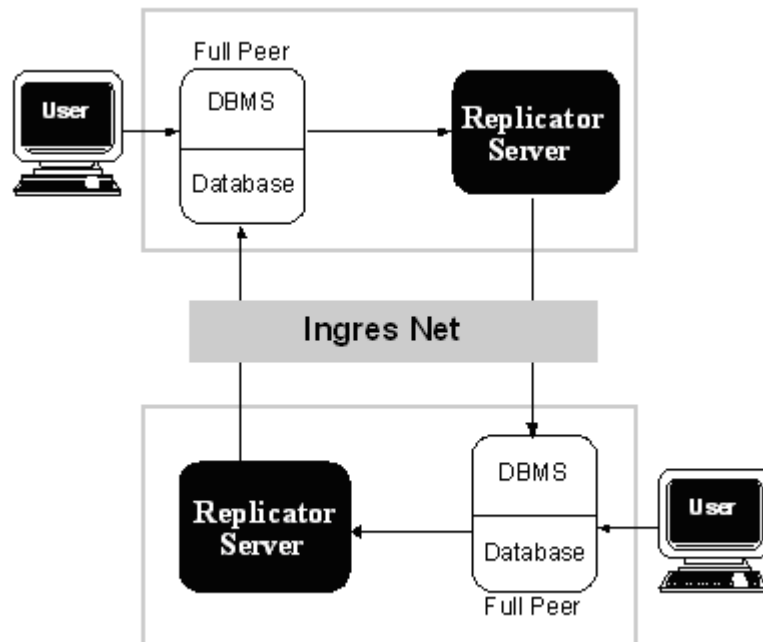
There are a number of possible replication schemes that you can use to propagate transaction data throughout your distributed processing environment. The way you choose to design Ingres Replicator on your network depends on the needs of your business and the purpose of your application.

## Peer-to-Peer Configuration

In peer-to-peer replication, illustrated in the following figure, each Ingres Replicator site plays a sending and receiving role on a peer basis with other target databases. There is no hierarchical relationship between the various servers.

The peer-to-peer arrangement implies that each Ingres Replicator site is functioning autonomously, using its own replicated database. All sites have the same information by having their own copies of an enterprise database.

**Note:** Although the following figure shows only two databases, a peer-to-peer scheme can encompass more than two, with each database transmitting directly to each of the other databases.



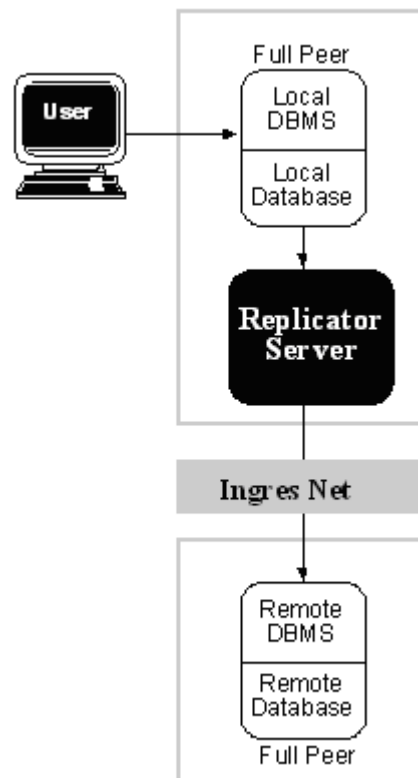
## Central-to-Backup Configuration

In a central-to-backup or standby design scheme, illustrated in the following figure, if a source database fails, an emergency backup database is available to resume activity.

Using central-to-backup with a peer-to-peer design scheme allows Ingres Replicator to ensure data is replicated to the failed database upon recovery; consistency is thereby restored across databases. Without a peer-to-peer scheme, a DBA must use a copy from a read-only backup and perform manual recovery.

**Note:** A two-database peer-to-peer backup (or standby) scheme is identical to a two-database peer-to-peer, except that in the standby scheme the backup database is normally not accessed by users.

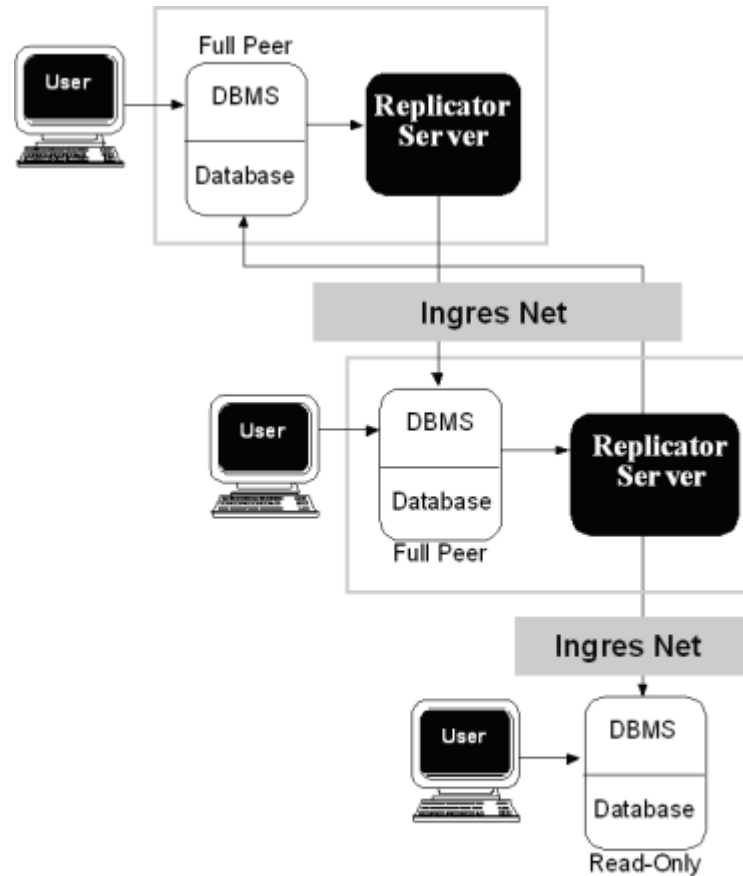
The central-to-backup configuration can maximize database availability across an enterprise without sacrificing fault tolerance. A central-to-backup configuration can also be used for continuous (24 x 7) operation. When database maintenance tasks such as recreating indexes are necessary on the main database, the users can be switched to access the secondary copy.





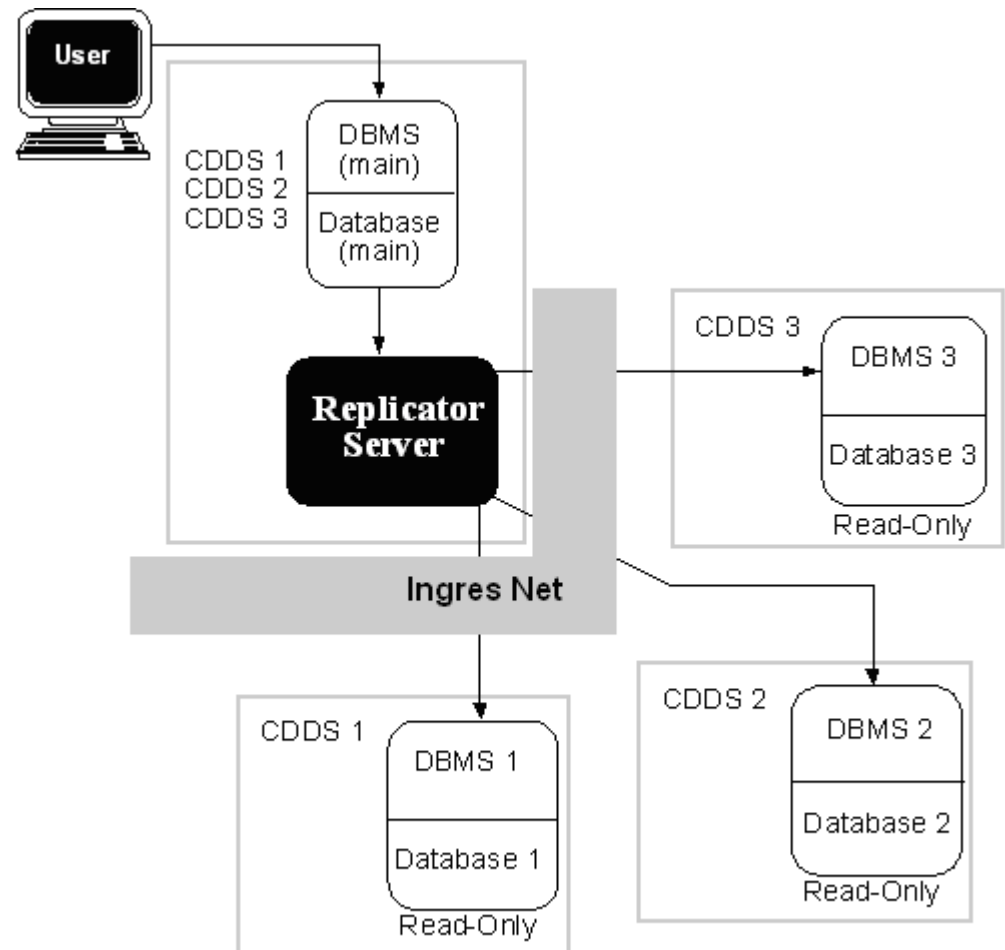
## Cascade Configuration

In a cascade replication, illustrated in the following figure, data from a source database is moved to a target database. That target database in turn moves the data to yet another target database. Every database that propagates the replication to another database must be designated as full peer. This scheme is useful in that the source database does not have to be responsible for distributing the data it creates throughout the entire replication system.



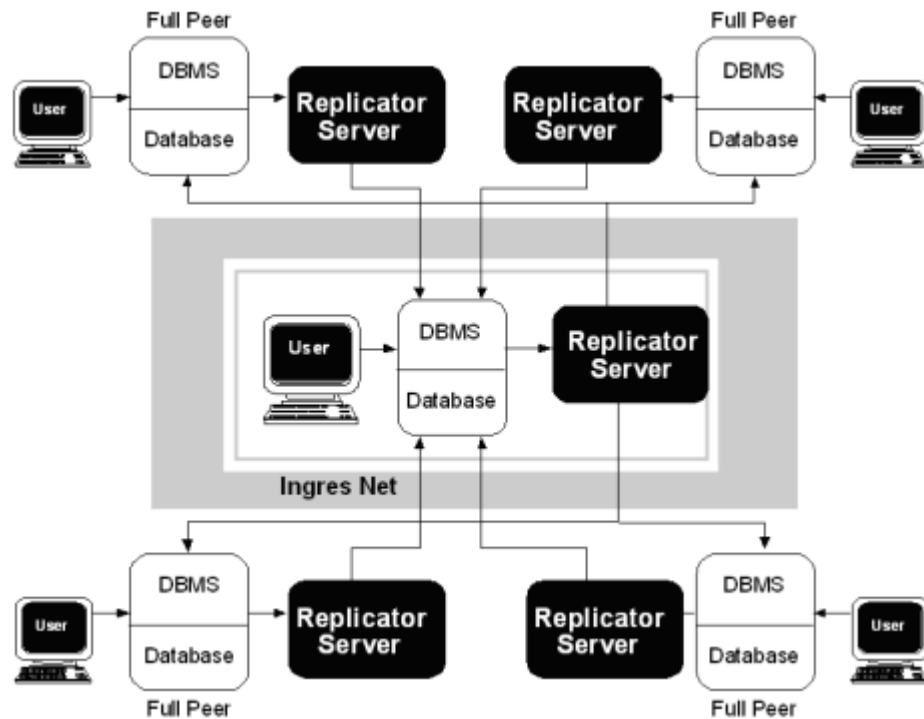
## Central-to-Branch Configuration

In a central-to-branch scheme, illustrated in the following figure, subsets of a central database are designated for replication to branch sites. The diagram shows read-only targets, but central-to-branch is often combined with peer-to-peer. In this arrangement, branches can update information from headquarters or create their own data and send the new data back to headquarters.



## Hub-and-Spoke Configuration

In a hub-and-spoke scheme, illustrated in the following figure, the hub database has a peer-to-peer relationship with each of its spokes. Implicit in this relationship is a cascade replication, in which each of the spokes receives replicated data whenever the hub database or any of the other spoke databases are manipulated.

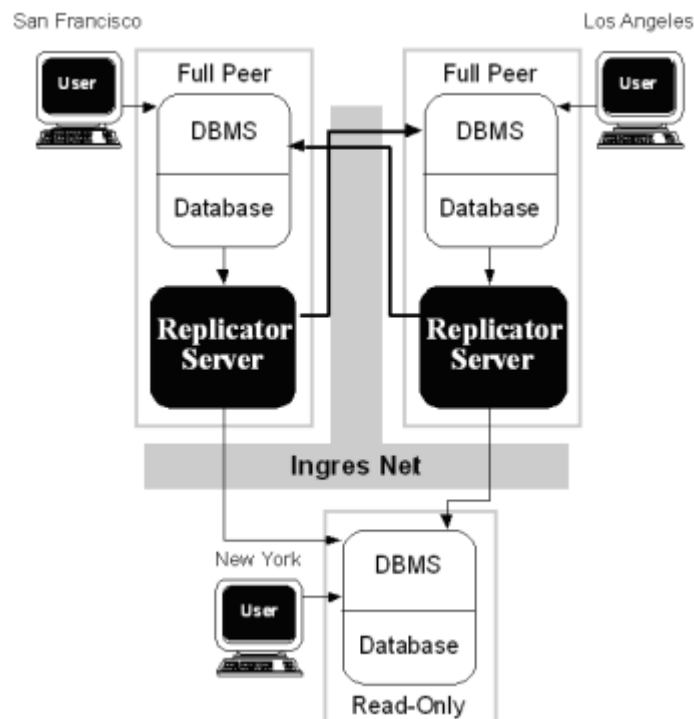


## Combination Configurations

Because no one technique necessarily accommodates all the needs of your environment, you may need to use more than one design technique.

For example, if you have a site that uses data for decision support applications, but requires no updates to data, you can use a fault-tolerant peer-to-peer and cascade scenario, as shown in the following illustration.

Data from San Francisco is replicated to Los Angeles on a peer-to-peer basis and to New York on a protected read-only cascade basis. Data from Los Angeles is replicated to San Francisco on a peer-to-peer basis and to New York on a protected read-only cascade basis. In the event of a failure at New York, decision support is performed at either San Francisco or Los Angeles with a degree of performance impact.



## Responsibilities of the Distributed Database Administrator

In conventional DBMS installations, a database administrator (DBA) oversees the DBMS at a designated local site and is the owner of the databases created within the DBA's account.

As a distributed database system, Ingres Replicator can span many local and remote locations. Using Ingres Replicator not only requires local DBAs to maintain their own part of the system, including the replicated databases that they own, but also requires a *distributed* database administrator to:

- Design and plan the replication system, including how and when data is shared among its users. Only after the distributed DBA has developed a complete design plan can Ingres Replicator be configured successfully.
- Coordinate the installation and re-configuration of the system among its different sites.
- Monitor and oversee the system at an enterprise level, rather than a local level.



# Chapter 3: Planning Your Replication Scheme

---

This section contains the following topics:

[Replication Planning](#) (see page 47)  
[Application Design Issues](#) (see page 47)  
[Replication Scheme Design Issues](#) (see page 52)  
[How You Implement Replicator](#) (see page 62)  
[How You Prepare for Replication](#) (see page 63)  
[Replication Scheme](#) (see page 66)  
[Replication Scheme Planning Worksheets](#) (see page 66)  
[Replication Scheme Examples](#) (see page 78)

## Replication Planning

The key to success with Ingres Replicator is careful planning. There are a number of possible replication schemes that you can use to propagate transaction data throughout your distributed processing environment. The way you choose to design replication on your network depends on the needs of your business and the purpose of your application.

## Application Design Issues

When you design a distributed replicated database, you must consider the type of data that is contained in the database, how that data is being used to process information, and which data is owned by a database.

This section defines these considerations and identifies the issues that can affect the accuracy, reliability, and efficiency of your replication system:

- Derived information
- Data ownership
- Data aggregation

## Derived Information

Derived information is information that is derived or calculated from other information in the database. In relational terms, this redundant information makes the table it is in de-normalized. For example, a bank balance is information that can be derived. Instead of adding up all the deposits and withdrawals from the time the account was opened, a bank can maintain a balance in the account record and manipulate the balance every time a transaction occurs on the account. Although derived data can be an efficient way to manage computing resources in a non-replicated environment, you must be aware of the risks of using derived information in a replicated environment.

### Example: Risks of Replicating Derived Data

The following scenario shows the problems of using derived data to obtain bank balances in a replicated environment. Assume there is a banking application with three replicated database copies. There is one database copy per branch, designated respectively as Databases A, B, and C.

On Monday afternoon, a customer opens an account at Branch A and deposits \$100.

As of Tuesday morning, the customer's account balance is reflected correctly in Databases A, B, and C, which show a balance of \$100.

A	B	C
\$100	\$100	\$100

On Tuesday at 9:00 am, the customer visits Branch C. At this time data communications is disrupted between Branch C and the other branches. The customer withdraws \$50 from the account.

The local database at Branch C commits the transaction at 9:05 am, showing a balance of \$50. The account balance is queued for distribution to Branches A and B.

A	B	C
\$100	\$100	\$50



The customer drives across town to Branch A. There, the customer withdraws another \$50 from the account.

Due to the communications problem, the account balance at Branch C has not yet arrived at Branch A. The local database at Branch A commits the transaction at 9:20 am. The Branch A database shows a balance of \$50 in the customer's account. The account balance at Branch A is queued for distribution to Branches B and C. Branch B receives the update from Branch A and changes the customer's balance to \$50.

<b>A</b>	<b>B</b>	<b>C</b>
\$50	\$50	\$50

The result of these steps is that all three branches end up showing that the customer has a balance of \$50 in the account. However, the true balance must be zero.

Assume now that the communication problem at Branch C is repaired on Tuesday at 9:30 am. The Replicator Server at Branch C detects a collision when it attempts to update the customer's balance at Branch A, and again at Branch B. Similarly, the Replicator Server at Branch A detects a collision at Branch C.

Each Replicator Server's response to collision is dependent on the way the CDDS containing the balance information is configured. The collision can be resolved in one of the following ways:

- Branches A and C abandon their attempts to distribute changes. The customer's balance at Branches A and B remains \$50 with a timestamp of 9:20 am. The customer's balance at C remains \$50 with a timestamp of 9:05 am. Data at all three sites is incorrect and inconsistent.
- Branches A and C distribute their changes. The customer's balance at both Branches A and B is \$50 with a timestamp of 9:05 am. The customer's balance at Branch C is \$50 with a timestamp of 9:20 am. Data at all three sites is incorrect and inconsistent.
- Branch A's change which has a later timestamp prevails over Branch C's. The customer's balance is \$50 with a timestamp of 9:20 am at all three branches. Data is incorrect, but consistent.

The problem in these resolution scenarios is that information is lost (or at best temporarily misplaced if we assume that the archive tables or log files contain the missing information). Even though the customer performed two transactions, the collision handling permitted the recording of only one transaction. The aggregate result of the two transactions is therefore overlooked. A possible solution to this problem is shown in Example 2.

**Example: Avoiding Risks in Replicating Derived Data**

The potential risk of replicating derived data, as shown in the previous example, can be avoided if each branch maintains its own balance on its own local database, as shown in the following example.

The customer withdraws \$50 from the account at Branch C. Data communications are disrupted to all other branches.

<b>A</b>	<b>B</b>	<b>C</b>
\$100	\$100	\$50

Branch C queues the \$50 transaction for replication to Branches A and B.

The customer withdraws \$50 from Branch A. Branch A queues the \$50 transaction for replication to Branches B and C. The transaction from A is replicated to B.

<b>A</b>	<b>B</b>	<b>C</b>
\$50	\$50	\$50

The customer's balance is now \$50 in both Branches A and B. When the network comes up for Branch C, Branch A and B are informed of the \$50 withdrawal that the customer made at Branch C. Branches A and B can update their balances. Branch A informs Branch C about the customer's other \$50 withdrawal transaction. All branches show the correct balance of \$0.

<b>A</b>	<b>B</b>	<b>C</b>
\$0	\$0	\$0

In this example, the transactions are replicated in the form of deposits and withdrawals. Because the balance is not being replicated, the replication does not collide on the balance table.

## Data Ownership

In designing a replication system, consider allowing each location ownership over specific data. Data ownership can reduce Ingres Replicator maintenance by eliminating collisions and keeping databases synchronized.

For example, assume that there is a sales organization in one location and a warehouse in another location. The warehouse owns the inventory tables and all information about shipping, while the sales organization owns the sales order records, except for shipping data. In this case, assume also that the sales organization needs to know the state of the inventory and the warehouse needs to know about sales orders. These two sets of information react to each other, but do not manipulate each other's data.

Following is the replicated process flow for this scenario:

1. A salesperson reviews a replicated copy of the inventory summary information, determining that it is possible to sell three widgets. The salesperson makes the sale and enters the sales order for three widgets, without manipulating inventory.
2. The sales orders are replicated back to the warehouse. On arrival of the sales order in the warehouse database, the following actions occur:
  - a. Rules execute database procedures to generate the required packing slips.
  - b. Database events alert the server responsible to print the new packing slips.
  - c. A database procedure adjusts the inventory table to subtract three widgets from the total number.
  - d. The sales order records are updated with the shipping information.
3. Sales order records are replicated back to the sales organization so those customers can be informed of their forthcoming widget shipment.

Because salespersons can only manipulate their sales orders, and the servers at the warehouse can only update inventory information, no replication collisions are possible. If a database is unavailable, all other databases can continue business. When the failed database is again in service, it is automatically brought back in synch with other databases and is collision free.

## Data Aggregation

Data aggregation prevents you from replicating unnecessary information and allows you to maximize your computing resources. Assume that corporate headquarters wants an overall total of each store's revenue for the day. You can run a program at the end of the day that performs an insert into a daily totals table. This replicated table can be distributed to corporate headquarters. The alternative, which is to replicate each transaction and send its value to headquarters, is resource intensive.

## Replication Scheme Design Issues

You must consider the following while designing your replication scheme:

- Collision design
- Error handling
- Assigning Replicator servers
- Storage structures

## Collision Design

In a multiple database environment, there is always the possibility of two users updating the same record simultaneously on two different databases. When creating a replicated system, consider designing the system to avoid or at least minimize collisions.

To reduce collisions significantly or eliminate them altogether, design transactions that allow newly created information to be presented in the form of inserts and deletes instead of updates. This is particularly true for updates to key columns.

For information on how to resolve collisions, see Collision Resolution (see page 182).

## How Collisions Are Caused

A *collision* is an event that occurs when simultaneous changes are made in two different databases to rows with *identical* keys. The Replicator Servers detect this collision condition when the data is transmitted between the two databases.

In a collision condition, Ingres Replicator cannot synchronize the databases without destroying information.

There are five possible situations that cause collision between a source and target database. Collisions can occur during an insert, update, or a delete transaction and is detected in these instances:

- Insert transaction detects a duplicate key.
- Update transaction is attempted on a record that no longer exists.
- Update transaction is attempted on a record that does not match the original.
- Delete transaction is attempted on a record that no longer exists.
- Delete transaction is attempted on a record that does not match the original.

**Note:** If you use referential integrity constraints in a distributed environment, you create collision-like conflicts that cannot be resolved. For example, assume a customer record is deleted in one database while at the same time a new order for that customer is entered in a different database. If there is a referential constraint requiring orders to reference existing customers, there is an exception when the order is replicated back to the first database.

The best defense against a collision is prevention. Design your replication system to reduce the probability of collisions. Even in well-designed databases, collisions can occur during a system failure when it becomes necessary to switch between replicated databases. For this reason alone, you need to plan how to handle collisions in your replication system.

## Collision Handling

Collision handling happens at the CDDS level; when defining your CDDS you must specify which collision mode you want to use. There are two ways to handle collisions, automatically or manually. Each method has advantages and disadvantages. Automatic resolution takes less time but can produce unexpected results. Manual resolution can give you more control and, in some cases, is the only way you can resolve a conflict.

All collisions are counted as errors; they cause the Replicator Server to log the error, increase the error count, and send e-mail to the mail notification list. If the collision is resolved, the Replicator Server continues processing. If the collision is not resolved, the Replicator Server behaves in accordance with the error mode setting.

## Automatic Resolution of Collisions

With automatic resolution, when two records collide, one record prevails over the other. If the operation was an insert or an update, a record survives a collision by overwriting the record in the target database (the target row is deleted and the prevailing source row is inserted in its place). If the transaction was a delete, the record in the target database is deleted. If a record does not survive a collision, its replication operation (insert, update, or delete) for that target is removed from the distribution queue.

**Note:** Automatic resolution overwrites the entire record and can overwrite columns that have correct information with columns that do not have correct information. You must not use automatic resolution if the information contained in the losing record is important. For example, if your database contains documents that are continually updated with new information, you can lose information with automatic resolution.

## Manual Resolution of Collisions

You can resolve collisions manually by editing the base table, the shadow table, and the distribution queue. For instructions, see *How You Resolve Collisions Manually* (see page 184). You can also resolve collisions manually using Visual DBA. For step-by-step instructions for resolving collisions manually, see the Visual DBA online help.

## Collision Modes

How a Replicator Server acts when it detects a collision depends on the collision mode of the CDDS to which the colliding records belong. The Passive Detection mode involves no action on the part of the Replicator Server. Modes other than Passive Detection require the Replicator Server to search the target database to see if the row it is propagating already exists. If a Replicator Server detects a collision, its subsequent actions are determined by the collision mode and error mode assigned to the CDDS to which the colliding row belongs.

The available collision modes are:

- **Passive Detection**

(Default) This mode detects insert collisions and update and delete collisions where the row to be updated or deleted does not exist. The Replicator Server does not resolve the collision. The collision is an error and the Replicator Server action is dependent on the error mode setting.

This mode requires you to perform manual collision resolution.

- **Active Detection**

This mode detects all collisions, but does not resolve them.

Before propagating the row, the Replicator Server searches the target database to see if the row already exists. If it detects a collision, however, the Replicator Server does not resolve it. The collision is an error and the Replicator Server action is dependent on the error mode setting.

This mode requires you to perform manual collision resolution.

- **Benign Resolution**

This mode detects and resolves benign collisions (when the same row with the same transaction ID arrives at the same target more than once).

Before propagating the row, the Replicator Server searches the target database to see if the row already exists. If the row does exist and it came from the same transaction, the Replicator Server issues a warning message and removes the operation from the distribution queue. Otherwise, the server action is dependent on the error mode setting.

This mode requires you to perform manual collision resolution on all but benign collisions.

- **Priority Resolution**

This mode detects and resolves all collisions according to assigned priorities.

Before propagating the row, the Replicator Server searches the target database to see if the row already exists. If the Replicator Server detects a collision, it resolves it by comparing the priority numbers assigned to the underlying rows. The row with the highest priority number prevails. If the priorities are the same or do not exist, the row with the lower database number survives the collision.

Priority numbers are assigned in a priority collision resolution lookup table. For instructions for creating this lookup table, see chapter "Using Advanced Features."

In this mode, all collisions are resolved automatically.

- Last Write Wins

This mode detects and resolves all collisions according to transaction timestamps.

Before propagating the row, the Replicator Server searches the target database to see if the row already exists. If the Replicator Server detects a collision, it resolves it by comparing the transaction timestamps. The row with the later timestamp prevails.

If the timestamps are *identical*, the row with the lower database number survives the collision.

In this mode, all collisions are resolved automatically.

## How Errors Are Handled

Propagation error handling is specified at the CDDS level; when defining your CDDS you must specify which propagation error mode you want to use. The method the servers use to handle an error detected while transmitting information depends on the error mode.

For all propagation error modes, when a server detects an error it does the following:

- Logs the error, thereby increasing the error count  
For more information on the role of the server in error handling, see Error Handling (see page 139).
- Issues e-mail messages to any users on the mail notification list
- For more information on the mail notification list, see Create a Mail Notification (see page 130).

**Note:** E-mail error notification is not available on Windows. Using the -NML server flag also turns off error notification. For more information, see the chapter "Using the Replicator Server."



## How Error Modes Affect Server Behavior

Server behavior differs for each propagation error mode.

**Note:** In the following descriptions, the “current replication transaction” is in the context of the Replicator Server, which can disagree with the original user transaction if the latter updated more than one CDDS.

Error modes and how they affect server behavior are described here:

- Skip Transaction

(Default) The Replicator Server:

- Rolls back the current replication transaction
- Processes the next transaction
- Retries the transaction in error during the next processing of the distribution queue

- Skip Row

The Replicator Server continues from the error with no rollback performed. The record in error is skipped and processing proceeds to the next record. The record in error is removed from the queue.

**Note:** This setting disables transaction consistency, therefore data integrity needs to be maintained manually.

- Quiet CDDS

The Replicator Server rolls back the current replication transaction. After the rollback of the transaction in error, the Replicator Server quiets the CDDS on the database where the transaction error occurred. (In this context, *quiets* means disabling propagation from the local database to the target database where the error occurred.)

The Replicator Server continues processing the remaining replicated transactions for the same CDDS on other databases, and for other CDDSs.

For information on reactivating the CDD, see chapter “Using the Replicator Server.”

- Quiet Database

The Replicator Server rolls back the current replication transaction. After the rollback of the transaction in error, the Replicator Server quiets all CDDSs in the database where the transaction error occurred. (In this context, *quiets* means disabling propagation from the local database to the target database where the error occurred.)

The Replicator Server continues processing the remaining replicated transactions for other databases.

For information on reactivating the database, see chapter “Using the Replicator Server.”

- Quiet Server

The Replicator Server:

- Rolls back the current replication transaction
- Stops processing of the remaining batch of replicated transactions
- Changes its status from active to quiet mode

## Replicator Server Assignment

You must assign Replicator Servers on full peer databases to transmit replications to targets. Protected and unprotected read-only databases do not need Replicator Servers because they do not transmit information. They do, however, need server numbers assigned to them so that they can receive information.

Assign a Replicator Server number for each database in your replication scheme according to the following guidelines:

- The number must be in the range of 1-999 and must correspond to the name of the directory where the server resides. When you set up the Ingres Replicator software, ten server directories (named server1 through server10) are created. You can add more server directories if necessary. For instructions, see chapter “Using the Replicator Server.”
- The number must be unique within an Ingres Replicator installation but can be the same as a number on another installation.

## Replicator Server Scenarios

The following are Replicator Server assignment scenarios that you can use:

- Single server

Each local database can have a single Replicator Server that services all targets. If the replication traffic is heavy, performance can be affected.

- Different server for every CDDS

Each full peer database can have a Replicator Server for each CDDS. This allows each CDDS to have a different priority (at the operating system level) and different scheduling. Also, if there is a problem with one server, the whole system is not necessarily affected.

- Different server for every target database

Each target that the full peer database transmits can have its own server. This further localizes transmission problems that can occur and reduces traffic on individual servers. Many servers can improve performance over the network, however, each server uses resources. For an example of using a different server for every target database, see Example 1: R.E.P.'s CDDS 0 (see page 78).

- Different server for every target database and CDDS

Even if the same target database exists in more than one CDDS, it has a different server transmitting to it for every CDDS it participates in. For an example of using different servers for every target database and CDDS, see Example 3: R.E.P.'s Server Assignments (see page 80).

## Storage Structures

In Ingres Replicator, deadlock can occur when tables involved in a query are replicated. When you update the base table, the DBMS must insert a record in the shadow table, possibly the archive table, and the input queue table within the same transaction. The additional inserts can cause deadlock.

The following example shows how deadlock can occur with just two sessions:

1. User A inserts a record into page 10 of the base table; this locks page 10 of the base table, page 5 of the shadow table, and page 8 of the input queue table to maintain the necessary support tables. The insert is completed and the user holds the transaction—and therefore the locks—open.
2. User B inserts a record into page 11 of the base table; this locks page 11 of the base table and also requires a lock on page 5 of the shadow table to insert the shadow record. The lock on page 5 of the shadow table cannot be obtained, so the session waits for it.
3. Meanwhile, user A tries to insert a record into page 11 of the base table. This insert waits for the lock, which user B holds on the base table. Hence, deadlock. User A needs the lock on page 11 of the base table that user B holds, and user B needs the lock on page 5 of the shadow table that user A holds.

The first part of the primary key on both the shadow table and the input queue table is the `transaction_id` column. This column contains the low part of the Ingres internal 8-byte transaction ID. This transaction ID is incremented by 1 each time a new transaction—which can be user or system internal—is started in the Ingres installation. This means that a nearly monotonically increasing primary key value is generated for each replicated update.

The default structure for the shadow and archive tables is B-tree. Because the primary part of the key is always increasing, every insert to the support tables causes a ladder lock down the right side of the B-tree index, resulting in a lock on the far right leaf page to insert the new index value. This structure therefore makes it likely that all users want to lock the same pages at the same time.

The column ordering on which Ingres Replicator support tables is indexed cannot be changed because access using these keys is built into the internals of the DBMS (much like the Ingres standard system catalogs, of which the Ingres Replicator support tables are just the visible extension). This makes the alteration or re-ordering of key columns impossible.

To avoid deadlock, the solution is to make sure the sessions do not try to lock the same pages in the same tables at the same time. To achieve this, you need some understanding of the keying and table structures of the Ingres Replicator support tables. For detailed information and examples of how to use locking parameters to avoid deadlock, see *Strategies for Avoiding Deadlock* (see page 210). The other means of avoiding deadlock is to use storage structures. For more information, see *Using Hash Structures to Avoid Deadlock* (see page 61).

## Using Hash Structures to Avoid Deadlock

One option to avoid deadlock is to modify the table structure from B-tree to hash. Hash is the best structure to use to avoid the problem of users requiring locks on the same pages. If the modify creates enough hash buckets (minpages), each new insert is hashed to a different bucket, and users never require locks on the same pages.

The actual value of minpages used in the modify statement depends on the number of concurrent users and the volume and frequency of updates to the replicated tables. The value must be as large as possible even though more disk space is needed when the table is modified. The table does not grow in size until more records are inserted that hash to a single page than fits in that page; this event becomes less likely with more hash buckets, which offsets the additional disk space requirement.

The shadow and archive tables present a different situation. Records are continually inserted into these tables for each insert (shadow only), update, and delete (both shadow and archive) on the base replicated table. None of the inserted records are automatically removed, which means that the tables are continually growing in size (until the arcclean utility is used to remove rows that are no longer needed). This causes a problem with hash tables because no matter how many hash buckets are provided, they are eventually filled, and the tables contain unwanted overflow pages. Because each insert must lock the entire overflow chain for the relevant bucket, the longer the chain becomes, the more likely it is that the insert causes lock contention (and therefore possible deadlock).

The key to the efficient use of hash tables is planning. You need to calculate the likely volume of replicated updates for each table in a day and ensure that there are enough spare hash buckets to handle any new additions. The tables must be remodified each night to remove any overflow and increase the number of hash buckets for the next day's use. The shadow and archive tables must also be cleaned out periodically when a logical consistency point (for replicated databases) can be found and while replication is disabled. If there are situations where this type of maintenance is impossible, the row-locking alternative described in the chapter "Using Advanced Features" must be investigated.

Keep in mind that a hash table is more efficient for updates than a B-tree table (provided there is no unwanted overflow); for performance considerations, you must try to use hash tables. Remember that records are auto-deleted from the input queue table, so there is no reason for using row locking on this table. The only exception is where the distribution threads are allowed to lag behind the user update (because of insufficient threads or the value of `rep_txq_size` is set too low—neither of which is desirable).

## How You Implement Replicator

The process for getting Ingres Replicator up and running is as follows:

1. Prepare the databases by making sure they conform to conventions and by defining Ingres Net virtual nodes.

See the chapter "Planning Your Replication Scheme."

2. Plan your replication scheme by filling out the Database Worksheet, the CDDS Worksheet, and the optional Table Worksheet.

See the chapter "Planning Your Replication Scheme."

3. Configure Ingres Replicator CDDSs using Visual DBA or Replicator Manager.

See the chapters "Configuring Replication Using Visual DBA" and "Configuring Replication Using Replication Manager."

4. Move the configuration to target databases.

See the chapters "Configuring Replication Using Visual DBA" and "Configuring Replication Using Replication Manager."

5. Configure the servers.

See the chapter "Using the Replicator Server."

6. Activate Change Recording.

See the chapters "Configuring Replication Using Visual DBA" and "Configuring Replication Using Replication Manager."

7. Start the servers.

See the chapter "Using the Replicator Server."

## How You Prepare for Replication

You need to perform these preliminary tasks to prepare for replication:

- Ensuring that your databases conform to Ingres Replicator rules for database objects
- Assigning the appropriate privileges to the DBAs who own the databases to be replicated
- Defining Ingres Net entries for each database

**Important!** *The first time you implement a replication scheme, use test databases. Do not configure Ingres Replicator on production tables before verifying the validity of your scheme. Propagation path configuration errors can result in lost data.*

### Rules for Database Objects

When creating database objects in an *Ingres Replicator* environment, be aware of the following restrictions:

- Replicated table and columns names must each be identical in each database. For example, a table named emp in database A can only be replicated to the table emp in database B, not to the table employee.
- Table names cannot begin with ii or conflict with Ingres Replicator catalogs. For more information, see the appendix "Data Dictionary Tables."
- Tables cannot have column names that begin with \_ii\_.
- Table names cannot exceed 32 bytes in length.
- Column names cannot exceed 28 characters in length.

**Note:** If you plan to replicate into a gateway database, limit the table and column name lengths to the OpenSQL-recommended standard of 18 characters.

- Column names cannot be the same as the column names Ingres Replicator uses as parameters in database procedures or columns in the shadow table. These column names include table\_no, sequence\_no, sourcedb, cdds\_no, transaction\_id, trans\_time, s\_new\_key. For more information, see the appendix "Data Dictionary Tables."
- Each table to be replicated must have a PRIMARY KEY or UNIQUE column(s), a UNIQUE primary storage structure or a UNIQUE secondary index on one or more columns (up to a maximum of 31 columns). The column or columns that are part of the unique storage structure or index cannot be defined as nullable.

- Columns defined as replicated keys must be designated not null not default.
- Total width of the columns to be replicated is limited by the overhead of the archive and shadow rows. The archive table adds 10 bytes to the base row. Therefore, if you are using a 4 KB page size, the maximum base table row width is 3978. The shadow table adds 55 bytes to the base table primary key columns. These limits are not applicable if long varchar or long byte data types are used. For more information, see the chapter “Calculating Disk Space” in the *Database Administrator Guide*.
- You can define up to 32,767 databases per replicated system.
- Replication of system catalogs is not supported.
- Abstract data type date is not supported for gateway targets on certain platforms. Check your product documentation for details.
- Abstract data type money is not supported for Enterprise Access targets because money is not an OpenSQL data type.
- Replication of tables with an identity column used as the replication key is not supported between full peer databases.

Ingres Replicator cannot replicate:

- User-Defined Data Types (UDTs), including spatial data types.
- Tables with any of the following data types as the table key: float, money, system\_maintained table\_key, or system\_maintained object\_key.

## Database Administrator Privileges

The DBA responsible for Ingres Replicator operations needs the trace privilege.

If you want to replicate tables owned by users other than the DBA, the DBA must be granted privileges for select, insert, update, and delete operations with the grant option at all Full Peer and Protected Read-only targets.

The DBA of each database that contains a full peer or protected read-only CDDS must have the security privilege or be granted database administrator privileges in all other full peer or protected read-only targets within the replication system.



## Define the ingrep Role

If the DBAs of the source and target databases are different, before a Replicator Server is started, the ingrep role must be created in the iidbdb database of every installation that has a database participating in the replication scheme and that contains a full peer or protected read-only CDDS. For more information on creating a role, see the *SQL Reference Guide*.

Installations running Ingres must also grant the ingrep role to the DBA by executing a grant statement in the iidbdb, as follows:

```
grant ingrep to public
```

or

```
grant ingrep to user replicator_dba
```

where:

***replicator\_dba***

Is the database administrator for Replicator.

## Ingres Net Entries

You must have Ingres Net installed on every machine in your replication scheme. Ingres Replicator uses Ingres Net to communicate between databases on different installations and different machines. Each database that is in a different database installation must have an Ingres Net entry. For detailed instructions on installing Ingres Net and creating connection data entries, see the *Connectivity Guide*.

Make a note of the Ingres Net virtual node name, which you need when you fill out the Database Worksheet.

Ingres Net allows you to control authorization to databases. For a discussion on how Ingres Net ensures security on your system, see Security for Replicator (see page 17).

## Replication Scheme

A replication scheme is the set of replication definitions for a database. These definitions identify the following:

- Consistent Distributed Data Sets (CDDs)
- Data propagation paths
- CDD target types

Your replication definitions are codified when you configure your Ingres Replicator installation.

## Replication Scheme Planning Worksheets

You need an accurate representation of your replication scheme to accurately configure Ingres Replicator.

The planning worksheets help you to represent your replication scheme. When you are ready to start configuring Ingres Replicator, these worksheets will already contain most of the information you need to select appropriate options in configuration screens.

We recommend that you fill out the following worksheets:

- Database Worksheet, which contains a list of the databases in the replicated environment.
- CDD Definition Worksheet, which contains information about the CDDs that you need when configuring Ingres Replicator.
- Table Worksheet (optional), which contains table and column information. You can use this worksheet if you need to list table information or if you are planning vertical or horizontal partitioning.

The following table is a checklist of the tasks described in this section:

Task	Check
<b>Fill in the Database Worksheet. Assign values for:</b>	
Database numbers	
<b>Fill in the CDDS Worksheet. Assign values for:</b>	
CDDS number	
CDDS name	
Collision mode	
Error mode	
CDDS target type	
Replicator Server numbers	
Propagation paths	
<b>Fill in the Table Worksheet (if necessary)</b>	

This chapter uses a sample replication scheme, which is also used in the chapters "Configuring Replication Using Visual DBA" and "Configuring Replication Using Replicator Manager." For examples of other replication schemes and their worksheets, see Replication Scheme Examples (see page 78).

## Database Worksheet

The Database Worksheet defines the distributed environment for all CDDs in your replication scheme. The following table describes the guidelines for filling in the Database Worksheet.

Value	Guidelines												
Database Number	<p>This number is an enterprise-wide unique integer in the range of 1-32,767.</p> <p>Assign database numbers carefully; Priority Resolution and Last Write Wins collision modes can use the database number as the basis for deciding priority when resolving collisions. We recommend assigning database numbers in increments of ten, so that new databases can be added in between the original ones.</p>												
Database Vnode/Name	<p>Database names must follow DBMS guidelines; Ingres Replicator does not impose further restrictions.</p> <p>The vnode is the Ingres Net virtual node name for the database. For more information on vnodes, see the <i>Connectivity Guide</i>.</p>												
Database Owner	The owner (DBA) of the database.												
DBMS Type	<p>The DBMS Type can be any one of the following:</p> <table><tr><td>CA-Datcom/DB</td><td>Microsoft SQL Server</td></tr><tr><td>CA-IDMS</td><td>Oracle</td></tr><tr><td>IBM DB2</td><td>OpenVMS RMS</td></tr><tr><td>IBM DB2 UDB</td><td>Star</td></tr><tr><td>IMS</td><td>Sybase</td></tr><tr><td>Ingres</td><td>VSAM</td></tr></table> <p>For more DBMS types, check the Readme file.</p>	CA-Datcom/DB	Microsoft SQL Server	CA-IDMS	Oracle	IBM DB2	OpenVMS RMS	IBM DB2 UDB	Star	IMS	Sybase	Ingres	VSAM
CA-Datcom/DB	Microsoft SQL Server												
CA-IDMS	Oracle												
IBM DB2	OpenVMS RMS												
IBM DB2 UDB	Star												
IMS	Sybase												
Ingres	VSAM												
Remarks	User-specified, up to 80 characters.												

The following is an example of a Database Worksheet.

Database Number	Database Vnode/Num	Owner	DBMS Type	Remarks
<b>Comments</b>				

The following figure illustrates a completed Database Worksheet.

Database Worksheet				
Database Number	Database Vnode/Name	Owner	DBMS Type	Remarks
10	ny::ha	rep-dba	Ingres	Headquarters
11	sfo::west	rep-dba	Ingres	U.S. Western Region
12	dal::central	rep-dba	Ingres	U.S. Central Region
20	lon::eu/rope	rep-dba	Ingres	European Region
30	hkg::asia	rep-dba	Ingres	Asian Region

## Identifying the CDDS

Carefully examine the way data is used among the different databases in your system to determine your CDDSs. Remember that CDDSs are indivisible and mutually exclusive; you cannot split the information in a CDDS so that it goes to different targets, and you cannot contain the same data in more than one CDDS. If you later decide to divide a CDDS, you must reconfigure Ingres Replicator.

For the definition of a CDDS and examples, see Consistent Distributed Data Set (CDDS) (see page 32).

The CDDS Worksheet is divided into the following sections:

- CDDS summary
- CDDS diagram
- Database information
- Propagation paths

The CDDS Worksheet and a sample completed CDDS Worksheet are at the end of this section.

## CDDS Summary Information

The CDDS summary information defines the scope of the CDDS and how it handles collisions and errors. The following table describes how to fill in the CDDS summary section of the CDDS Worksheet.

Value	Guidelines
CDDS number	This number is an integer in the range of 0-32,767. The default number is 0.
CDDS name	This name is for your information only. Maximum length is 32 bytes. The default is Default CDDS.
Major tables	Fill in the names of the tables to be replicated. Use the Table Worksheet for more detail.
Collision mode	<p>The possible values are:</p> <ul style="list-style-type: none"> <li>■ PassiveDetection</li> <li>■ ActiveDetection</li> <li>■ BenignResolution</li> <li>■ PriorityResolution</li> <li>■ LastWriteWins</li> </ul> <p>For more information, see Collision Modes (see page 55).</p>
Error mode	<p>The possible values are:</p> <ul style="list-style-type: none"> <li>■ SkipTransaction</li> <li>■ SkipRow</li> <li>■ QuietCDDS</li> <li>■ QuietDatabase</li> <li>■ QuietServer</li> </ul> <p>For more information, see How Errors Are Handled (see page 56).</p>

## CDDS Diagram

Use the CDDS Diagram to create a visual representation of your replication scheme. This overview of your replication scheme gives you an instant understanding of your plan and can expose errors or omissions in your plan.

The CDDS Diagram consists of the following elements:

### Shapes (databases)

Label each database with its name, number, and target type.

### Arrows (propagation paths)

Draw a line for every path, and an arrow pointing at the target. Read-only targets have only incoming arrows, while full peer targets have incoming and outgoing arrows. Label each arrow with the number of the Replicator Server that is propagating to that target.

## Database Information

The database information describes the behavior of the database within the CDDS. The following table describes how to fill in the database information section of the CDDS Worksheet.

Value	Guidelines
Database Number/Name	The database number, vnode, and name from the Database Worksheet.
Target Type	<p>Possible values are:</p> <ul style="list-style-type: none"><li>■ Full peer</li><li>■ Protected read-only</li><li>■ Unprotected read-only</li></ul> <p>For a description of each target type, its behavior, and the criteria for its use, see CDDS Target Types (see page 36).</p>
Server Number	<p>The number of the Replicator Servers assigned to propagate transactions to the database.</p> <p>For more information, see Replicator Server Assignment (see page 58).</p>

## Propagation Paths

Based on the CDDS diagram, fill in propagation paths. For an explanation and examples of data propagation paths, see Data Propagation Paths in the CDDS (see page 34).



## CDDS Worksheet

Define worksheet entries so that an update that occurs on *any* of the full peer databases within the CDDS is propagated to *all* of the other databases that participate in the CDDS. For example, if you have two databases in the CDDS that act as full peers, each of them must be an originator in one propagation path and a target in another propagation path. The following table describes how to fill in the propagation path section of the CDDS Worksheet:

Value	Guidelines
Originator DB	Fill in the number of the database where the transaction originated
Local DB	Fill in the number of the database that propagates the transaction to the target
Target DB	Fill in the number of the database that receives the transaction
Comment	This is for worksheet purposes only

The following forms provide examples of CDDS Worksheets.

<b>CDDS number:</b> _____ <b>CDDS name:</b> _____		
<b>Major tables:</b> _____		
<b>Collision mode:</b> PassiveDetection   ActiveDetection   BenignResolution PriorityResolution   LastWriteWins		
<b>Error mode:</b> SkipTransaction   SkipRow   QuietCDDS   QuietDatabase QuietServer		
<b>CDDS diagram</b>		
<b>Database Summary</b>		
Database Number/Name	Target Type	Server Number

CDDS number: _____ CDDS name: _____			
<b>Propagation Paths</b>			
Originator DB	Local DB	Target DB	Comment

The following figure illustrates a completed CDDS Worksheet.

**CDDS Worksheet**

CDDS number: 1 CDDS name: Inventory

Major tables: book\_list

Collision mode: PassiveDetection ActiveDetection BenignResolution  
PriorityResolution LastWriteWins

Error mode: SkipTransaction SkipRow QuietCDDS QuietDatabase QuietServer

**CDDS Diagram**

**Database Summary**

Database Number/Name	Target Type	Server Number
10 NYC::hq	full peer	1
11 SFO::west	full peer	4
20 LON::europe	full peer	2

**Propagation Paths**

Originator DB	Local DB	Target DB	Comment
10	10	11	NYC to SFO
10	10	20	NYC to LON
11	11	10	SFO to NYC
11	11	20	SFO to LON via NYC
20	20	10	LON to NYC
20	20	11	LON to SFO via NYC

Table Worksheet

The Table Worksheet allows you to document your replicated table and column information in detail.

You must complete this worksheet if you are including vertical or horizontal partitioning in your CDDS. Vertical partitioning refers to replicating a subset of a table’s columns. Horizontal partitioning refers to replicating a subset of a table’s rows based on values in a particular column.

For information on how to use partitioning in your replication scheme, see Consistent Distributed Data Set (CDDS) (see page 32). For information on horizontal partitioning, see How You Set Up Horizontal Partitioning (see page 201).

The following figure shows an example of a Table Worksheet.

CDDS number: _____ CDDS Name: _____			
Table Name	Column Name	Replicated?	Horizontal Lookup Table

The following figure illustrates a completed Table Worksheet.

<b>Table Worksheet</b>			
CDD S number: <u>1</u>		CDD S Name: <u>Inventory</u>	
Table Name	Column Name	Replicated?	Horizontal Lookup Table
book_list	book_no	Y	N/A
	title	Y	
	author	Y	
	price	Y	
	category	Y	
	stock	Y	
	dist_no	Y	

## Replication Scheme Examples

This section provides three replication scheme examples, including samples of worksheet entries and corresponding diagrams. These examples are based on the 'Round the Earth Publishing (R.E.P.) scenario used in this chapter and in the chapter "Understanding Ingres Replicator." In these examples, the Company is not using horizontal partitioning.

### Example: R.E.P.'s CDDS 0

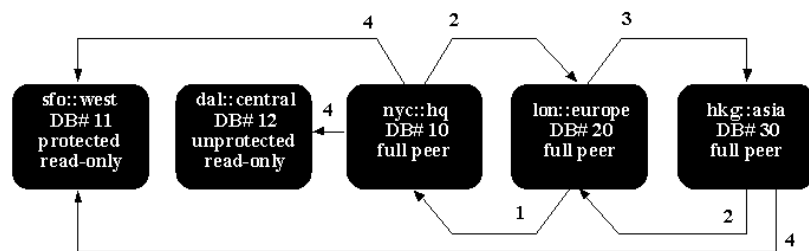
The Company's main CDDS (CDDS 0) is shared by its offices in New York City, San Francisco, Dallas, London, and Hong Kong. New York, London, and Hong Kong are full peers in this CDDS. San Francisco is a protected read-only target, and Dallas is an unprotected read-only target. The following table illustrates the database information on the CDDS Worksheet for this CDDS:

Database Number/Name	Target Type	Server Number
10 nyc::hq	1 (full peer)	1
11 sfo::west	2 (protected read-only)	4
12 dal::central	3 (unprotected read-only)	4
20 lon::europe	1 (full peer)	2

Database Number/Name	Target Type	Server Number
30 hkg::asia	1 (full peer)	3

New York acts as the U.S. hub and distributes data to San Francisco, Dallas, and London. London acts as the international hub and sends data to New York and Hong Kong. San Francisco is the U.S. alternate site and has a redundant path from Hong Kong.

The Company's CDDS Diagram for CDDS 0 looks like this:



The Company's Propagation Path entries on the CDDS Worksheet for CDDS 0 are listed in the following table:

Originator DB	Local DB	Target DB	Comment
10	10	11	NYC to SFO
10	10	12	NYC to DAL
10	10	20	NYC to LON
10	20	30	NYC to HKG through LON
10	30	11	NYC to SFO through LON and HKG
20	20	10	LON to NYC
20	20	30	LON to HKG
20	30	11	LON to SFO through HKG
20	10	12	LON to DAL through NYC
20	10	11	LON to SFO through NYC
30	30	11	HKG to SFO
30	30	20	HKG to LON
30	20	10	HKG to NYC through LON

Originator DB	Local DB	Target DB	Comment
30	10	11	HKG to SFO through LON and NYC
30	10	12	HKG to DAL through LON and NYC

Because the San Francisco database has a redundant path, it has collisions when the same transaction arrives from both New York and Hong Kong. The Company handles this situation by using the collision mode BenignResolution, which resolves collisions that have the same transaction ID.

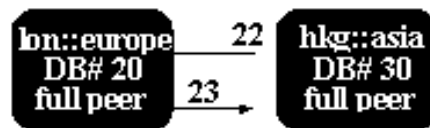
## Example: R.E.P.'s CDDS 2

The Company's London and Hong Kong offices share an international sales CDDS (CDDS 2), with each of them acting as full peers. The database information on the CDDS Worksheet for this CDDS is as follows:

Database No./Name	Target Type	Server Number
20 lon::europe	1 (full peer)	22
30 hkg::asia	1 (full peer)	23

The Replicator Server numbers assigned to each database are different than the Replicator Server numbers for CDDS 0. Therefore, the replications from CDDS 2 do not add traffic to the CDDS 0 Replicator Servers.

The Company's CDDS diagram for CDDS 0 looks like this:



The Company's Propagation Path entries on the CDDS Worksheet for CDDS 0 are as follows:

Originator DB	Local DB	Target DB	Comment
30	30	20	HKG to LON
20	20	30	LONG to HKG

## Example: R.E.P.'s Server Assignments

The Company decides that it wants to move the transactions for the inventory CDDS (CDDS 1) off the Replicator Servers that also handle transactions for the main CDDS (CDDS 0). The Company changes the Replicator Server assignments for CDDS 1.



### Database Information for CDDS 1

The new database information for the CDDS Worksheet for CDDS 1 is listed in the following table:

Database Number/Name	Target Type	Server Number
10 nyc::hq	1 (full peer)	11
11 sfo::west	1 (full peer)	14
20 lon::europe	1 (full peer)	12

The other sections of the CDDS Worksheet for CDDS 1 remain the same.

The Company now has different Replicator Servers for almost every target and CDDS.

### Server Assignments for CDDS 0

The Company must assign the Replicator Server numbers for CDDS 0 according to the following table:

Server Number	Database
1	nyc::hq
2	lon::europe
3	hkg::asia
4	both sfo::west and dal::central

### Server Assignments for CDDS 1

The Company must assign the Replicator Server numbers for CDDS 1 according to the following table.

Server Number	Database
11	nyc::hq
12	lon::europe
14	sfo::west

## Server Assignments for CDDS 2

The Company must assign the Replicator Server numbers for CDDS 2 according to the following table.

Server Number	Database
22	lon::europe
23	hkg::asia

# Chapter 4: Configuring Replication Using Visual DBA

---

This section contains the following topics:

[How You Configure Replicator Using VDBA](#) (see page 84)

[Installation of Replication Objects](#) (see page 85)

[Replication Branch in DOM Window](#) (see page 85)

[How You Work with Replication Databases](#) (see page 86)

[How You Create a CDDS](#) (see page 87)

[CDDS Definition Dialog](#) (see page 88)

[Creation of Replication Keys](#) (see page 91)

[How Configuration Information Is Propagated](#) (see page 92)

[How Replication Is Activated](#) (see page 93)

[How You Deactivate Replication](#) (see page 93)

[Error Mail Destinations](#) (see page 93)

In this chapter, you will need to refer to the completed worksheets from the chapter “Planning Your Replication Scheme.”

## How You Configure Replicator Using VDBA

To configure Ingres Replicator, you should perform each of the following configuration procedures in the order shown. Perform these procedures on test tables first, before you register production tables for replication.

1. Install replication objects on full peer and protected read-only targets.  
For details, see [Installing Replicator Objects in VDBA online help](#).
2. Define the local database.  
For details, see [Installing Replicator Objects in VDBA online help](#).
3. Define other replicated databases.  
For details, see [Adding Replication Databases in VDBA online help](#).
4. Create CDDS definitions.  
For details, see [Creating a CDDS in VDBA online help](#).
5. Enter your propagation paths.  
For details, see [Creating a CDDS in VDBA online help](#).
6. Select and register tables.  
For details, see [Creating a CDDS in VDBA online help](#).
7. Create support objects for your registered tables.  
For details, see [Creating a CDDS in VDBA online help](#).
8. Create replicated transaction keys, if necessary.  
For details, see [Creating Replication Keys in VDBA online help](#).
9. Move your configuration to other full peer and protected read-only targets.  
For details, see [Propagating Configuration Information in VDBA online help](#).
10. Activate change recording for any or all CDDSs and databases.  
For details, see [Creating a CDDS or Activating Replication in VDBA online help](#).
11. (Optional) Set up an e-mail list on every installation for message notification.  
For details, see [Defining Error Mail Destinations in VDBA online help](#).
12. Configure the servers to the settings best suited to your replication system.  
For more information, see the chapter "Using Replicator Server."

## Installation of Replication Objects

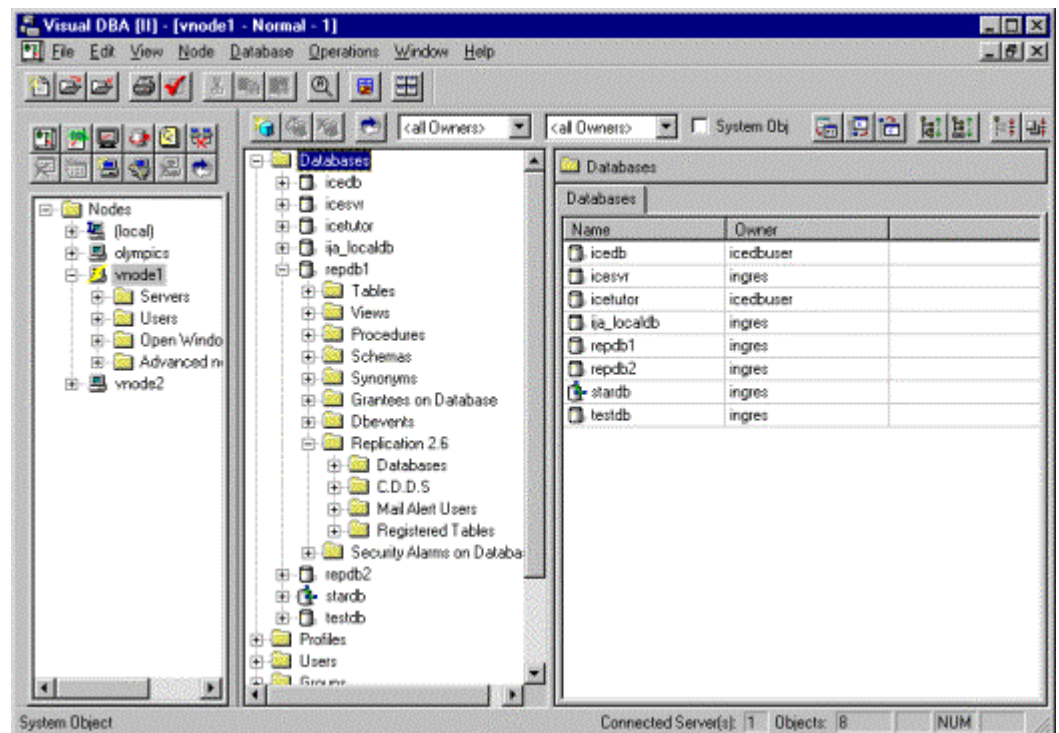
You must install replication objects on the database where you are defining the replication scheme. You do not need to install replication objects on databases with only Unprotected Read-only CDDs. Replication objects must be installed before you can move the Ingres Replicator configuration from a configuration database to the other participating databases.

When you move configuration information (in the replication scheme definitions step), you are prompted automatically to install replication objects on the appropriate databases.

You install replication objects by choosing the Install Replication Objects command from the Operations menu. For more information, see the online help topic Installing Replicator Objects.

## Replication Branch in DOM Window

After installing replication objects, sub-branches appear under the Replication object category branch in the Database Object Manager (DOM), as shown in the following example. (Your release of Ingres Replicator appears on the Replication object category branch when it is expanded.)



## How You Work with Replication Databases

When you expand the Databases branch, all databases that Ingres Replicator recognizes are displayed. The local database information that you entered in the Installation Local DB Information dialog appears in the expanded branch.

You must add all other databases that belong in your replication scheme to the list. Your completed Database Worksheet must contain all the information you need to enter.

In the Replication branch of the Database Object Manager window, you can use the Databases branch to:

- Create and alter replication databases
- View existing database objects, including the detailed properties of each individual object
- Drop database objects

Detailed steps for performing these procedures can be found in the following Visual DBA online help topics:

- Adding Replication Databases
- Viewing Object Properties
- Altering a Database
- Dropping a Database

## How You Create a CDDS

You create a CDDS by performing the following procedures on the CDDS Definition dialog:

1. Specify collision and error modes.

You can specify collision and error modes for each CDDS. For details, see the online help topic CDDS Definition dialog.

2. Add or drop propagation paths for a CDDS.

For more information, see the online help topic Choosing Propagation Paths. See also Replication Scheme Examples (see page 78).

3. Specify the databases in the CDDS and assign their target types and server numbers.

4. Register tables for replication; create Ingres Replicator support objects; turn on the Change Recorder; and specify lookup and priority lookup tables associated with a base table.

5. Choose columns in each table that are to be replicated.

The information for defining your CDDSs is contained on the CDDS Worksheet, an example of which is in Replication Scheme Examples (see page 78).

## CDDS Definition Dialog

The CDDS Definition dialog lets you define a CDDS.

This dialog has the following items:

### Collision Mode

Specifies the collision mode for the CDDS.

### Error Mode

Specifies the error mode for the CDDS.

### Propagation Path

Specifies the propagation path for the CDDS.

### Database Information for CDDS

Lists databases that you have specified for your propagation paths, as shown in this example:

**CDDS Definition on vnode1::repdb1**

Cdds No: 0 Name: Default CDDS Collision Mode: Passive Detector Error Mode: Skip Transaction

**Propagation Path**

Originator	Local	Target
vnode1::repdb1	vnode1::repdb1	vnode2::repdb2
vnode2::repdb2	vnode2::repdb2	vnode1::repdb1

**Database Information for CDDS**

DB No	Vnode/Database Name	Type	Server
10	vnode1::repdb1	Full Peer	1
20	vnode2::repdb2	Full Peer	2

**Tables**

All None ☐ Support Objects ☐ Table Activated

Lookup Table:  Priority Lookup Table:

**Columns**

Name	Rep.Key	Rep.
col1	1	✓
col2	0	✓

You can change, add, or drop the parameters for a particular database by clicking the appropriate button in this group box.

For more information, see the online help topic Target Type and Server Dialog.



## Tables

Displays all tables in the local database that are available to register for replication. For each table that you want to be replicated from your source database to the target databases, enable the check box. Additionally, you can specify the table parameters:

### Support Objects Check Box

Creates Ingres Replicator support objects for the selected table, including Ingres Replicator procedures and the shadow and archive tables.

**Note:** For support objects to be created the relationship between the CDDS assigned to the table and local database must be defined. In addition, to correctly configure a horizontally partitioned table, the CDDS lookup table must exist and be populated with the required rows.

### Table Activated Check Box

Activates the Change Recorder for the particular table. Once activated, any changes to the table are collected by Ingres Replicator and sent to the input queue for distribution.

You can activate or deactivate individual tables. Activating a table enables change recording, and deactivating a table disables change recording.

**Note:** Normally, it is not necessary or advisable to activate or deactivate individual tables. For more information, see the online help topics *Activating Replication* and *Deactivating Replication*.

### Lookup Table

Specifies the CDDS lookup table, if any, associated with a base table. To use horizontal partitioning, you must use lookup tables that specify your parameters.

You must create the lookup table before you can assign it to a CDDS. You must create the lookup table individually for each database that needs it. For instructions on creating lookup tables, see *Lookup Tables* (see page 201).

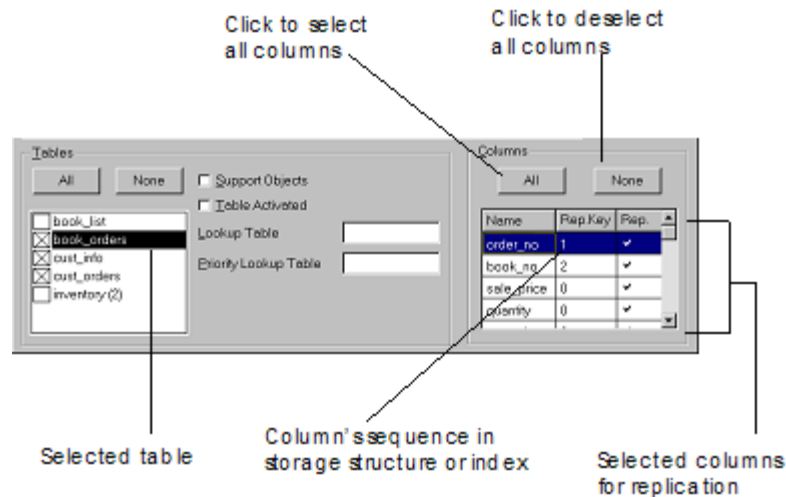
### Priority Lookup Table

Specifies the priority lookup table, if any, associated with a base table.

## Columns

Specifies the columns to be registered for replication for the selected table.

The Columns container is filled with the columns defined for the table selected in the list box, as shown in the following window fragment:



For more information, see the online help topic Choosing Columns for Replication.

## View Table Registration Information

You can view table registration information for the tables assigned to a particular CDDS.

### To view table registration information

See the Tables group box in the CDDS Definition dialog.

For more information, see the online help topic Registered Tables page, Replicator CDDS branch.

## Creation of Replication Keys

The Create Keys option creates the replicated transaction keys for every row in the base table and populates the shadow table and the input queue. You must use Create Keys if you install Ingres Replicator on an existing database that contains data. If you install Ingres Replicator on empty databases, replicated transaction keys are created and the shadow table is populated automatically when you insert data. For more information, see the online help topic [Creating Replication Keys](#).

How you use Create Keys varies depending on the contents of your databases.

There are two options when using Create Keys:

### **Shadow Table Only**

This option populates the shadow table.

### **Both Queue and Shadow Table**

This option populates the shadow table and the input queue. Use this option if your databases are not synchronized; the altered rows are placed in the input queue for reconciliation or distribution.

You create replicated transaction keys for tables by choosing Create Keys from the Operations menu. For more information, see the online help topic [Creating Replication Keys](#). For information about using Create Keys options, see the online help topic [Creating Replicated Transaction Keys dialog](#).

## How Configuration Information Is Propagated

Once you have completed the configuration tasks for the replicated database, you can move the configuration information to your target databases (Full Peer and Protected Read-only databases). The propagation procedure ensures that the replication scheme is defined identically in each database of the scheme.

You can propagate configurations by choosing the Propagate command from the Operations menu. For more information, see the online help topic *Propagating Configuration Information*.

**Note:** If you make any changes to the configuration once you have Ingres Replicator set up and running, you must copy the changes to the target databases. In general, you must not change the collision or error mode for a CDDS on just one of the databases that participate in a CDDS.

The propagating configuration procedure completes the following tasks on the databases it is run against:

- Copies the configuration data to the target Replicator catalogs
- Makes local context changes to the target data
- Creates support objects; if support objects already exist in the target database, these older objects are first destroyed

The propagating configuration procedure does *not* perform any of the following tasks on remote databases:

- Create lookup tables

You must create or copy lookup tables in each database. The propagating configuration procedure copies lookup table assignments. For more information, see *Lookup Tables* (see page 201).

- Activate the remote databases

You must activate each database by choosing the Activate Replication command from the Operations menu (alternatively, use the Table Activated option in the CDDS Definition dialog) in order for Ingres Replicator to run. (See the online help topics *CDDS Definition dialog* and *Activating Replication* for details).

- Define error mail destinations

If you want to create individual error mail destination lists, you must define error mail destinations on every installation by expanding the Replication branch and selecting the Mail Alert Users branch. For more information, see the online help topic *Defining Error Mail Destinations*.

## How Replication Is Activated

You can activate replication (change recording) for all full peer databases of the CDDS, or you can activate replication at the table level, by choosing the Activate Replication command from the Operations menu. For more information, see the online help topic [Activating Replication](#).

When you activate a table, the DBMS enables change recording for that table. Once activated, any changes to a table are collected by the DBMS and placed on the input queue for distribution.

## How You Deactivate Replication

You can deactivate change recording for a selected CDDS by choosing the Deactivate Replication command from the Operations menu. For more information, see the online help topic [Deactivating Replication](#).

## Error Mail Destinations

The Replicator servers can be configured to send mail on certain error conditions. You can define the list of users who receives this mail.

To create individual error mail destination lists, you must define error mail destinations on every installation by expanding the Replication branch and selecting the Mail Alert Users branch. For more information, see the online help topic [Defining Error Mail Destinations](#).



# Chapter 5: Configuring Replication Using Replicator Manager

---

This section contains the following topics:

[Replicator Manager](#) (see page 95)  
[How You Configure Replicator Using Replicator Manager](#) (see page 101)  
[repcat Command—Create Replicator Catalogs](#) (see page 102)  
[Run the Replicator Manager the First Time](#) (see page 103)  
[Configuration Menu](#) (see page 104)  
[Editing Operations](#) (see page 108)  
[Database Summary](#) (see page 108)  
[CDDS Summary](#) (see page 110)  
[Table Registration Summary](#) (see page 117)  
[Move Configuration Data Window](#) (see page 126)  
[Activate Change Recording Window](#) (see page 128)  
[Mail Notification List Window](#) (see page 129)

## Replicator Manager

The Replicator Manager is a forms-based utility for configuring Replicator to conform to your replication scheme.

This utility allows the Distributed DBA for Replicator to complete the following tasks:

- Monitor the distributed replication in a local database
- Connect to a single database for table registration, a task that is required to set up a table or database for replication
- Configure the replication scheme
- Copy replication configuration information to other databases
- Check for replication configuration errors
- Issue database events to determine server action
- Monitor the replications into and out of a database
- Run Replicator integrity reports

## repmgr Command—Start the Replicator Manager

The repmgr command invokes Replicator Manager.

**Note:** To run Replicator Manager, you must have the correct system privileges and use the correct DBA name. An Ingres user with security privilege can impersonate the DBA and modify Replicator Manager information.

The repmgr command has the following format:

```
repmgr [-udba_name] [vnode:]dbname
```

**-udba\_name**

(Optional) Specifies the effective user for the session. You must run this command as the owner of the database.

**[vnode:]dbname**

Specifies the database to connect to. By specifying the node name of a remote database to Replicator Manager, the database administrator can administer the Ingres Replicator network from the local machine. However, replication servers cannot be started from a different node, and the RepServer configuration file cannot be edited from a remote node.

For example, the following command, executed from a remote San Francisco computer, allows Replicator Manager to be run in client-server mode from the local San Francisco computer to the hq database on the nyc node and assumes that the San Francisco DBA is impersonating the New York DBA (nyc\_dba).

```
repmgr -unyc_dba nyc::hq
```



## Navigation and Operation of Replicator Manager

You operate Replicator Manager in exactly the same way as all other forms-based utilities. A brief summary of standard and window editing operations is provided in this section. You can also refer to *Character-Based Querying and Reporting Tools User Guide* for general information on operating each forms-based product.

Replicator Manager uses pop-up windows to display prompts and messages and to list available choices on some menu operations.

Replicator Manager menu operations are displayed at the bottom of each window. For example:

```
Save  Clear  ListChoices  Help  >
```

The key on your keyboard that facilitates a particular operation depends on your terminal and the individual key mappings you have chosen.

To move the cursor to the menu line, press the Menu key. To leave the menu line, press Enter. The symbols > or < indicate the presence of additional menu items that are not visible on the menu line. To view additional menu items, press the Menu key repeatedly to cycle through all options.

### Execute Menu Items

#### To execute any of the menu items on a Replicator Manager window

1. Highlight the required menu option or applicable row on the window.
2. Do one of the following:
  - Press the key that is designated for the option.
  - Press the Menu key to move the cursor to the menu line, type the full name of the menu item or enough letters to uniquely identify it, and then press Enter.

For example, if the menu items Create and Configure are displayed on the menu line, you must type at least **cr** to identify create and **co** to identify configure.

## Standard Operations

The following lists operations that are common to most windows in Replicator Manager. Menu operations required for Ingres Replicator configuration are described wherever applicable in the remainder of this chapter.

### **Save**

Saves data on the window

### **Select**

Calls the function that is currently highlighted on the menu

### **Go**

Executes an operation

### **ListChoices**

Provides data entry requirements or choices for the currently highlighted area on the window

### **Cancel**

Cancels the current operation and restores the previous menu line

### **Help**

Displays the help text for the currently displayed window

### **End**

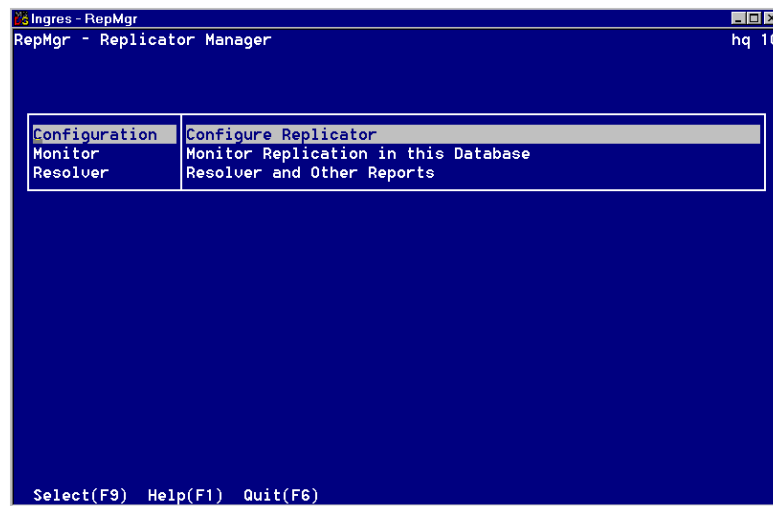
Returns to the previous window

### **Quit**

Exits Replicator Manager

## Replicator Manager Menu Window

When you start Replicator Manager, the main menu is displayed, as shown in the following example:



The main menu has the following items:

### Database name and number

Is displayed in the upper right corner. Shown are the database name, the vnode name (if entered on the command line), and the database number accessed with Replicator Manager.

### Configuration

Lets you configure Replicator. For more information, see Configuration Menu (see page 104).

### Monitor

Lets you monitor Replicator. For more information, see Replication Monitor Window (see page 158).

### Resolver

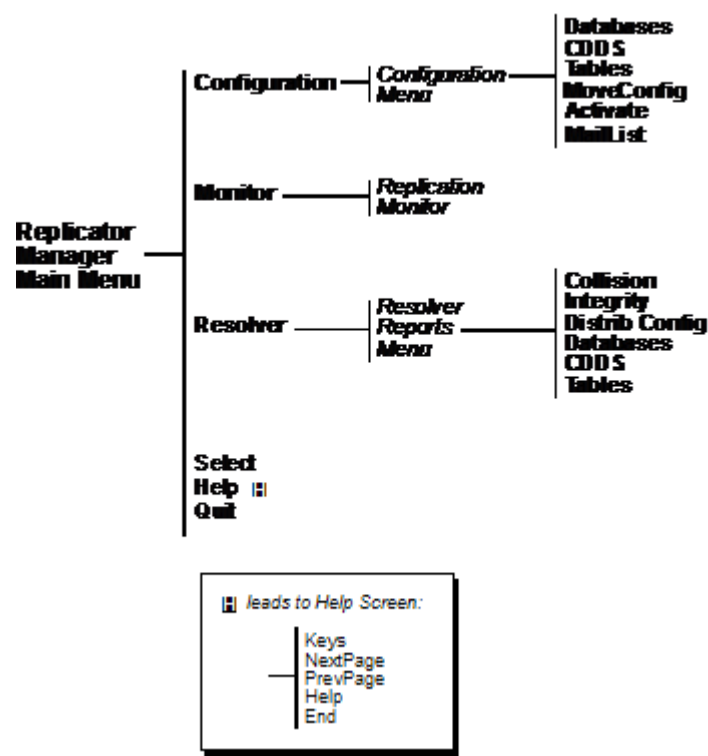
Lets you resolve problems. For more information, see Resolver Reports Menu (Replicator Manager) (see page 186).

Replicator Manager Main Menu Map

The following map shows each selection that is accessible from the Replicator Manager main menu.

The menu map uses the following conventions:

- Bold typeface—Indicates items selected from boxed list window areas
- Regular typeface—Indicates items selected from menu list at the bottom of the window
- Italics typeface—Indicates window names



## How You Configure Replicator Using Replicator Manager

To configure Ingres Replicator, you should perform each of the following configuration procedures in the order shown. Perform these procedures on test tables first, before you register production tables for replication.

1. Create Replicator Catalogs on full peer and protected read-only targets.  
For details, see [repcat Command](#) (see page 102).
2. Start Replicator Manager from a database that has a full peer CDDS  
For details, see [Run the Replicator Manager the First Time](#) (see page 103).
3. Define the local database.  
For details, see [Run the Replicator Manager the First Time](#) (see page 103).
4. Define other replicated databases.  
For details, see [Add a Database to the Database Summary List](#) (see page 109).
5. Create your CDDS definitions.  
For details, see [Add a CDDS](#) (see page 111).
6. Specify CDDS databases.  
For details, see [CDDS Database and Servers Window](#) (see page 112).
7. Assign server numbers.  
For details, see [CDDS Database and Servers Window](#) (see page 112).
8. Enter your propagation paths.  
For details, see [Propagation Path Definition Window](#) (see page 114).
9. Register tables.  
For details, see [Register Tables](#) (see page 118).
10. Create support objects for your registered tables.  
For details, see [Create Support Objects](#) (see page 122).
11. Create replicated transaction keys, if necessary.  
For details, see [Create Keys Options](#) (see page 124).
12. Move your configuration to other full peer and protected read-only targets.  
For details, see [Move Configuration Data Window](#) (see page 126).
13. Activate change recording for any or all CDDSs and databases.  
For details, see [Activate Change Recording Window](#) (see page 128).
14. (Optional) Set up an e-mail list on every installation for message notification.  
For details, see [Create a Mail Notification List](#) (see page 130).

15. Configure the servers to the settings best suited to your replication system.

For more information, see the chapter "Using Replicator Server."

## repcat Command—Create Replicator Catalogs

The repcat utility creates and populates the Ingres Replicator catalogs and creates Ingres Replicator database events.

You must run repcat before starting the Replicator Manager for the first time. The repcat utility must be run on every Ingres Replicator database containing Full Peer or Protected Read-only CDDs. You do not need to run repcat on databases with only Unprotected Read-only CDDs; repcat must be run before you can move the Ingres Replicator configuration from a configuration database to the other participating databases.

The repcat command has the following format:

```
repcat [-udba_name] [vnode::] dbname
```

**-udba\_name**

(Optional) Specifies the effective user for the session.

**[vnode::]dbname**

Is the name of the database optionally preceded by the virtual node name (vnode).

After repcat is invoked, the following message appears:

```
Creating Replicator catalogs on database 'dbname' ...  
Replicator catalogs for database 'dbname' created successfully.
```

## Run the Replicator Manager the First Time

You must start the Replicator Manager from a database that has at least one full peer CDDS. You must configure Ingres Replicator on such a database and move the configuration to all databases that participate in the replication scheme.

**Note:** Before you run the Replicator Manager for the first time, you must create the Ingres Replicator catalogs utility. For more information, see `repcat Command` (see page 102).

### To start the Replicator Manager and define the local database

1. Issue the `repmgr` command. For details, see `repmgr Command` (see page 96).

The Define Local Database window appears:

```

Ingres - RepMgr
RepMgr - Define Local Database

Database Number: 10___
Virtual Node Name: nyc_____
Database Name: hq
Database Owner: nyc_dba
DBMS Type: ingres
Remarks: Headquarters_____

Save(F3) Clear(SH-F2) Help(F1) End(F10)

```

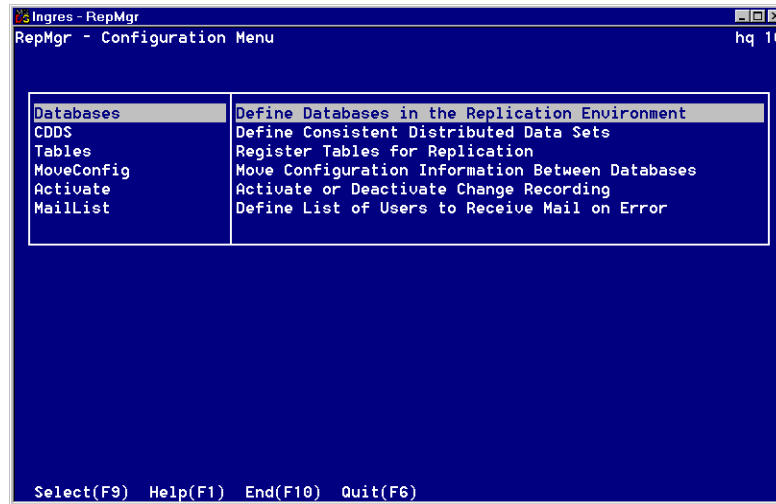
**Note:** The Define Local Database window is displayed only the first time Replicator Manager is run against a database.

2. Enter appropriate values from your Database Worksheet in the Database Number, Virtual Node Name, and Remarks fields (use the tab key to move from one field to the next), and choose Save.

The local database details are saved and the Replicator Manager main menu is displayed with the name and number of the database in the upper-right corner.

## Configuration Menu

The Configuration Menu window (shown here) appears when you choose Configuration option on the main menu. It allows you to perform various configuration tasks.





The Configuration menu has the following items:

**Databases**

Lets you define the replicated databases that send or receive replicated data.

For details, see Database Summary (see page 108).

**CDDS**

Lets you define Consistent Distributed Data Sets, their locations, and data propagation paths.

For details, see CDDS Summary (see page 110).

**Tables**

Lets you register tables to be replicated, and create tables and database procedures to support the replication of those tables.

For details, see Table Registration Summary (see page 117).

**MoveConfig**

Lets you move the configuration information to the other databases participating in the replication scheme.

For details, see Move Configuration Data Window (see page 126).

**Activate**

Lets you activate change recording for selected CDDSs and databases.

For details, see Activate Change Recording Window (see page 128).

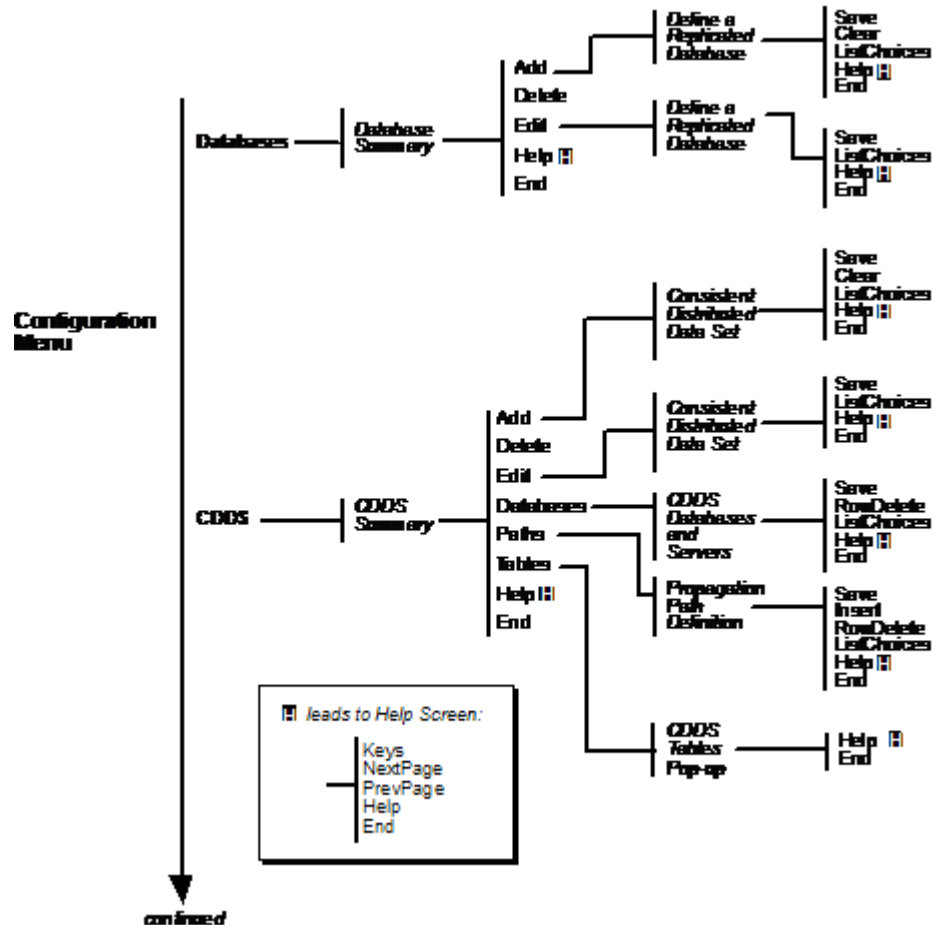
**MailList**

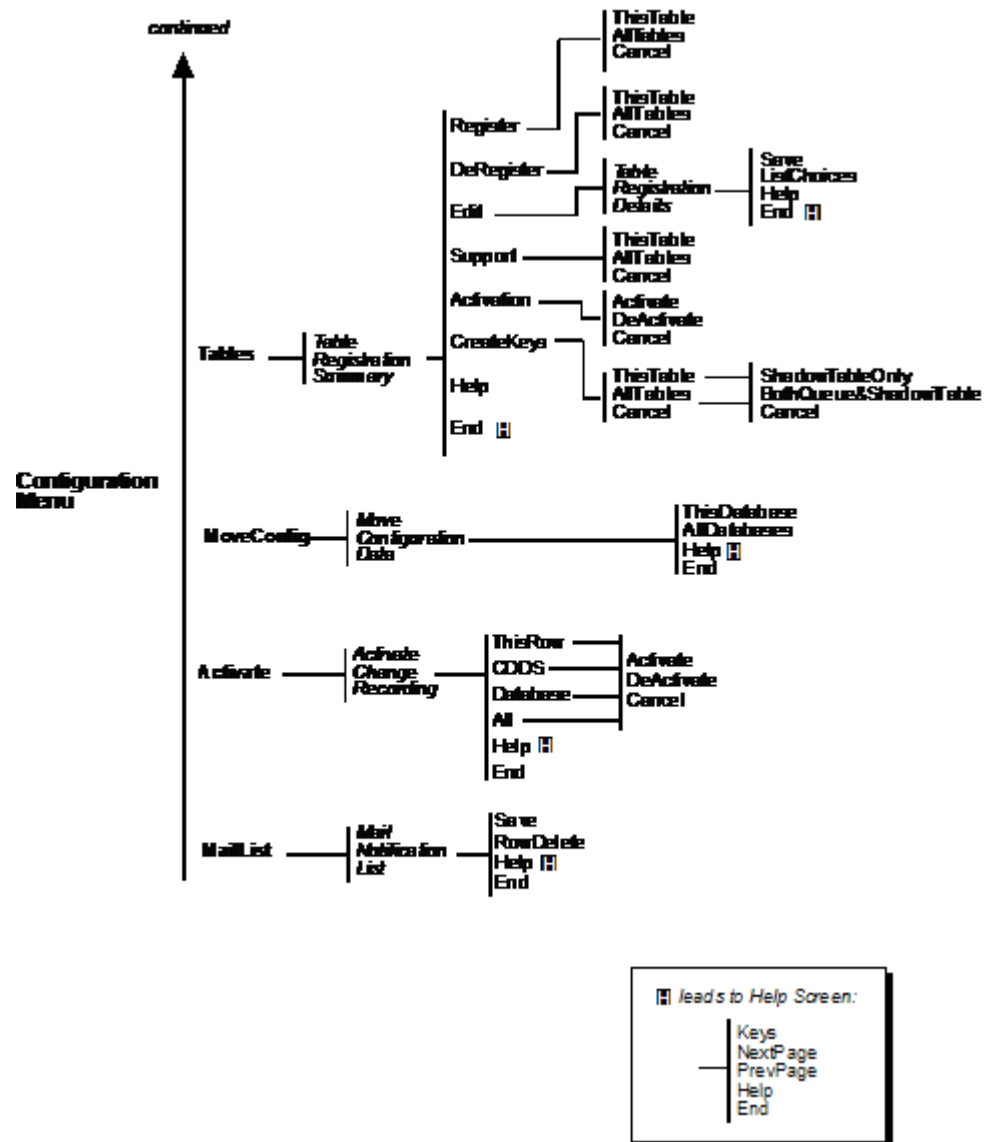
Lets you specify email addresses of the users who receive Ingres Replicator error messages through email.

For details, see Mail Notification List Window (see page 129).

## Configuration Menu Map

The various menu options available with the Configuration Menu are shown in the following map:





## Editing Operations

The following is a list of the menu line operations that provide ways to edit window data in Replicator Manager:

### Clear

Clears every field on the window

### RowDelete

Deletes the highlighted row on the window

### Insert

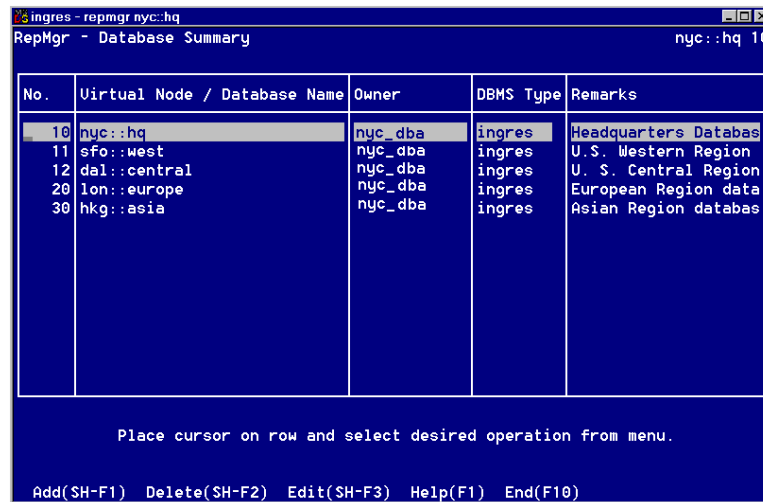
Inserts a row on the window above the highlighted row

## Database Summary

The Database Summary window (shown here) appears when you choose the Databases option from the Configuration Menu.

The Database Summary windows lists the databases that Ingres Replicator recognizes. By default, the local database information that you entered in the Define Local Database window appears in the list.

You must add all other databases that belong in your replication scheme to the list. Your completed Database Worksheet must contain all the information you need to enter.



No.	Virtual Node / Database Name	Owner	DBMS Type	Remarks
10	nyc::hq	nyc_dba	ingres	Headquarters Databas
11	sfo::west	nyc_dba	ingres	U.S. Western Region
12	dal::central	nyc_dba	ingres	U. S. Central Region
20	lon::europe	nyc_dba	ingres	European Region data
30	hkg::asia	nyc_dba	ingres	Asian Region databas

Place cursor on row and select desired operation from menu.

Add(SH-F1) Delete(SH-F2) Edit(SH-F3) Help(F1) End(F10)

In this window, you can:

- Add a database to the list
- Edit an existing database's information
- Delete a database from the list

For descriptions of the fields in this window, see online help.

## Add a Database to the Database Summary List

You must add each database in your replication scheme to the Database Summary list.

### To add a database to the Database Summary list

1. Open the Database Summary window and choose Add.

The Define a Replicated Database window appears. The fields in this window are similar to the fields in the Database Summary window.

2. Fill in each field and choose Save.

The database information is saved and you are returned to the Database Summary window.

## Edit a Database

### To edit a database's information

1. Open the Database Summary window and place the cursor on the database to be edited.

2. Choose Edit.

The Define a Replicated Database window appears.

3. Edit the fields to be changed (the Database Number field cannot be edited), and choose Save.

The database information is saved and you are returned to the Database Summary window.

## Delete a Database

### To delete a database from the Database Summary list

1. In the Database Summary window, place the cursor on the database to be deleted.
2. Choose Delete.

A pop-up window prompts you to verify that you really want to delete this database.

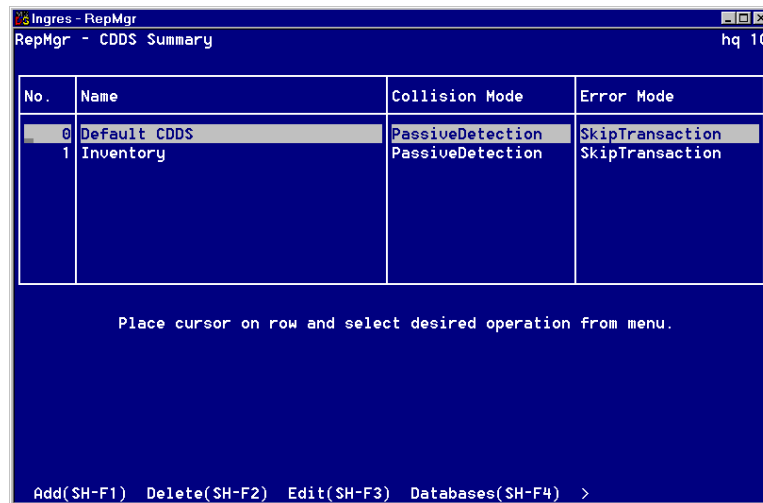
3. Choose Yes and press Select.

The selected database's information is deleted.

**Note:** Deleting a database from this window also deletes it from any CDDSs where it has been defined and from any propagation paths containing it.

## CDDS Summary

The CDDS Summary window (shown here) appears when you choose the CDDS option from the Configuration Menu. It contains a list of all defined CDDSs.



No.	Name	Collision Mode	Error Mode
0	Default CDDS	PassiveDetection	SkipTransaction
1	Inventory	PassiveDetection	SkipTransaction

Place cursor on row and select desired operation from menu.

Add(SH-F1) Delete(SH-F2) Edit(SH-F3) Databases(SH-F4) >

The information for defining your CDDSs is contained in the CDDS Worksheet.

In the CDDS Summary window you can:

- Add or edit (define) a CDDS
- Delete a CDDS
- Specify the databases in the CDDS and assign their target types and server numbers
- Define propagation paths for a CDDS
- View tables registered under a CDDS

For a description of the fields in the CDDS Summary window, see online help.

## Add a CDDS

### To add a CDDS to the CDDS Summary list

1. Open the CDDS Summary window and choose Add.

The Consistent Distributed Data Set window appears. The fields in this window are the same as the fields in the CDDS Summary window.

2. Fill in each field and choose Save.

The CDDS information is saved and you are returned to the CDDS Summary window.

## Edit a CDDS

### To edit a CDDS in the CDDS Summary list

1. Open the CDDS Summary window, place the cursor on the CDDS to be edited, and choose Edit.

The Consistent Distributed Data Set window appears.

2. Fill in each field (the CDDS Number field is not editable) and choose Save.

The CDDS information is saved and you are returned to the CDDS Summary window.

## Delete a CDDS

### To delete a CDDS in the CDDS Summary window

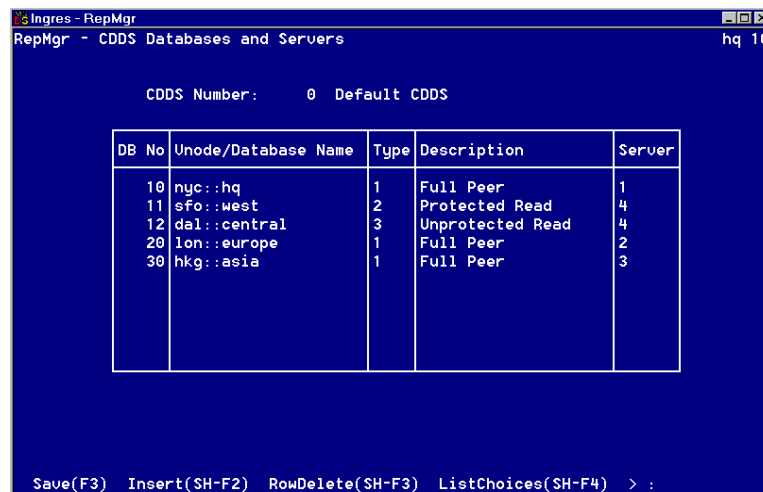
Place the cursor on the CDDS to be deleted and choose Delete.

**Note:** Deleting a CDDS also deletes all its database locations and its propagation paths and reassigns any tables registered under that CDDS to CDDS 0.

## CDDS Database and Servers Window

The CDDS Databases and Servers window (shown here) appears when you choose the Databases option in the CDDS Summary window.

You must specify in which databases your CDDS is located. You must also specify the CDDS target type and assign a Replicator Server number to databases included in your CDDS.



For a description of the fields in the CDDS Databases and Servers window, see online help.

From this window you can:

- Add new database locations, server assignments, and target types to the CDDS
- Edit database locations, server assignments, and target types in the CDDS
- Delete database locations, including the associated server assignments and target types from the CDDS



## Add Database and Server Information

### To add database and server information to a CDDS

1. Open the CDDS Summary window, place the cursor on the CDDS for which you want to add database information, and choose Databases.

The CDDS Databases and Servers window appears.

2. Fill in each field for all the databases in your CDDS (the Vnode/Database Name and Description fields are automatically filled in), and choose Save.

The changes are saved and you are returned to the CDDS Summary window.

## Edit Database and Server Information

### To edit database information

1. Open the CDDS Databases and Servers window and place the cursor on the row to be edited.
2. Move to the field you want to change, type over the existing information, and choose Save.

The changes are saved and you are returned to the CDDS Summary window.

## Delete Database and Server Information

### To delete a particular row in the CDDS Databases and Servers window

1. Place the cursor on the row and choose RowDelete.
2. Choose Save.

The row is deleted.

## Propagation Path Definition Window

The Propagation Path Definition window (shown here) appears when you choose the Paths option in the CDDS Summary window.

Orig#	Originator Database	Loc#	Local Database	Targ#	Target Database
10	nyc::hq	10	nyc::hq	11	sfo::west
10	nyc::hq	10	nyc::hq	12	dal::central
10	nyc::hq	10	nyc::hq	20	lon::europe
10	nyc::hq	20	lon::europe	30	hkg::asia
10	nyc::hq	30	hkg::asia	11	sfo::west
20	lon::europe	10	nyc::hq	11	sfo::west
20	lon::europe	10	nyc::hq	12	dal::central
20	lon::europe	20	lon::europe	10	nyc::hq
20	lon::europe	20	lon::europe	30	hkg::asia
20	lon::europe	30	hkg::asia	11	sfo::west

Save(F3) Insert(SH-F2) RowDelete(SH-F3) ListChoices(SH-F4) >

For a description of the fields in the Propagation Path Definition window, see online help.

From this window, you can:

- Add propagation paths
- Edit propagation paths
- Delete propagation paths

## Add Propagation Paths

Before you can add propagation paths for a CDDS, you must have specified the database locations for the CDDS. For instructions, see CDDS Database and Servers Window (see page 112).

### To add propagation paths to a CDDS

1. Open the CDDS Summary window, place the cursor on the CDDS to which you want to add paths, and then choose Paths.

The Propagation Path Definition window appears.

2. Choose Insert. (If no propagation paths are defined for the CDDS, you are automatically in insert mode.)
3. Fill in the Orig#, Loc#, and Targ# fields, using the Tab key to move from field to field.

The database name fields are filled in automatically.

4. Repeat Steps 3 and 4 for all paths, and then choose Save.

Your changes are saved and you are returned to the CDDS Summary window.

## Edit Propagation Paths

### To edit propagation paths

1. Open the Propagation Path Definition window and place the cursor on the row to be edited.
2. Move to the field you want to change, type over the existing information, and choose Save.

The changes are saved and you are returned to the CDDS Summary window.

## Delete Propagation Paths

### To delete a row in the Propagation Path Definition window

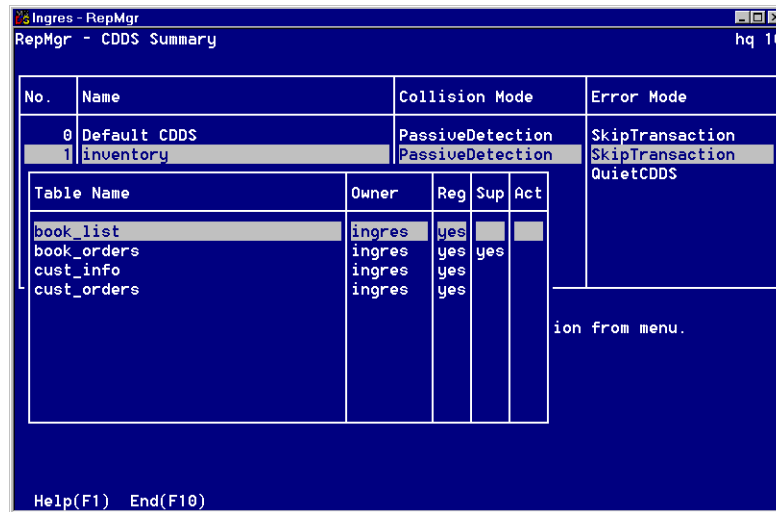
Move the cursor to the row to be deleted and choose RowDelete.

The row is deleted.

## View Table Registration Information

To view the table registration information for the tables assigned to a particular CDDS, place the cursor on the CDDS in the CDDS Summary window and choose Tables.

The CDDS Tables pop-up window appears, as shown in this example:



The screenshot shows a window titled "Ingres - RepMgr" with a subtitle "RepMgr - CDDS Summary" and a status bar "hq 10". The main content area displays a table with the following data:

No.	Name	Collision Mode	Error Mode
0	Default CDDS	PassiveDetection	SkipTransaction
1	inventory	PassiveDetection	SkipTransaction QuietCDDS

Table Name	Owner	Reg	Sup	Act
book_list	ingres	yes		
book_orders	ingres	yes	yes	
cust_info	ingres	yes		
cust_orders	ingres	yes		

ion from menu.

Help(F1) End(F10)

For a description of the fields in the Tables pop-up window, see online help.

For more information on table registration, see Table Registration Summary (see page 117).

## Table Registration Summary

The Table Registration Summary window (shown here) appears when you choose the Tables option from the Configuration menu. Once you have registered tables under a CDDS, this screen shows which tables are assigned to a particular CDDS. For more information, see Register Tables (see page 118).

This window contains a list of all tables in the database and shows their registration, support, and activation status. By default they are assigned to Default CDDS (0). Ingres Replicator can only detect changes from tables that are activated.

Table Name	Owner	CDDS Name	Reg	Sup	Act
book_list	ingres	inventory	yes		yes
book_orders	ingres	inventory	yes	yes	
cust_info	ingres	inventory	yes		
cust_orders	ingres	inventory	yes		
dd_arcclean_tx_id	ingres	Default CDDS			
dd_server_special	ingres	Default CDDS			
inventory	ingres	Inventory	yes		yes

Place cursor on row and select desired operation from menu.

Register(F9) Deregister(SH-F2) Edit(SH-F3) Support(SH-F4) >

For a description of the fields in this window, see online help.

You can perform the following tasks from the Table Registration Summary window:

- Register tables for replication
- Remove table registration from tables that do not require replication or require reconfiguration
- Edit table registration information, including deregistering specific columns, changing the CDDS assignment, and assigning lookup tables for horizontal partitioning and priority collision resolution
- Create internal Ingres Replicator tables and procedures that support replication of a database table
- Activate or deactivate a table by enabling or disabling change recording for a table
- Optionally create replicated transaction keys and populate shadow tables for tables that already contain data

## Register Tables

You must register all tables that you want to replicate. By default, tables are initially registered as belonging to CDDS 0. Once you have registered them, you can edit their registration information, including the CDDS number. For more information on editing a table's registration, see Table Registration Details Window (see page 119).

### To register a table or all tables

1. Open the Table Registration Summary window, place the cursor on the table to be registered, and choose Register.  
A new menu line appears.
2. Choose ThisTable if you want to register the highlighted table, or choose AllTables if you want to register all tables.  
A yes appears in the Reg column of the Table Registration Summary window next to the table or tables you registered.

**Note:** To be registered, a table must have a unique primary storage structure or a unique secondary index.

## Deregister Tables

When you deregister a table, all support objects are destroyed.

### To deregister a table

1. Open the Table Registration Summary window, move the cursor to the table to be deregistered, and choose Deregister.  
A new menu line appears.
2. Choose ThisTable if you want to deregister the highlighted table, or choose AllTables if you want to deregister all tables.  
Blanks appear in the Reg, Sup, and Act columns of the Table Registration Summary window next to the table or tables you deregistered.

## Table Registration Details Window

The Table Registration Details window appears (shown here) when you choose the Edit option from the Table Registration Summary window.

**Note:** If you attempt to edit a table that has not yet been registered, you are prompted to accept the default registration.

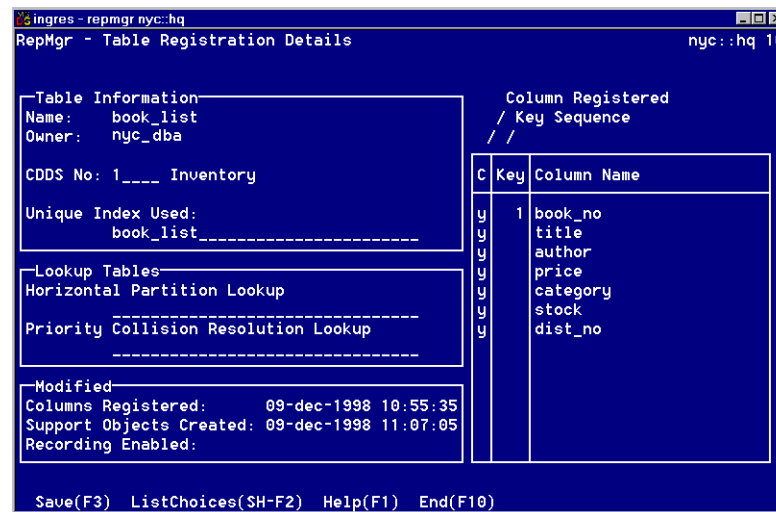


Table Information		Column Registered / Key Sequence	
Name:	book_list	C	Key
Owner:	nyc_dba		Column Name
CDDS No:	1	y	1 book_no
Unique Index Used:	book_list	y	title
		y	author
		y	price
		y	category
		y	stock
		y	dist_no

Save(F3) ListChoices(SH-F2) Help(F1) End(F10)

From this window you can perform the following tasks:

- Deregister specific columns (vertical partitioning)
- Change the CDDS number
- Specify a different unique index to be used as replicated table key
- Assign a horizontal partition lookup tableAssign a priority collision resolution lookup table

If you edit table registration information after creating support objects, when you select Save, the following message appears:

There are existing support objects for this table.

Saving this table registration requires that they be dropped. Do you want to do that now?

If you answer yes, the existing objects are dropped and you must recreate them using the Support menu item in the Table Registration Summary window.

For a description of the fields in this window, see online help.

## Deregister a Column From a Register Field

### To deregister a column from the Column Registered field

1. Open the Table Registration Details window, move the cursor to the column to be deregistered, and type **n**.

**Note:** Key columns and mandatory (NOT NULL NOT DEFAULT) columns cannot be deregistered.

Use the arrow keys to move to other columns to be deregistered.

Use the Tab key to move to other fields in the window,

2. Choose Save.

Your changes are saved and you are returned to the Table Registration Summary window.

## Assign a Table to a Different CDDS

You can move a table from the default CDDS to any other CDDS. We recommend that you assign tables to CDDSs before creating support objects.

### To assign a different CDDS to a table

1. Open the Table Registration Details window and move to the CDDS No (number) field.
2. Type the number of the CDDS to which the table is assigned, or choose ListChoices to display the list of defined CDDSs.
3. Edit other fields in the window if necessary, and then choose Save

Your changes are saved and you are returned to the Table Registration Summary window.



## Assign Lookup Tables

To use horizontal partitioning or priority collision resolution, you must use lookup tables that specify your parameters.

You must create the lookup table before you can assign it to a CDDS. You must create the lookup table individually for each database that needs it. For instructions on creating lookup tables, see [Lookup Tables](#) (see page 201).

### **To assign a lookup table to a CDDS**

1. Open the Table Registration Details window and move to either the Horizontal Partition Lookup field or the Priority Collision Resolution Lookup field.
2. Type the name of the lookup table in the appropriate field or choose ListChoices to see a list of tables.
3. Choose Save.

Your changes are saved and you are returned to the Table Registration Summary window.

## Create Support Objects

Before you can create Ingres Replicator support objects for a table, you must have registered it. Support objects include Replicator Server database procedures and the shadow and archive tables, all of which are created individually for each table.

**Note:** To create support objects, the relationship between the CDDS assigned to the table and local database must be defined. In addition, to correctly configure a horizontally partitioned table, the CDDS lookup table must exist and be populated with the required rows.

### To create support objects for a table or all tables

1. Open the Table Registration Summary window, place the cursor on the table where support objects are to be created, and choose Support.

A new menu line appears.

2. Choose ThisTable if you want to create support objects for the highlighted table, or choose AllTables if you want to create support objects for all registered tables.

A yes appears in the Sup column of the Table Registration Summary window next to the table or tables for which you created support objects.

If the table is assigned to a CDDS that is full peer in the local database, a shadow table, archive table, and remote replication procedures are created. If the table is assigned to a CDDS that is protected read-only in the local database, a shadow table and remote replication procedures are created. If the table is assigned to a CDDS that is unprotected read-only in the local database, no support objects are necessary.

## Activate or Deactivate a Table

You can activate or deactivate individual tables. Activating a table enables change recording, and deactivating a table disables the change recording. Before you can activate a table, you must have registered it and created its support objects.

**Note:** Typically, it is not necessary or advisable to activate or deactivate individual tables. For instructions on activating or deactivating CDDs, databases, or the entire replication scheme, see Activate Change Recording Window (see page 128).

### To activate or deactivate a table

1. Place the cursor on the table to be activated or deactivated, and choose Activation.

The menu line changes to:

Activate   DeActivate   Cancel

2. Choose Activate or DeActivate.

The selected table is activated or deactivated accordingly. A yes appears in the Act column if the table has been activated.

## CreateKeys Options

The CreateKeys option creates replicated transaction keys for every row in the base table and populates the shadow table. It can optionally populate the input queue. You must use CreateKeys if you install Ingres Replicator on an existing database that contains data. If you install Ingres Replicator on empty databases, replicated transaction keys are created and the shadow table is populated automatically when you insert data.

How you use CreateKeys varies depending on the contents of your databases. There are two options when using CreateKeys:

### ShadowTableOnly

This option populates the local shadow table. You need to copy the populated shadow table to all Full Peer or Protected Read-only targets that are part of the table's CDDS.

**Note:** Once you use ShadowTableOnly on a database, you need to use the copydb utility to copy the resulting shadow table to other databases in the CDDS. In other words, you cannot use ShadowTableOnly against the same table in two different databases, because the resulting replicated transaction keys do not match.

### BothQueue&ShadowTable

This option populates the shadow table and the input queue. Use this option if your databases are not synchronized; the altered rows are placed in the input queue for reconciliation or distribution. This option also needs to be used if the table is horizontally partitioned.

The following table explains when to use which CreateKeys option:

Databases	How To Use CreateKeys
Both are empty	Do not need to use CreateKeys. Because there are no rows of data, no replicated transaction keys can be generated.
One contains data, one is empty	Only use BothQueue&ShadowTable option of CreateKeys if your database is very small. The recommended procedure is to unload the tables with data and reload them into the empty database.
Both contain data and are synchronized	Use the ShadowTableOnly option of CreateKeys. This option populates the shadow tables so that Ingres Replicator does not detect collisions when manipulating rows for the first time.

Databases	How To Use CreateKeys
Both contain data but are not synchronized	If a table to be replicated is rather large, attempt to synchronize the databases through some other means or use the BothQueue&ShadowTable option of CreateKeys. Ingres Replicator populates the shadow tables and sends the rows from the local database to the target database. Nonmatching rows cause collisions, which need to be resolved manually or by using collision resolution. For more information on collision handling, see Collision Design (see page 52).

**Note:** If the BothQueue&ShadowTable option of CreateKeys is used, the rows placed in the input queue are not immediately transferred to the distribution queue. To effect the transfer, ensure that all users disconnect from the database. The transfer takes place when you re-connect.

## Create Replicated Transaction Keys

### To create the replicated transaction keys for the table

1. Open the Table Registration Summary window, place the cursor on the table where keys are to be created, and choose CreateKeys.

The following menu line appears:

ThisTable    AllTables    End

2. Choose ThisTable to create replicated transaction keys for the selected table, or choose AllTables to create keys for all registered tables that have support objects.

The following menu line appears:

ShadowTableOnly    BothQueue&ShadowTable    Cancel

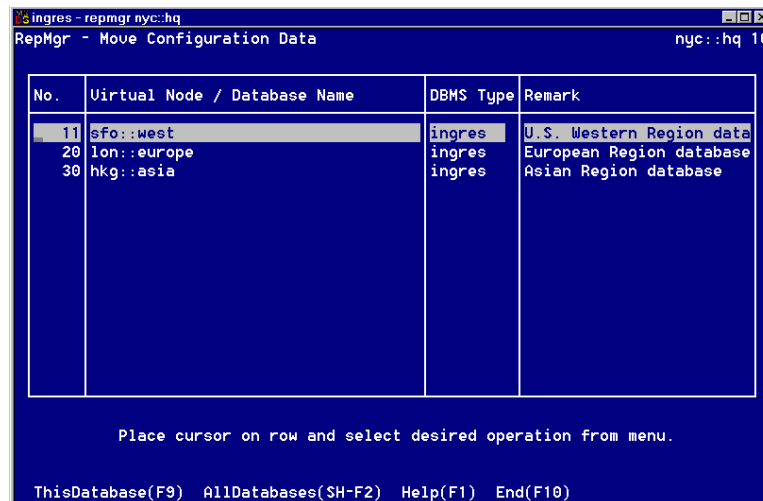
3. Choose ShadowTableOnly or BothQueue&ShadowTable.

The replicated transaction keys are created and placed in the shadow table. The input queue is populated, if appropriate.

## Move Configuration Data Window

The Move Configuration Data window (shown here) appears when you choose the MoveConfig option from the Configuration Menu.

This window allows you to copy your Ingres Replicator configuration from the initial database to other databases in the replication scheme. Only databases with full peer and protected read-only targets appear in this window. Unprotected read-only targets do not have an Ingres Replicator installation, and therefore do not appear on this window. Of course, the local database you are running the Replicator Manager against does not appear on this list.



No.	Virtual Node / Database Name	DBMS Type	Remark
11	sfo::west	ingres	U.S. Western Region data
20	lon::europe	ingres	European Region database
30	hkg::asia	ingres	Asian Region database

Place cursor on row and select desired operation from menu.

ThisDatabase(F9) AllDatabases(SH-F2) Help(F1) End(F10)

The fields in this window are similar to the fields in the Database Summary window. For descriptions of the fields in this window, see online help.

You must configure Ingres Replicator completely from one database to ensure consistency. Ingres Replicator support objects have generated names, based on the table name and a generated table number, which must be unique system-wide. Therefore you must create your replication configuration on one database and use the MoveConfig option to move it to all the other databases in the replication scheme.

**Note:** If you make any changes to the configuration once you have Ingres Replicator set up and running, you must use MoveConfig to copy the changes to the target databases. For example, in general you must not change the collision or error mode for a CDDS on just one of the databases that participates in a CDDS.

## Move Configuration Data

### To move the configuration from the local database to other databases

1. Open the Move Configuration Data window and put the cursor on the row of the database to which you want to move the configuration.
2. Choose ThisDatabase to move the configuration information to the selected database, or choose AllDatabases to move the configuration information to all databases displayed on the window.

Ingres Replicator copies the configuration information, and creates support objects and procedures for the specified databases.

## How the MoveConfig Operation Works

The MoveConfig operation completes the following tasks on the databases it is run against:

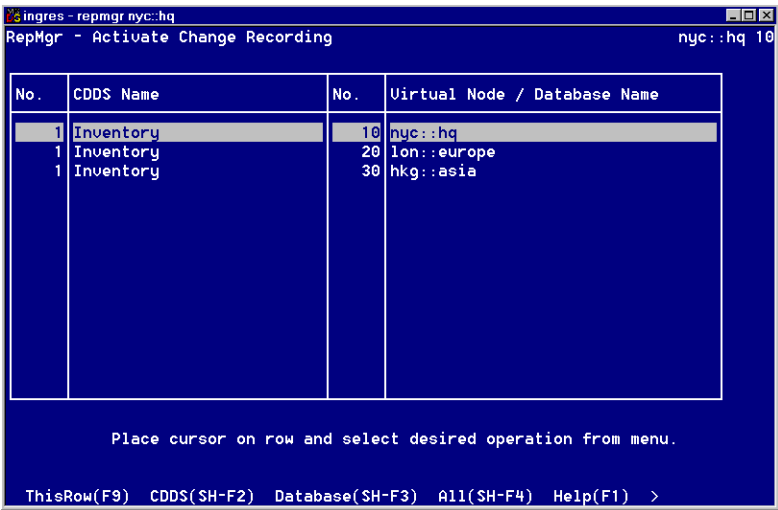
- Copies the configuration data to the target Replicator catalogs
- Makes local context changes to the target data  
Creates support objects; if support objects already exist in the target database, these older objects are first destroyed

The MoveConfig operation does not accomplish any of the following tasks on remote databases:

- Create lookup tables  
You must create, or copy, and populate lookup tables in each database. The MoveConfig operation copies lookup table assignments. For instructions, see Lookup Tables (see page 201).
- Activate the remote databases  
You must activate each CDDS or database from the Activate Change Recording window in order for Ingres Replicator to start working. For activation instructions, see Activate or Deactivate a CDDS or Database (see page 129) .
- Create a mail notification list  
You must run the Replicator Manager on every installation if you want to create individual mail notification lists. For instructions, see Create a Mail Notification List (see page 130).

## Activate Change Recording Window

The Activate Change Recording window (shown here) appears when you choose the Activate option from the Configuration Menu.



For a description of the fields in this window, see online help.

The Activate Change Recording window has the following operations:

**ThisRow**

Activates the selected CDDS on the selected database

**CDDS**

Activates the selected CDDS on all databases that belong to it

**Database**

Activates all CDDSs on the selected database

**All**

Activates all CDDSs on all databases (all rows displayed)



## Activate or Deactivate a CDDS or Database

### To activate or deactivate a CDDS or database

**Note:** Before running Activate, run MoveConfig on the databases to be activated.

1. Open the Activate Change Recording window, place the cursor on the item to be activated or deactivated, and choose a command from the menu line.

The menu line changes to:

Activate   DeActivate   Cancel

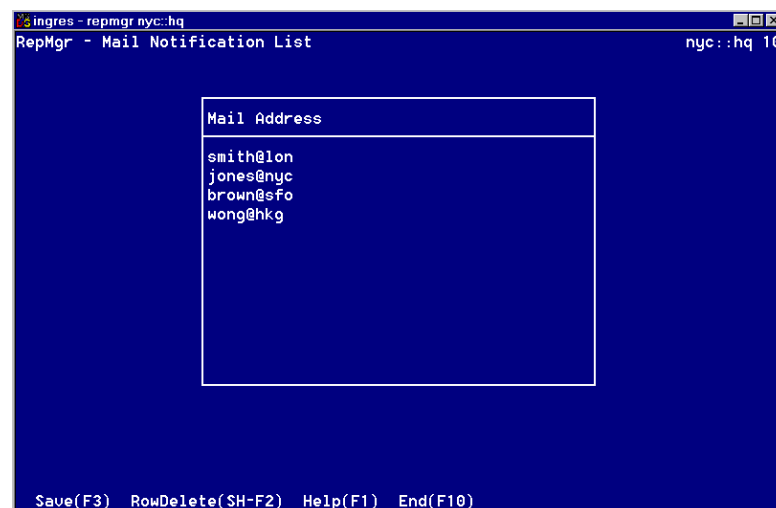
2. Choose Activate or DeActivate.

The chosen items are activated or deactivated accordingly. The Stat column appears at the far right side of the Activate Change Recording window with the word "done" next to the items you activated or deactivated.

## Mail Notification List Window

The Mail Notification List window (shown here) appears when you choose the MailList option from the Configuration Menu.

This window allows you to specify who receives message notification for Replicator Server errors that occur when the connection is to the local database. The mail notification list for each Ingres Replicator database must be set up separately while running the Replicator Manager on it.



For a description of the fields in this window, see online help.

## Create a Mail Notification List

### To create a mail notification list

1. Open the Mail Notification List window, enter the email address of the person to whom the messages are sent.

**Note:** The user name entry is not limited to local users or the intranet. It can also be an Internet user. Local users on the same machine as the Ingres Replicator installation do not need to have a node name specified.

2. Move to the next line in the table.
3. Repeat Step 1 for each email recipient, as needed.
4. Choose Save.

Your entries are saved and you are returned to the Configuration Menu.

## Delete Mail Notification Entries

### To delete a mail notification entry in the Mail Notification List window

Place the cursor on the entry to be deleted and choose RowDelete, Save.

The selected entry is deleted.

# Chapter 6: Using the Replicator Server

---

This section contains the following topics:

[Tools for Performing Replicator Server Tasks](#) (see page 131)

[Tools for Monitoring Replicator](#) (see page 131)

[Install Replicator Service on Windows](#) (see page 132)

[How You Start Replicator Server](#) (see page 133)

[How Server Behavior Is Controlled](#) (see page 134)

[Server Parameters](#) (see page 144)

[Replication Monitoring Using Visual Performance Monitor or Visual DBA](#) (see page 156)

[Replication Monitoring Using Replication Manager](#) (see page 158)

[Advanced Options](#) (see page 168)

The Replicator Server propagates replications from the local distribution queue to the target database. This chapter assumes that you have configured the propagation scheme using either Visual DBA or the Replicator Manager.

## Tools for Performing Replicator Server Tasks

Procedures pertaining to the Replicator Server can be performed using these tools or methods:

- Visual Performance Monitor
- The Performance Monitor in Visual DBA
- The Replication Monitor in Replicator Manager
- Commands entered at the command line

## Tools for Monitoring Replicator

Ingres Replicator monitoring tasks can be performed using either Visual DBA or Visual Performance Monitor. As the smaller and simpler tool, Visual Performance Monitor is preferred.

For a discussion of the role of the Replicator Server, see Replicator Components (see page 22).

## Install Replicator Service on Windows

If you are configuring Ingres Replicator on Windows using the repmgr utility, you must install the Replicator service using the following procedure. The DBA who owns one or more of the replicated databases must perform this procedure.

If you are configuring Ingres Replicator using Visual DBA, you are prompted to install the Replicator service and Visual DBA performs the install task for you.

To install the replicator service using command line, follow these steps:

1. Type the following command:

```
repinst n
```

where  $n$  is the number of servers.

Repinst creates up to  $n$  Windows services to support an equal number of Replicator Servers.

2. Click Administrative Tools in the Control Panel, and then click Services.

The Services dialog appears.

3. Highlight Ingres Replicator (XX) # , where (XX) is the installation identifier, and click Start.

4. Enter the Windows logon password of the DBA.

If there are three DBAs, nyc\_dba, sfo\_dba, and rep\_dba. The nyc\_dba and rep\_dba are in the same New York machine. Each of the three performs the replication startup procedure independently because the rep\_dba cannot complete the procedure for the nyc\_dba, and, of course, the sfo\_dba must complete the procedure separately.

Further, if there are three Replicator Servers in the New York machine—two run from the nyc\_dba account and the third from the rep\_dba account—either nyc\_dba or rep\_dba can run repinst 3; or nyc\_dba can run repinst 2, and rep\_dba runs repinst 3, creating an additional Replicator Server service. The nyc\_dba must go to the Services Control Panel and enter the username and password for servers 1 and 2, and rep\_dba must do the same for server 3.

For more information about the repinst command, see the *Command Reference Guide*.

## How You Start Replicator Server

The Replicator Server allows replication to begin. You are ready to configure and start the Replicator Server after you have installed the Ingres Replicator software and configured your replication scheme.

You must configure and start the server for each database individually through the Performance Monitor in Visual DBA or the Replicator Monitor in Replicator Manager.

**Note:** Before you start a server, Change Recording must have been activated. If Change Recording is not active, there are no transactions for the server to propagate. For more information, see *How Replication Is Activated* (see page 93) or *Activate Change Recording Window* (see page 128).

**Visual DBA and Performance Monitor:** To start the server, follow these steps:

1. Complete the Server Startup Worksheet.  
For details, see *Server Startup Settings Worksheet* (see page 152).
2. Configure startup settings.  
For details, see *Startup Settings Page* (in online help).
3. Start the server.  
For details, see *Starting a Replicator Server* (in online help).

**Replicator Manager:** To start the server, follow these steps:

1. Complete the Server Startup Worksheet.  
For details, see *Server Startup Settings Worksheet* (see page 152).
2. Configure startup settings.  
For details, see *Edit the Configuration File* (see page 164).
3. Start the server.  
For details, see *Start a Server* (see page 165).

## How Server Behavior Is Controlled

The behavior of the Replicator Server is controlled by configuration options, or flags. These flags are contained in the configuration file for each server. Each server can have customized behavior based on the flags set.

When a server is started, it reads its configuration file to set its parameters. The server only reads its configuration file during startup, so although you can edit the file, the changes do not take effect until the server is stopped and restarted. You can, however, send commands to a running server to alter its behavior.

For instructions on sending commands to a running server, see Database Events (see page 172). For more information on server parameters, see Server Parameters (see page 144).

## Server Monitoring Configurations

For effective monitoring, you can configure your servers in the following ways:

- Notify users of server errors through their e-mail accounts
- Specify automatic shutdown of the server at a maximum error count
- Monitor replication activity
- Specify message logging levels

## Mail Alert Setup

Through electronic mail, servers can provide notice to the DBA or other users of any errors as they occur. The level of message logging is dependent on the Log Level flag (-LGLn) setting. For more information, see Message Logging (see page 137).

**Note:** The e-mail notification list feature is not available on Windows.

**Visual DBA:** In VIsual DBA, you set up the mail alert feature by expanding the Replication branch in the Database Object Manager and expanding the Mail Alert Users branch. For more information, see the online help topic Defining Error Mail Destinations. ■

**Replicator Manager:** In Replicator Manager, you set up the mail alert feature using the Mail Notification List window by adding the e-mail account for those users who require e-mail notification. For more information, see [Create a Mail Notification List](#) (see page 130).

The Error Mail (-MLE) flag, which controls the sending of error messages, is set by default. If you set the No Error Mail (-NML) flag, no mail messages are sent. ■

**Note:** When a transaction (for example, an unresolved collision) cannot be transmitted to its target, it remains on the distribution queue. Because the Replicator Server retries propagation every time it rereads the queue (as when new items are queued up), multiple mail notifications are sent. Consider this when deciding whether to use the -MLE or -NML flags and who to include in the mail notification list.

## Error Count Maximum

If the volume of errors reaches a certain point (the default is 100), the Replicator Server shuts down to maintain the integrity of the system and to allow you to troubleshoot problems. The server automatically keeps track of its error count. You can control at what point the server shuts down by specifying a number that represents the maximum error count in the Maximum Error (-EMXn) flag.

If the error count of the server reaches the maximum error count, the server is automatically shut down. Each time the server is started, the error count of the server is reset to 0. If collision resolution is set on your system, be sure to increase the error count maximum to accommodate collision errors. Each collision, even if resolved automatically, counts as an error.

To change the maximum error count at any time, use one of the following methods:

**Visual DBA and Performance Monitor:** Use the Performance Monitor Raise Events page. For more information, see the [Raising Events at the Server Level](#) topic in Visual DBA or Visual Performance Monitor online help.

**Replicator Manager:** Use the Send Database Event Window (see page 166) in Replicator Manager.

## Replication Activity Monitoring

The status of a server is sent to the Activity page in the Visual DBA Performance Monitor window or the Replication Monitor window at server startup, shutdown, and in response to a ping operation. The ping operation returns the status of the server and its error count.

**Note:** It is recommended that you use the `rsstatd` command to display Replicator Server statistics that are cumulative from server startup. Using `rsstatd` does not incur `dbevent` overhead.

You can monitor your system more closely with the Send Notifications flag (`-MON`). The `-MON` flag causes the servers to issue activity notifications.

The `-MON` and Send No Notifications (`-NMO`) flags control whether the server issues activity notifications. Every time a row is propagated to a target and every time a transaction is committed, a notification is sent (or not sent if `-NMO` is selected) that updates the Incoming Records, Outgoing Records, and Activity display fields in the Performance Monitor or Replication Monitor. The `-MON` and `-NMO` flags also cause the server to notify the Performance Monitor or Replication Monitor when an error occurs. The default is `-MON`.

Though the `-NMO` flag stops the server from issuing activity notifications on errors, you can ping a server to obtain its status, overriding any `-NMO` flag.



## Message Logging

When a Replicator Server starts, it creates three log files:

### **replicat.log**

The replicat.log file contains Replicator Server messages. This file writes its buffers after every message is logged, allowing you to see the latest logged messages in real time. Messages are appended to this file from run to run.

### **print.log**

The print.log file contains information that normally goes to the standard output (stdout), such as error output from the DBMS Server. This file contains Replicator Server messages at its specified logging level. This file gets initialized from run to run.

Unlike replicat.log, it is possible that the print.log does not contain the latest error messages.

**Note:** The print.log is not created on Microsoft Windows.

### **commit.log**

The commit.log file is the two-phase commit recovery log for the Replicator Server. This file must not be manipulated because it contains information to resolve leftover willing-to-commit transactions. If the server has been shut down cleanly and there are no willing-to-commit transactions, it is safe to delete this file. Otherwise, *do not* delete it. This file is initialized from run to run, unless there are incomplete two-phase commit transactions, in which case they are recovered first.

Periodically, you must shut down the servers to clean out the log files.

The Log Level (-LGL) flag controls which messages get written to the replicat.log file, while the Print Logging Level (-PTL) flag controls the messages written to the print.log. Both the -PTL and -LGL flags share various settings that indicate the message levels that are logged.

The message levels that require attention are the Level 1 error messages and the Level 3 warning messages. You are advised to set the -LGL flag to receive Level 1 through Level 3 messages.

## Message Logging Levels

Message logging levels 0 through 3 are as follows.

**Note:** Message Levels 4 and 5 are reserved for internal use only and can be set at the request of Customer Support to troubleshoot your system.

### 0

SILENT

Logs no messages.

**Note:** The exception is DBMS Server error messages, for example, informing users of deadlock. The server continues to channel these messages to the standard output (stdout).

### 1

Logs ERRORS

Errors are any occurrence that causes the Replicator Server to alert you to take some action. Error messages include configuration, timeout, transmission, and fatal errors such as the inability to replicate, the issuance of an invalid database event, or the inability to complete a distributed transaction.

### 2

Logs ERROR and INFORMATIONAL messages

Informational messages provide normal server status information such as startup, shutdown, and the setting of flags.

### 3

(Default) Logs ERROR, INFORMATIONAL, and WARNING messages

Warning messages convey some information about a problem or identify an unusual situation that requires your attention, for example, an invalid startup flag in the configuration file.

## Error Handling

The way each Replicator Server detects, handles, and notifies you of errors depends on the error mode of the CDDS. For more information, see *How Errors Are Handled* (see page 56).

Error handling also depends on flag settings that you can change while the server is running.

A server detects errors in these categories:

- Configuration
- DBMS Server
- Replication transmission

## Configuration Errors

If the `runrepl.opt` file settings are invalid or inconsistent, the server logs the error, changes the offending flag to the default setting, and continues. If the `runrepl.opt` file settings are incomplete, that is, required options are missing, the server shuts down.

You can specify whether to bypass the unique key check on the replicated tables when the server first connects to a database.

If primary key checking is enabled, Replicator Server checks for unique keys for the tables assigned to it on the local and remote databases to which it transmits. If key checking discovers an error on the local database, the server is shut down. If an error is found when checking a remote database, the target is quieted.

The primary key-checking options are:

**-NSR**

Enables key checking (default)

**-SCR**

Disables key checking

## DBMS Server Errors

DBMS server errors include timeout, deadlock, and log file full errors. If a DBMS server error occurs during the transmission of data or the distribution of queue information, the servers increase the error count, issue a mail message to those users listed in the Mail Notification List, and retry the transaction.

DBMS server timeout means that database resources are currently unavailable and require a retry. During the retry, the server rolls back the current transaction and re-attempts the timed-out transaction.

**Note:** If the error count reaches the maximum specified in the -EMX flag, the server shuts down.

## Replication Transmission Errors

If errors other than a DBMS server timeout occur during the transmission of data, the server performs the action as determined by the error code of the CDDS. For more information, see *How Errors Are Handled* (see page 56).

## Server Processing Activity

The way the server processes replicated transactions is affected by configuration flags. You can use these configuration flags to do the following:


- Quiet and activate CDDSs, databases, and servers
- Detect lock contention
- Manage memory to fine tune processing performance


## Server, Database, and CDDS Status

You can control traffic in a server by quieting or activating:

- The whole server
- Specific target databases
- Specific CDDSs

To affect a running server, you must send an event to it.

**Visual DBA and Performance Monitor:** For more information, see the Raising Events at the Server Level topic in online help for Visual DBA or Visual Performance Monitor. 

**Replicator Manager:** You can send an event from the Replication Monitor or from a terminal monitor. For more information, see Send Database Event Window (see page 166). 

Servers, databases, or CDDSs can also be quieted by errors, depending on the error mode setting.

When you start a server, all databases and CDDSs are made active. You can, however, start a server in quiet mode by setting the Quiet Server (-QIT) flag as a startup parameter.

The flags that control server, database, and CDDS status are described below:

### **-NQT (Default)**

An active server continually processes transactions for replication.

The Unquiet Server (-NQT) flag is primarily used on a server to resume propagation suspended with the -QIT flag.

### **-QIT**

A quiet server ignores the activity in the replicated database and processes only the pending replication transactions when it receives a `dd_go_servern` database event. When used in conjunction with the -EVT flag, the server also processes transactions on a periodic basis. For more information, see -EVTn Flag (see page 171).

If the CDDS has QuietServer error mode and an error occurs, this flag is set against a server. Use the -NQT flag to activate the server.

### **-SGL**

A single run server processes the pending replication transactions for its server number before it shuts down. This status is generally used to schedule the replication with the operating system's job scheduling system.

### **-CLQ**

If an error occurs that quiets a CDDS or database, the Clear Quiet Targets (-CLQ) flag resumes propagation. When

-CLQ is issued against a server, the server continues quiet; all quieted CDDSs and databases that the server is assigned to are activated for propagation.

**Note:** The value of this flag is stored in the replicator system catalogs; using the `dd_set_server $n$`  event merely causes the replication server to update the value.

### **-QDB $n$**

The Quiet Database (-QDB $n$ ) flag suspends propagation activity to database number  $n$ . This flag can be set using the `dd_set_server $n$`  event or in response to an error (if the error mode QuietDatabase was set).

**Note:** The value of this flag is stored in the replicator system catalogs; using the `dd_set_server $n$`  event merely causes the replication server to update the value.

Use the -QDB $n$  or -CLQ flag with the `dd_set_server $n$`  event to resume propagation activity to a quiet database.

### **-UDB $n$ (default)**

The Unquiet Database (-UDB) flag resumes propagation activity to a database after it is suspended with the Quiet Database (-QDB) flag or by an error (if error mode QuietDatabase was set).

### **-QCD $n$**

The Quiet CDDS (-QCD $n$ ) flag suspends propagation activity to CDDS number  $n$ . This flag can be set using the `dd_set_server $n$`  event or in response to an error (if the error mode QuietCDDS was set).

**Note:** The value of this flag is stored in the replicator system catalogs; using the `dd_set_server $n$`  event merely causes the replication server to update the value.

Use the -UCD $n$  or -CLQ flag with the `dd_set_server $n$`  event to resume propagation activity to a quiet database.

### **-UCD $n$ (default)**

The Unquiet CDDS (-UCD $n$ ) flag resumes propagation activity to a CDDS after it was suspended with the -QCD $n$  flag or by an error (if error mode QuietCDDS was set).

**Note:** The QDB, UDB, QCD, and UCD flags do not take effect immediately; instead, they prevent or allow propagation the next time the servers read transactions from the distribution queue.

## Lock Contention Detection

From the first time a record is manipulated in a distributed transaction until the last record of that transaction is processed, records applicable to the transaction are locked in the local and the target database and remain locked until the final commit (or rollback) statement is issued.

Lock timeout is controlled by the Lock Timeout (-TOT $n$ ) flag, where  $n$  is the number of seconds. Ingres Replicator waits to obtain a lock before aborting the transaction and reprocessing the replication. The recommended value for  $n$  is 30 or greater. The default value is 60. The -TOT $n$  flag can only be set in the configuration file; therefore, to change the value of -TOT $n$ , you must stop and restart the server after editing the configuration file.

**Important!** *If deadlocking is a problem in your environment, replication can make the situation worse. If you are unable to resolve a deadlock, see [Strategies for Avoiding Deadlock](#) (see page 210). If the problem persists, contact Customer Support for assistance.*

## Memory Management

Replicator Servers read the distribution queue into memory. If there is a high volume of replication activity, there is potential for the queue to become so large that the Replicator Server cannot efficiently handle it. To prevent this from occurring, Ingres Replicator provides the Transaction Break Limit (-QBT) and the Queue Read Limit (-QBM) flags, which define how many queue records can be read into memory.

The Replicator Server reads rows from the distribution queue up to the value of the -QBT flag (4000 rows by default) and begins looking for a logical break in the transaction to stop reading and start propagation. The maximum number of rows a Replicator Server can read into memory from the distribution queue is the value of the -QBM flag (5000 rows by default).

**Important!** *If the number of rows in a single transaction exceeds the value of the -QBM flag, the Replicator Server writes an appropriate error message to the replicat.log file and shut down. You need to increase the value of the -QBM flag and restart the Replicator Server.*

The settings you choose for -QBM or -QBT depend on your operating system environment and the nature of your database transactions. If you find you are running out of virtual memory or you are paging excessively in the operating system, use values lower than the default. If you are reaching the default limits and are not having memory problems, increase the default.

**Note:** Once memory is acquired, a Replicator Server continually re-uses it. If a Replicator Server acquires too much memory, dynamically resetting the -QBT or -QBM values has no effect. To change memory usage, a Replicator Server must be shut down and restarted.

## Server Parameters

The Replicator Server has mandatory and optional flags. Mandatory flags must be specified in the configuration file. The mandatory flags are:

- **-SVR***n*
- **-IDB**[*vnnode::dbname*]
- **-OWN***owner*

All other flags are optional; if you do not set them, default values are used.

Use the Server Startup Settings Worksheet (see page 152) to document your flag settings.

## How the Replication Cycle Works

The replication cycle is a cycle of propagation activity performed by the Replicator Server. This cycle is referred to in some server parameter descriptions.

During the replication cycle, the Replicator Server completes the following tasks:

1. Checks the status of closed connections to determine whether they must and can be reopened. For more information, see the -ORT flag.
2. Reads the distribution queue into memory.
3. Transmits rows to the target databases.
4. Checks the status of open connections and closes those that have been inactive. For more information, see the -CTO flag.
5. Waits for the next database event.



## Server Flags

The following table lists each Replicator Server flag, its default, a description of its behavior, and a list of options or associated flags.

Server Flag	Description	Default
-CLQ (unquiet all targets)	<p>Allows suspended propagation activity to resume for all CDDSSs and databases from the local database.</p> <p><b>Note:</b> This flag can only be used on a running server from the Performance Monitor in Visual DBA or the Replication Monitor in the Replicator Manager.</p> <p>For more information, see <i>Server, Database, and CDDSS Status</i> (see page 141).</p>	None
-CTOn (inactive connection timeout)	<p>Sets time before a connection to a database is terminated due to inactivity.</p> <p>The options for this flag are:</p> <ul style="list-style-type: none"> <li>0 - Does not disconnect for lack of activity</li> <li><math>n</math> - Connection to a database is terminated after <math>n</math> seconds of inactivity</li> </ul> <p>The connection to the database is re-established when there is traffic for it, or after the number of replication cycles specified in the -ORT flag (<math>n</math>) have elapsed.</p>	-CTO0
-EMXn (maximum error)	<p>Sets the maximum error count that this Replicator Server can accumulate before shutting down, where <math>n</math> is a non-negative integer.</p> <p>For more information, see <i>Error Count Maximum</i> (see page 135).</p>	-EMX100
-EVTn (event timeout)	<p>Sets the interval, in seconds, between the beginning of replication cycles.</p> <p>The options for this flag are:</p> <ul style="list-style-type: none"> <li>0 - Server does not wait; it processes replications in response to notifications (default)</li> <li><math>n</math> - Server begins a new replication cycle every <math>n</math> seconds, where <math>n</math> ranges from 0 to 9,999</li> </ul> <p><b>Note:</b> This flag has different effects depending on whether the server is active or in quiet mode. For more information, see <i>-EVTn Flag</i> (see page 171).</p>	-EVT0

Server Flag	Description	Default
-IDB[ <i>vnode::</i> ] <i>dbname</i>  (local database name)	Specifies the name of the local database that the server connects to. If applicable, <i>vnode</i> is the virtual node where the local database resides.	Mandatory (no default)
-LGL <i>n</i>  (log level)	<p>Sets the level of messages to be written to the replicat.log file. This file is contained in the server home directory.</p> <p>The options for this flag are:</p> <ul style="list-style-type: none"><li>■ 0 - Silent</li><li>■ 1 - Log error messages</li><li>■ 2 - Log error and informational messages</li><li>■ 3 - Log informational, error, and warning messages (default)</li></ul> <p>For more information, see Message Logging (see page 137).</p>	-LGL3
-MLE  (error mail)	<p>Send mail messages on error to usernames specified in the Error Mail Notification List.</p> <p>Associated flag:</p> <ul style="list-style-type: none"><li>■ -NML - Do not send error mail.</li></ul> <p>For more information, see the Mail Alert Users page in Visual DBA online help or Mail Alert Setup (see page 134).</p>	-MLE
-MON  (send notifications)	<p>Send activity notifications to any process that is registered to receive them. This setting impacts performance.</p> <p>Associated flag:</p> <ul style="list-style-type: none"><li>■ -NMO - Do not send activity notifications.</li></ul> <p>For more information, see Replication Activity Monitoring (see page 136).</p>	-MON
-NML  (no error mail)	<p>Do not send mail messages on error to usernames specified in the Error Mail Notification List.</p> <p>Associated flag:</p> <ul style="list-style-type: none"><li>■ -MLE - Send mail to usernames specified in the Mail Alert Users page in the Error Mail Notification List. (default).</li></ul> <p>For more information, see the Mail Alert Users page topic in Visual DBA online help or Mail Alert Setup (see page 134).</p>	-MLE


Server Flag	Description	Default
-NMO (send no notifications)	<p>Do not send activity notifications to the Performance Monitor in Visual DBA or the Replication Monitor in Replicator Manager. This setting has a positive effect on performance.</p> <p>Associated flag:</p> <ul style="list-style-type: none"> <li>■ -MON - Sends activity notifications to the Performance Monitor in Visual DBA or the Replication Monitor in Replicator Manager (default)</li> </ul> <p>For more information, see Replication Activity Monitoring (see page 136).</p>	-MON
-NQT (unquiet server)	<p>Resumes suspended propagation in the server. The server processes the replication as activity happens on the local database.</p> <p>Associated flags:</p> <ul style="list-style-type: none"> <li>■ -QIT - Quiets the server</li> </ul> <p>For more information, see Server, Database, and CDDS Status (see page 141).</p>	-NQT
-NSR (check for keys)	<p>Checks for the existence of unique keys in the local and remote databases for tables assigned to the Replicator Server. Checking occurs at Replicator Server startup or when first connecting to a remote database.</p> <p>Associated flag:</p> <ul style="list-style-type: none"> <li>■ -SCR - No checking for the existence of unique keys</li> </ul> <p>For more information, see Configuration Errors (see page 139).</p>	-NSR
-ORT $n$ (retry opening of target database)	<p>Sets the number of replication cycles that are to elapse before the server tries to open any target databases that are currently closed.</p> <p>The options for this flag are:</p> <ul style="list-style-type: none"> <li>■ 0 - Does not try to open closed target databases (default)</li> <li>■ <math>n</math> - Tries to open the connection to the target database after <math>n</math> replication cycles</li> </ul>	-ORT0
-OWNowner (local database owner)	Specifies the owner (DBA) of the local database.	Mandatory (no default)


Server Flag	Description	Default
-PTL <i>n</i> (print logging level)	<p>Sets the level of messages to be written to standard output, which is normally redirected to the print.log file. This is a debugging feature.</p> <p>The options for this flag are:</p> <ul style="list-style-type: none"><li>■ 0 - Silent</li><li>■ 1 - Log error messages</li><li>■ 2 - Log error and informational messages</li><li>■ 3 - Log informational, error, and warning messages (default)</li></ul> <p>For more information, see Message Logging (see page 137).</p>	-PTL3
-QBM <i>n</i> (queue read limit)	<p>This is a memory throttle, where <i>n</i> is a positive integer between 1 and 999,999. Sets the absolute number of rows that can be read into memory from the distribution queue before the Replicator Server stops reading. The value of -QBM must be greater than or equal to -QBT.</p> <p>For more information, see Memory Management (see page 143).</p>	-QBM5000
-QBT <i>n</i> (transaction break limit)	<p>This is a memory throttle, where <i>n</i> is a positive integer between 1 and 999,999. Sets the maximum number of rows that can be read from the distribution queue before the Replicator Server looks for a logical break in the transaction to maintain consistency. The value of -QBT must be less than -QBM.</p> <p>For more information, see Memory Management (see page 143).</p>	-QBT4000
-QCD <i>n</i> (quiet CDDS)	<p>Quiets the CDDS where <i>n</i> is the CDDS number; transactions from the local database to the selected CDDS cannot be propagated.</p> <p>Associated flag:</p> <p>-UCD<i>n</i> - Unquiets the selected CDDS</p> <p>For more information, see Server, Database, and CDDS Status (see page 141).</p>	Not applicable

Server Flag	Description	Default
-QDB $n$ (quiet database)	<p>Quiets the database where <math>n</math> is the database number; transactions from the local database to the selected database cannot be propagated.</p> <p>Associated flag:</p> <p>-UDB<math>n</math> - Unquiets the selected database</p> <p>For more information, see Server, Database, and CDDS Status (see page 141).</p>	Not applicable
-QIT (quiet server)	<p>Quiets the Replicator Server. The Replicator Server ignores notifications that signify activity is taking place in the local database. The Replicator Server only processes replicated transactions as a result of a <code>dd_go_server</code> database event. A quiet server can also process transactions in response to periodic timers set up with the -EVT flag.</p> <p>Associated flags:</p> <p>-NQT - Activates the Replicator Server (default)</p> <p>For more information, see Server, Database, and CDDS Status (see page 141) and Scheduling Servers (see page 170).</p>	-NQT
SCR (no key checking)	<p>Specifies that no checking for the existence of unique keys for tables assigned to the server on the local and remote databases occurs upon connection.</p> <p>Associated flag:</p> <p>-NSR - Checks for the existence of unique keys at server startup (default)</p> <p>For more information, see Configuration Errors (see page 139).</p>	-NSR
-SGL (single run)	<p>Processes the pending replication transactions and shuts down. This flag permits server scheduling at an operating system level.</p> <p>For more information, see Server, Database, and CDDS Status (see page 141), Scheduling Servers (see page 170), and -SGL Flag (see page 170).</p>	-NQT

Server Flag	Description	Default
-SVR $n$ (server number)	<p>The number assigned to a server. The number is determined by the configuration set in Visual DBA or Replicator Manager.</p> <p>Options for this flag:</p> <ul style="list-style-type: none"><li>▪ <math>n</math> - Defines the number of the Replicator Server, where <math>n</math> ranges from 1-999, corresponding to the name of the directory where the server's configuration and log files are located. The number also defines to which targets the Replicator Server transmits.</li></ul> <p>The default number of installed servers is 10, but more can be added if events and additional directories are created to support the servers. For more information, see <i>How You Assign Server Numbers Greater Than Ten</i> (see page 169).</p>	Mandatory (no default)
-TOT $n$ (lock timeout)	<p>Allows <math>n</math> seconds before a server can terminate a transaction that is waiting for a lock held by another user.</p> <p>Options for this flag:</p> <ul style="list-style-type: none"><li>▪ 0 - Server waits for a lock indefinitely</li><li>▪ <math>n</math> - Server waits <math>n</math> seconds, where <math>n</math> ranges from 1 to 999, for a lock.</li></ul> <p>A setting of 30 or greater is recommended.</p> <p>For more information, see <i>Lock Contention Detection</i> (see page 143).</p>	-TOT60
-TPC $n$ (two-phase commit)	<p>Controls whether two-phase commit is used or not. For more information, see <i>Two-Phase Commit</i> (see page 28).</p> <p>The options for this flag are:</p> <ul style="list-style-type: none"><li>▪ 0 - Two-phase commit is not used to secure a transaction that has been propagated. Uncompleted transactions must be recovered manually from the commit log file.</li><li>▪ 1 - Two-phase commit is used to secure the propagation of transactions (default)</li></ul> <p><b>Important!</b> <i>Without two-phase commit, there is an increase in the possibility of data integrity problems because of uncompleted transactions.</i></p> <p>For situations where you must use -TPC0, see the appendix "VMS Cluster Support."</p>	-TPC1

Server Flag	Description	Default
-UCD <i>n</i> (unquiet CDDS)	<p>Enables propagation to a CDDS where <i>n</i> is the CDDS number; transactions from the local database to the selected CDDS can be propagated.</p> <p>Associated flag:</p> <ul style="list-style-type: none"> <li>▪ -QCD<i>n</i> - Quiets the CDDS</li> </ul> <p>For more information, see Server, Database, and CDDS Status (see page 141).</p>	Not applicable
-UDB <i>n</i> (unquiet database)	<p>Enables propagation to a database where <i>n</i> is the database number; transactions from the local database to the selected database can be propagated.</p> <p>Associated flag:</p> <ul style="list-style-type: none"> <li>▪ -QDB<i>n</i> - Quiets the database</li> </ul> <p>For more information, see Server, Database, and CDDS Status (see page 141).</p>	Not applicable

**Visual DBA and Performance Monitor:** You set flags by expanding the Replication branch in the Performance Monitor window, expanding the desired database, and selecting the desired server. (See the Startup Settings Page topic in Visual DBA or Visual Performance Monitor online help. )

**Replicator Manager:** To set flags using the Replication Monitor, see Edit the Configuration File (see page 164). Using Replication Monitor, unless otherwise noted, all flags can be set both through database events and directly in the configuration file. 

## Server Startup Settings Worksheet

You can use the Server Startup Settings Worksheet to document your server startup parameters. All flags that can be set in the configuration file for startup are listed on the worksheet. You can choose only one flag per category. Unless you specify differently, default values are used. For descriptions of all server parameters, see *Server Parameters* (see page 144).

The server information fields on the worksheet are as follows:

### Server Directory

Identifies the directory in which the server's configuration and log files are located. The default directory for RepServer number *n* is:  
`%II_SYSTEM%\ingres\rep\servers\servern`

### Server No.

Specifies the number assigned to the server. The number can be in the range of 1-999 and must conform to the directory name where the server's configuration and log files are located. For more information on server numbers, see *Replicator Server Assignment* (see page 58). For server numbers greater than 10, see *How You Assign Server Numbers Greater Than Ten* (see page 169).

### Local Database Name

Specifies the name of the database. For more information, see *Database Worksheet* (see page 68).

### Database No.

Specifies the number assigned to the database in the CDDS configuration process. For more information, see *Database Worksheet* (see page 68).

The following is an example of the Server Startup Settings Worksheet:

<b>Server Directory:</b> _____	<b>Server No:</b> _____	
<b>Local Database Name:</b> _____	<b>Database No.:</b> _____	
<b>Server Parameter/Behavior</b>	<b>Default</b>	<b>Setting</b>
Server number -SVR0, -SVR <i>n</i>	Mandatory	
Local database -IDB[ <i>vnode::</i> ] <i>dbname</i>	Mandatory	
Local database owner -OWN <i>owner</i>	Mandatory	



<b>Server Directory:</b> _____	<b>Server No.:</b> _____
<b>Local Database Name:</b> _____	<b>Database No.:</b> _____
Lock timeout	-TOT60
-TOT <i>n</i>	
Inactive connection timeout	-CTO0
-CTO0, -CTO <i>n</i>	
Event timeout	-EVT0
-EVT0, -EVT <i>n</i>	
Send activity notifications	-NMO
-MON, -NMO	
Retry opening target databases	-ORT0
-ORT0, -ORT <i>n</i>	
Log messages to stdout	-PTL3
-PTL0, -PTL1, -PTL2, -PTL3	
Log messages	-LGL3
-LGL0, -LGL1, -LGL2, -LGL3	
Mail messages on error	-MLE
-MLE, -NML	
Maximum error count	-EMX100
-EMX <i>n</i>	
Active quiet	-NQT
-NQT, -QIT	
Two-phase commit	-TPC1
-TPC0, -TPC1	
Check unique keys	-NSR
-NSR, -SCR	
Queue read limit	-QBM50000
-QBM <i>n</i>	
Transaction break limit	-QBT4000
-QBT <i>n</i>	

<b>Server Directory:</b> _____	<b>Server No:</b> _____
<b>Local Database Name:</b> _____	<b>Database No.:</b> _____
Single-run server	None
-SGL	

The following illustrates a completed Server Startup Settings Worksheet:

Server Directory: <u>\\SYSTEM\ingres\rep\servers\server2</u> Server No: <u>2</u>		
Local Database Name: <u>nyc::hg</u> Database No: <u>10</u>		
Server Parameter/ Behavior	Default	Setting
Maximum error count -EMX <sub>n</sub>	-EMX100	-EMX50
Active, quiet -NQT, -QIT	-NQT	
Two-phase commit -TPC0, TPC1	-TPC1	
Check unique keys -NSR, -SCR	-NSR	
Queue read limit	-QBM5000	

Server Startup Settings Worksheet		
Server Directory: <u>\\SYSTEM\ingres\rep\servers\server2</u> Server No: <u>2</u>		
Local Database Name: <u>nyc::hg</u> Database No: <u>10</u>		
Server Parameter/ Behavior	Default	Setting
Server number -SVR0, -SVR <sub>n</sub>	Mandatory	-SVR1
Local database -IDB[vnode::]dbname	Mandatory	-IDBhg
Local database owner -OWNowner	Mandatory	-OWNtestenv
Lock timeout -TOT <sub>n</sub>	-TOT60	
Inactive connection timeout -CTO0, -CTO <sub>n</sub>	-CTO0	-CTO1800
Event timeout -EVT0, -EVT <sub>n</sub>	-EVT0	
Send activity notifications -MON, -NMO	-NMO	
Retry opening target database -ORT0, -ORT <sub>n</sub>	-ORT0	-ORT50
Log messages to stdout -PTL0, -PTL1, -PTL2, -PTL3	-PTL3	
Log messages -LGL0, -LGL1, -LGL2, -LGL3	-LGL3	
Mail messages on error -MLE, -NML	-MLE	

## Replication Monitoring Using Visual Performance Monitor or Visual DBA

Once you have installed and configured your replication scheme and propagated it to appropriate databases, you can use the Visual Performance Monitor tool or the Performance Monitor in Visual DBA to monitor your replication activities. Because it is simpler to use, Visual Performance Monitor is the preferred tool for this task.

### Monitor Replication Activity

To monitor replication activity for all running servers, view the Performance Monitor Activity page when a database is selected. The Activity page shows the total number of outgoing and incoming records, the total number of transactions associated with the incoming and outgoing records, and the queue activity for each database that the servers support.

**Note:** The counts of incoming and outgoing records are transient. They show activity only since the time the Performance Monitor was first entered or since the last time the Now button was chosen to reset the Starting Time to the current time in the Records Activity Since group.

The following illustration shows the Activity page in Performance Monitor:

Replication on Database hq

Activity | Servers | Raise Events | Collisions | Integrity | Distrib

Input Queue: 0 Distribution Queue: 6

Records Activity Since

Now Starting Time: 02/09/99 11:48:18

Outgoing / Incoming Records Summary:

	Insert	Update	Delete	Total	Transactions
Outgoing	3	1	1	5	5
Incoming	0	0	0	0	0
Total	3	1	1	5	5

Detail per Database / Table:

Outgoing Incoming Total Outgoing Incoming Total

Database	Inserts	Updates	Deletes	Total	Tran
20 lon::europe ...	3	1	1	5	

For detailed instructions on performing replication monitoring tasks, see the Visual DBA or Visual Performance Manager online help.

## View Server Statuses

In monitoring replication, you need to check the statuses of the Replicator Servers. You can ping each server to obtain its status using the Servers page in the sub-branches under the Replication branch. For details, see the online help topic, *Viewing Replication Server Status*.

## Raise Events at the Database Level

To issue database events for *all* servers that are defined for a replication scheme at the database level, view the Raise Events page. For details, see the online help topic, *Raising Events at the Database Level*. Such events include stopping a server or processing pending replications. To raise an event for an individual server, see *Raise Events at the Server Level*.

## Raise Events at the Server Level

To issue database events for an individual Replicator Server, view the Raise Events page. (For details, see the online help topic, *Raising Events at the Server Level*.) Such events include stopping a server or processing pending replications. To raise an event for all servers that are defined for a replication scheme at the database level, see *Raise Events at the Database Level*.

## View the Queue Collision Report

On the Collisions page, you can generate the Queue Collision Report, which lists all collision conflicts that exist on the local database. For details, see the online help topic, *Viewing Collision Conflicts*. By reviewing this report, you can start the process of resolving the collision.

For sample report output, see *Queue Collision Report* (see page 188).

## Create and View the Table Integrity Report

The Integrity page allows you to specify a report to compare a table in two different databases. For details, see the online help topics, *Integrity Page* and *Verifying Table Integrity*.

You can specify the criteria for the comparison based on transaction begin and end time, and if desired, a CDDS value to designate a particular distributed data set. You can also specify the sort order of the report by replicated column or transaction time.

For sample report output, see *Table Integrity Report Window* (see page 189).

## View the Distributed Configuration Checker Report

The Distributed Configuration Checker Report is used to verify that a local database's replication configuration and its remote counterparts are consistent with one another. For details, see the online help topic, Distrib Page. The report pinpoints any significant differences between the replication setups of the individual databases.

For a sample of the Distributed Configuration Checker Report output, see Distributed Configuration Checker Report (see page 192).

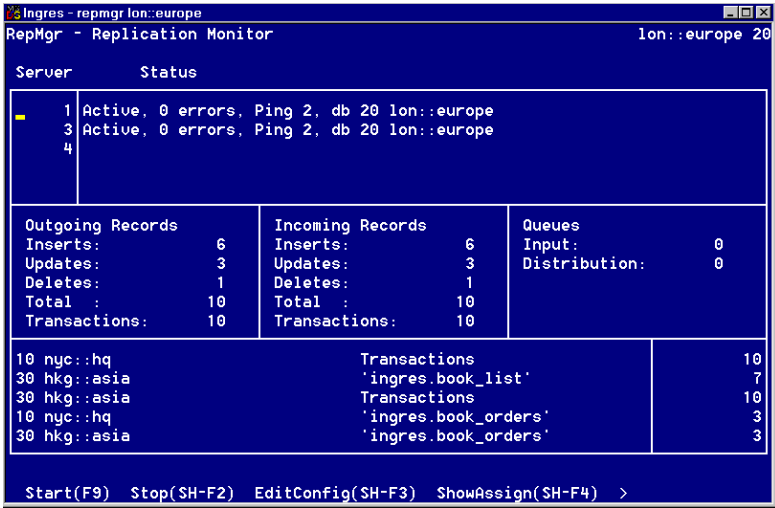
**Tip:** *It is recommended that you run this report when problems occur, or periodically, to ensure that the remote configurations have not been changed without authorization.*

## Replication Monitoring Using Replication Manager

You perform replication monitoring in Replication Manager by selecting the Monitor option from the Replicator Manager main menu window.

### Replication Monitor Window

The Replication Monitor window appears when you select the Monitor option from the Replicator Manager main menu window.



Server	Status
1	Active, 0 errors, Ping 2, db 20 lon::europe
3	Active, 0 errors, Ping 2, db 20 lon::europe
4	

Outgoing Records		Incoming Records		Queues	
Inserts:	6	Inserts:	6	Input:	0
Updates:	3	Updates:	3	Distribution:	0
Deletes:	1	Deletes:	1		
Total :	10	Total :	10		
Transactions:	10	Transactions:	10		

10 nyc::hq	Transactions	10
30 hkg::asia	'ingres.book_list'	7
30 hkg::asia	Transactions	10
10 nyc::hq	'ingres.book_orders'	3
30 hkg::asia	'ingres.book_orders'	3

Start(F9) Stop(SH-F2) EditConfig(SH-F3) ShowAssign(SH-F4) >

For all active servers, the Replication Monitor window shows the total number of outgoing and incoming records, the total number of transactions associated with the incoming and outgoing records, and queue information for the local database.

**Note:** You can monitor Ingres Replicator outgoing record counts since a Replicator Server was started by using the command line utility, `rsstatd`. For more information about the `rsstatd` command, see the *Command Reference Guide*.

You can use the Replication Monitor window to:

- Configure servers by setting server flags
- Start and stop a server
- Ping all servers
- View a server's CDDS and database assignments
- View the status of active servers
- Display the current activity of the servers
- Issue database events to override server flags

**Note:** The counts of incoming and outgoing records are transient. They show activity only since the time the Replication Monitor was first entered or since the last time that the Clear menu option was chosen. The Replication Monitor window uses the FRS time-out feature to determine when to look for server activity. The monitor does not attempt to retrieve status information until the keyboard has been idle for five seconds.

The Replication Monitor window has the following items:

#### **Server Status Box**

A running server is either active or quiet. An active server processes distributed transactions as they are created or received. A quiet server only processes transactions at the request of a `dd_go_server` database event or periodically when the Event Timeout flag is used.

Initially, if a server is not running, the window does not display a status. If a server does not report back shortly after it has been started, a warning message is displayed. The status is updated when the server is started, in response to a ping or any other database event, and when the server encounters an error. When the server shuts down normally, an appropriate status message is displayed.

#### **Error Count**

Number of Level 1 errors (configuration, time-out, transmissions, and fatal errors) the server has encountered. If the number of errors reaches the -EMX startup parameter value, the server shuts down.

### Ping

Number of times the server has been pinged. A ping requests all servers to respond with a status message. If this increases the ping number of the server, this indicates the server is running.

**Note:** You ping a server to determine whether it is running. The server status on the monitor indicates how the server is set up to process transactions, but it does not indicate whether the server is processing those transactions.

### Database

The number, virtual node, and name of the database to which the server is connected.

### Outgoing Records Box

Number of records that have been processed and transmitted by the servers attached to this source database since you entered the Replication Monitor window. A replicated insert, update, or delete operation that is successfully executed and rolled back is still reflected in the count.

The Transaction field displays the total number of transactions originating from this source database.

### Incoming Records Box

Number of records that have arrived at this database from other replicated databases since you entered the Replication Monitor window. A replicated insert, update, or delete operation that is successfully executed and rolled back is still reflected in the count.

The Transaction field displays the total number of transactions executed by remote Replicator Servers against this database.

**Note:** Fields in the Incoming and Outgoing boxes with zero values are reflected with spaces.

### Queues Box

Contains queue information entering the Replication Monitor window and when selecting the Queues operation. When selected, this box displays in the Input and Distribution fields a snapshot count of the records that are currently in the database's input queue and distribution queue.

**Note:** This information does not update automatically; it is only updated when you use the Queues operation.

### Activity Display Box

The display area under the three boxes breaks down the incoming and outgoing totals in the following formats:

*db\_no* *vnode::dbname* *'tablename'* *count*

and

*db\_no* *vnode::dbname* Transactions *count*



In the first format, the *tablename* field reflects the activity that is occurring on the table of that name. The count field reflects both the incoming records sent to this database from the target indicated in *db\_no* field, as well as the outgoing records sent to the target by *tablename* from this database.

If the display is in the second format, the count field reflects the number of transactions that the servers on this database have successfully executed on that database. For example:

10	nyc::hq	Transactions	10
30	hkg::asia	'rep_dba.book_list'	7

In the first line, assume you have run Replicator Manager against database 20. There are 10 transactions that the server on database 20 has successfully executed on database 10.

In the second line, assume you have run Replicator Manager against database 20. There are 7 records with activity on *tablename* *book\_list* owned by *rep\_dba* that have either been incoming to database 20 from database 30 or outgoing to database 30 from database 20.

## Monitor Menu Map

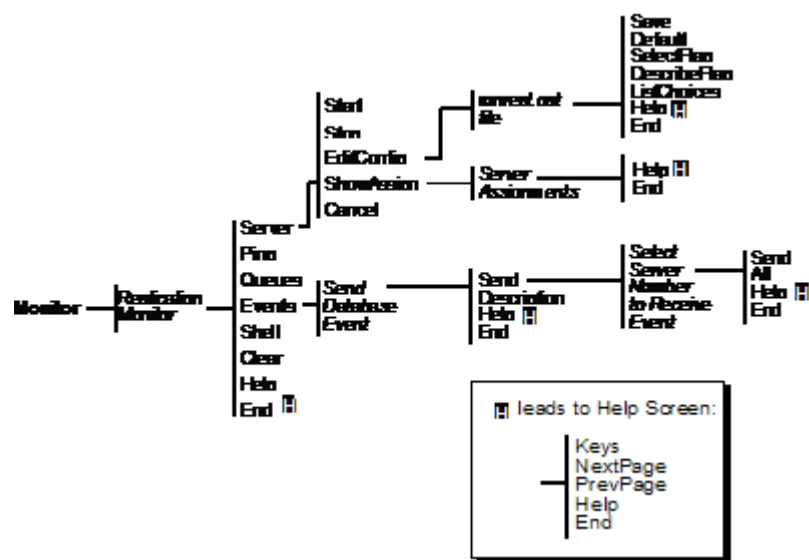
The following map shows each selection that is accessible from the Replication Monitor.

The menu map uses the following conventions:

**Bold typeface**—Indicates items selected from boxed list window areas

**Regular typeface**—Indicates items selected from menu list at the bottom of the window

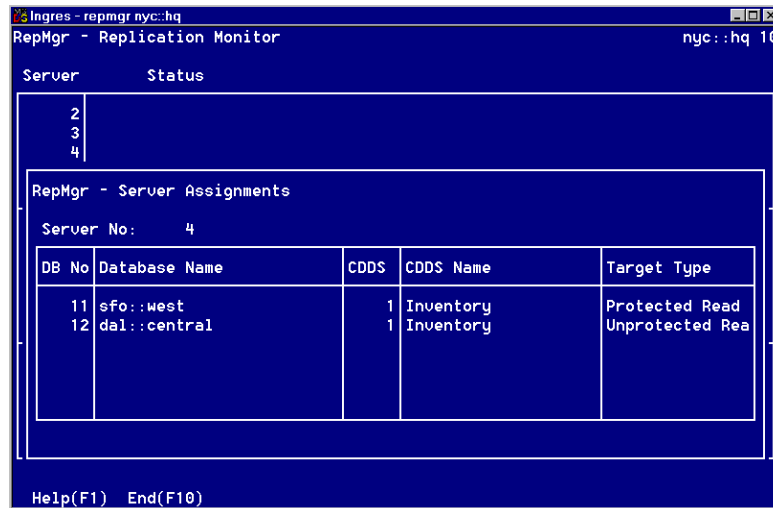
*Italics typeface*—Indicates window names



## Server Assignments Window

The Server Assignments pop-up window appears when you choose ShowAssign from the Replication Monitor window.

This window displays the databases and CDDs to which the particular Replicator Server transmits.



The Server Assignments window has the following fields:

### DB No

The number of the database to which the server transmits

### Database Name

The database's virtual node and name

### CDDS

The CDDS number to which the server transmits for the listed database

### CDDS Name

The CDDS name

### Target Type

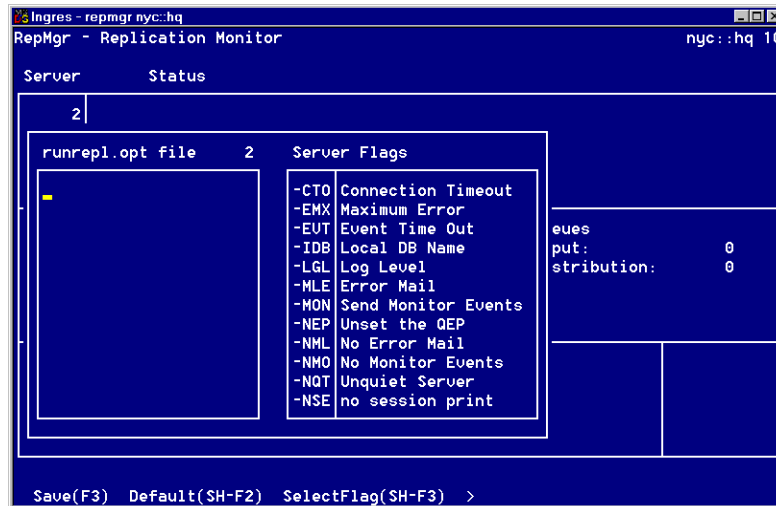
The target type for the listed CDDS

## Edit the Configuration File

### To edit a Replicator Server's configuration file

1. Open the Replication Monitor window, place the cursor on the number of the applicable server, and choose Servers, EditConfig.

The runrepl.opt file pop-up, containing the configuration file for the server and a list of server flags, appears as shown in the following example:



**Note:** For a brief description of a flag, move the cursor to the flag and choose DescribeFlag.

2. Edit the file using one of these methods:
  - Type over existing text.
  - Clear previous flag settings, if any, and replace them with the default settings by choosing the Default operation.
  - Select flags from the Server Flags list:
    - a. Move to the Server Flags field.
    - b. Place the cursor on desired flag.
    - c. Choose SelectFlag. For certain flags, complete the entry by adding parameters.
3. Choose Save.

The file is saved and you are returned to the Replication Monitor window.

## Start a Server

Before you can start a server, you must have configured the server's configuration file.

### To start a server

Open the Replication Monitor window, place the cursor on the number of the server that needs to be started, and choose Server, Start.

A message is displayed indicating that the server is starting.

Within a few seconds, a confirmation message similar to the one below is displayed beside the server number in the Replication Monitor window:

```
| 2|Active, 0 errors, Ping 1, db 10 nyc::hq |
```

If the server fails to start after about two minutes, the message, "The Server could not be started" appears instead.

## Check the Queues

### To check the status of the database input and distribution queues

Open the Replication Monitor window and choose Queues.

Within a few seconds, the Queues box on the Replication Monitor displays current counts of the rows in the input and distribution queues.

## Ping Servers

Sending a ping event allows you to check whether the connection to each server is still active.

### To ping all running servers

Open the Replication Monitor window and choose Ping.

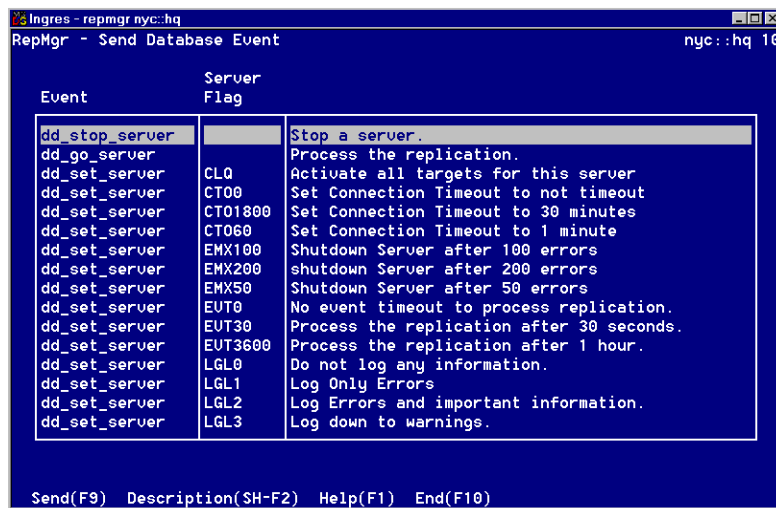
Within a few seconds, the count on the Ping field of the server status line for each running server must increment by 1. If the count does not increment, the server is not running or is too busy. Each event sent to a server also increments the ping count.

## Send Database Event Window

The Send Database Event window is displayed when you choose the Events menu option from the Replication Monitor window, as shown in the following example.

The Send Database Events window lets you do the following:

- Reconfigure a server while it is running by changing its startup flags. For a list of startup flags that can be modified through Replicator Manager, see Server Parameters (see page 144).
- Issue database events to perform such tasks as shutting down servers or processing pending replications.



The Send Database Events window has the following fields:

### Event

Shows the name of the database event

### Server Flag

Shows the parameter value of the database event

## Change Server Startup Flags Dynamically

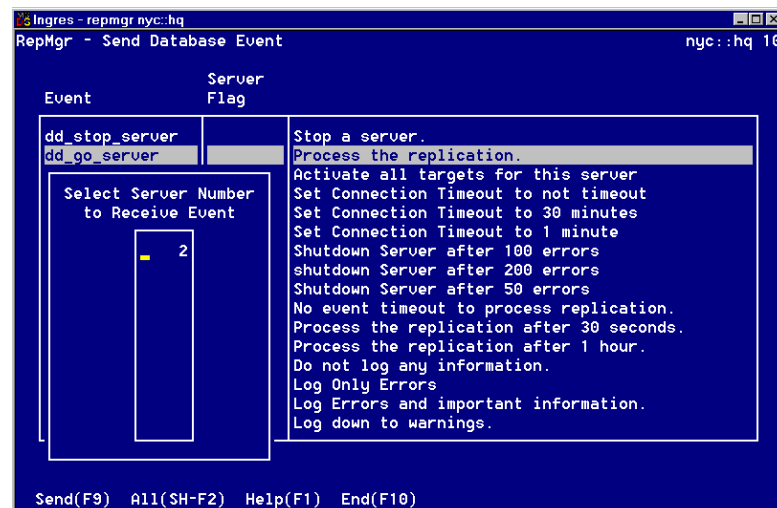
Changing startup flags dynamically on a server overrides the default or the assigned setting for the flag in the server's configuration file. For a list of configuration flags that you can change dynamically, see *Server Parameters* (see page 144). For other methods of sending events to a running server, see *Database Events* (see page 172).

### To change the parameters of a running server

1. Open the Send Database Event window, highlight the event corresponding to the action you want the server to take, and choose Send.

If no servers are currently running, a message is issued and no pop-up is displayed.

The Select Server Number to Receive Event pop-up appears, as in this example:



2. Designate the servers that receive the event:

To choose all servers, choose All.

To choose one server, move the cursor to the server number in the Select Server Number to Receive Event window and choose Send.

The message "Event issued" is displayed in the window to confirm that the event was sent.

## Stop a Server

### To stop a server

Open the Replication Monitor window, highlight the server that needs to be stopped, and choose Servers, Stop.

A database event is sent to the server requesting it to shut down. A message is displayed indicating that the server is stopping.

When the server shuts down, a confirmation message similar to the one below is displayed beside the server number in the Replication Monitor:

```
| 2|The server has shutdown |
```

**Note:** Before a server shuts down, it completes processing any current replicated transaction. 🗑

## Advanced Options

Advanced options for Replicator Server include:

- Running Replicator Servers with server numbers greater than ten
- Scheduling servers using server parameters and database events
- Using commands and database events to affect server behavior from sources other than Visual DBA and the Replicator Manager



## How You Assign Server Numbers Greater Than Ten

The Ingres Replicator setup script and the repcat utility support only server numbers 1 through 10. If you want to run a Replicator Server with a number greater than 10, you need to follow this process:

1. Assign the desired server number (N) in the Server field:

In Visual DBA, use the Define Replicator Databases dialog.

In Replicator Manager, use the CDDS Databases and Servers window.

A row is added to the dd\_servers catalog if N does not already exist.

2. Using the SQL Terminal Monitor, connect as the DBA of each replicated database with Full Peer or Protected Read-only targets and issue the following statements (replacing N with your server number):

```
CREATE DBEVENT dd_set_serverN;  
CREATE DBEVENT dd_serverN;  
CREATE DBEVENT dd_go_serverN;  
CREATE DBEVENT dd_stop_serverN;
```

3. Create the server work directory from the command line. For example:

**Windows:**

```
md %II_SYSTEM%\ingres\rep\servers\serverN
```

**UNIX:**

```
mkdir $II_SYSTEM/ingres/rep/servers/serverN
```

**VMS:**

```
create /dir II_SYSTEM:[ingres.rep.servers.serverN]
```

## Scheduling Servers

You can control *when* Ingres Replicator propagates changes to the target databases. If it is not critical to have the CDDS synchronized, you can turn on the servers at slow or no usage times. The main benefit of this strategy is that it conserves processing capacity at critical times; Ingres Replicator consumes resources.

You control replication timing through the servers. If the Replicator Server for a given CDDS is started and active, it is continually propagating changes to the target databases. If the Replicator Server is stopped or started but quiet, changes to the CDDS accumulate in the distribution queue until the Replicator Server is started or activated.

You can use the following methods to schedule Ingres Replicator by starting the replication cycle:

- At specific times using the -SGL flag
- At specific times using the -QIT flag and raising an event from an application
- At given intervals using the -QIT and -EVT flags

**Note:** To stop a server to perform maintenance on your system, see the online help topic Status Page [Performance Monitor window, Replication branch] for details. You can also stop servers in a terminal monitor or through your own customized shut down script by issuing the correct event to the server. For more information, see Database Events (see page 172).

### -SGL Flag—Run Replication Once

The -SGL flag configures the server to process its replication queues once before shutting down. You can schedule the replication with your operating system's job scheduling system—cron under UNIX, at on Windows, submit on VMS.

Place the -SGL flag in the server's runrepl.opt file.

### -QIT Flag—Run Server in Quiet Mode

The -QIT flag starts the Replicator Server in quiet mode.

You can schedule Replicator Server activity with database events by using the -QIT flag setting, and use an Ingres application or script to raise a `dd_go_server[n]` database event at specific times, where *n* is the number of the Replicator Server you want to start.

### -EVTn Flag—Use Event Timeout

You can use the event timeout flag, -EVTn, to affect server activity. The way the -EVTn flag affects server scheduling depends on several factors. The most common scenarios for using the -EVTn flag to schedule servers are:

#### **Replicator Server with the -NQT flag set**

The -EVTn flag controls the duration of the replication cycle in an idle database. On a relatively active database, -EVTn has little effect.

#### **Replicator Server with the -QIT flag set**

The -EVTn flag specifies that the Replicator Server reads the distribution queue only every *n* seconds.

**Note:** This configuration can cause significant backlogs because the Replicator Server does not take into account database activity or how large the backup is.

### rpserver Command—Start a Replicator Server

The rpserver command starts an individual Replicator Server from the operating system prompt. The command reads in parameters from the configuration file, which must be present in the corresponding server directory.

This command has the following format:

```
rpserver n
```

***n***

Is the number of the server to be started.

## Database Events

You can affect a server by using database events. Database events override the flag settings in the configuration file. An Ingres application must connect to the server's database to issue the event. You can affect a server in any mode: stopped, started and active, or started and quiet.

You can send events from the following sources:

- Visual DBA, Visual Performance Monitor, or SQL Scratchpad window  
For more information, see the online help topic Raising Events at the Server Level.
- Replication Monitor's Send Database Event Window (see page 166).
- Terminal Monitor
- Ingres 4GL, embedded SQL, or OpenAPI application

The database events you can use to affect server behavior are:

### **dd\_stop\_server[n]**

Stops the server, where  $n$  is the number of the server. If  $n$  is not specified, all started servers on that database are requested to shut down.

For example:

```
dd_stop_server7
```

**Note:** Servers shut down only after they finish current processing activities.

### **dd\_go\_server[n]**

Directs a quiet server to process pending replication transactions, where  $n$  is the number of the server. If  $n$  is not specified, all started servers on that database processes pending transactions.

For example:

```
dd_go_server7
```

You can use the `dd_go_server[n]` event to schedule replication at specific times with a quiet server. For more information, see [Scheduling Servers](#) (see page 170).

### **dd\_ping**

Sends a ping event to all servers.

For more information, see [Ping Servers](#) (see page 165).

**dd\_set\_server[n] '-flag'**

Overrides the server parameters set in the configuration file, where  $n$  is the number of the server and flag is a server parameter that can be set dynamically. For more information about server flags, see Server Parameters (see page 144).

For example:

```
dd_set_server7 '-NQT'
```

If  $n$  is not specified, all started servers connected to that database are altered.



# Chapter 7: Maintaining the Replicator

---

This section contains the following topics:

[Maintenance of Replicated Tables and Databases](#) (see page 175)

[Reconfiguration of the Replicator](#) (see page 177)

[Collision Resolution](#) (see page 182)

[Disaster Recovery](#) (see page 185)

[Resolver Reports Menu \(Replicator Manager\)](#) (see page 186)

[How to Deactivate Replication](#) (see page 199)

## Maintenance of Replicated Tables and Databases

The tasks to be performed when you create or maintain your replicated system include stabilizing the system, copying data into a replicated table, and removing replication objects.

### How You Stabilize the Replication System

To perform Ingres Replicator maintenance, your system must be in a stable state. To achieve system stability, perform these steps:

1. Exclude users from replicated databases.
2. Run servers until all queues are empty.

**Note:** Emptying the server queues is a way to keep replicated databases consistent. To verify that the databases are consistent, run an Integrity Report against tables and databases that are in question. For more information, see Table Integrity Report Window (see page 189).

3. Shut down Replicator Servers.


## Copying Data into a Replicated Table

You can use the copy statement to copy data into an existing replicated table.

**Note:** Using bulk copy bypasses the Change Recorder, so rows added to a table with bulk copy are not replicated. Typically, this is not a problem because replicated tables have a unique primary or secondary index, and bulk copy does not occur if an index exists.


The Import Assistant also uses the Ingres copy statement to copy data into a table. Hence, the above statement is valid for this application, and rows imported through the Import Assistant cannot be replicated.

For more information, see the section Bulk Copying in the *SQL Reference Guide*.

**Visual DBA:** For step-by-step procedures, see the online help topic, Copying Database Objects. 

**Replicator Manager and Command Line:** If bulk copying is used to copy data into a replicated table, use the CreateKeys operation (BothQueue&ShadowTable menu item) to propagate the new data that bypassed normal data capture.

**Note:** If the table was not empty, this procedure causes collisions.

For more information, see Creation of Replication Keys (see page 91) and CreateKeys Option (see page 124). 

## Using SQL Statements on Replicated Tables

Certain SQL statements can have unexpected effects on replicated tables. Be aware of the following statements when maintaining the replication system:

- modify to truncated
- create table
- alter table
- create integrity



## Modify to Truncated Statement

The SQL statement "modify to truncated" bypasses change recording and, therefore, must not be used with replicated tables. If you are deleting in bulk from a replicated table and the deletion of records is expected to be replicated throughout the replication system, use the delete statement instead of the modify to truncated statement.

Use a modify to truncated statement if you are completely cleaning out a table. In this case, issue modify to truncated on the table to be cleared in every database in the replicated system. It is also recommended that you run the Arcclean utility at this time.

## Integrity Constraints

Ingres constraints, which are defined using the create table and alter table statements, must be identical among replicated databases. If the statements are not identical, a Replicator Server transmission error can result. The error occurs when a server is attempting to replicate data into a target database and finds that this violates the constraints of the target database.

In addition, keep in mind that referential integrity constraints cause irresolvable situations if rows related by the constraint are updated on different databases. For example, if a user deletes a customer information row in one database, while another user adds a purchase order for that customer in a second database, and there is a constraint on the purchase order table to the customer table primary key, the purchase order cannot be propagated to the first database. Because this is not a conflict between the same row in the two databases, automatic collision resolution cannot correct this.

The create integrity statement, which is used to define integrity constraints, must also be identical among replicated databases.

## Reconfiguration of the Replicator

To reconfigure Ingres Replicator, you add new databases to the configuration and change the table schema. The procedures are basically the same as those you use to configure Ingres Replicator.

## How You Add a Database to the Replicated Configuration

To add a database to an existing replicated configuration, follow these steps.


### Visual DBA:

1. Stabilize the replication environment. For more information, see *How You Stabilize the Replication System* (see page 175).
2. Create and populate the new database.
3. Access the main database and add the new database, CDDS, and propagation path information.

Detailed steps for performing these procedures can be found in the following online help topics:

- Installing Replication Objects
- Creating a CDDS
- Adding Replication Databases
- Choosing Propagation Paths

For additional information, see the same topics as above in the chapter "Using Visual DBA for Configuration."


4. Move the replication configuration from the main database to the new database using the Propagate dialog. For more information, see the online help topic Propagate dialog. 

### Command Line and Replicator Manager:

1. Stabilize the replication environment. For more information, see *How You Stabilize the Replication System* (see page 175).
2. Create and populate the new database.
3. Access the Replicator Manager from the main database and add the new database, CDDS, and propagation path information.

For more information, see *Database Summary* (see page 108), *CDDS Definition Dialog* (see page 88).

4. Move the replication configuration from the main database to the new database with the MoveConfig command.

For more information, see *Move Configuration Data Window* (see page 126). 

## How You Copy Replicated Tables Between Replicated Databases

Copying a table between replicated databases is done for several reasons. For example, a table already exists in both databases and is being added to the replication scheme, in which case, it is important to synchronize; or a table exists in only one database and the CDDS is being extended to encompass a new or existing database.

### Command Line and Replicator Manager:

To copy replicated tables to other replicated databases, follow these steps:

1. Copy the base replicated table to the new database using the copydb command. (Do not copy shadow and archive tables at this time.) For example:  

```
copydb dbname tablename
```
2. Execute the copy.out file against the old database and the copy.in command against the new database. For example:  

```
sql old_dbname < copy.out  
sql new_dbname < copy.in
```
3. Access Replicator Manager on the old database to register the table in the new database.
4. Select Configuration from the Replicator Manager for the old database.
5. Select MoveConfig from the Configuration menu.
6. Place the cursor on the new database and select ThisDatabase, from the Move Configuration window, .
7. To confirm that the new table was registered correctly, access Replicator Manager on the new database.
8. From the Replicator Manager for the new database, select Configuration.
9. From the Configuration menu, select Tables.
10. From the Table Registration Summary window, place the cursor on the new table and select Edit. Determine the name of the shadow table and, if necessary, the archive table using the SQL Terminal Monitor.

**Note:** The archive table is needed if the base table is part of a Full Peer CDDS at the target database.

11. Use the copydb command to copy the shadow and archive tables from the old database.

For example:

```
copydb old_dbname shadow_name [archive_name]  
sql old_dbname < copy.out
```

12. Edit the copy.in script and remove any create table, modify, and create index statements. Copy the shadow and archive tables into the new database.

For example:

```
sql new_dbname < copy.in
```

## How You Change the Table Schema

It is possible to change the schema of any table in your replication system. Schema changes must be made on every target database where that table exists.

**Visual DBA:** For step-by-step procedures, see the online help topic Changing the Table Schema.

### Command Line and Replicator Manager:

To change the schema of any table in your replication system, follow these steps:

1. Access Replicator Manager against the database containing the table to be changed.
2. Deregister the table from the Table Registration Summary window of the Replicator Manager using the Deregister option. For more information, see Deregister Tables (see page 118).

**Note:** Deregistering the table causes existing data in the shadow and archive tables to be lost because deregister drops them. If large tables are being reconfigured to expand the width of a column, for example, it is recommended that you copy out the shadow and archive tables before deregistering and reload them later.

3. Use one or more alter table statements to effect the schema changes desired. Tables need to have 4 KB or larger page size.
4. In the Replicator Manager, use the Tables option to perform the following registration tasks on the table:
  - a. Register the columns.
  - b. Create the support objects.

If required, create the replicated transaction keys (ShadowTableOnly).

For more information, see Table Registration Summary (see page 117).

5. Use the MoveConfig option to move the new database information to other affected databases in your replication system.

For more information, see Move Configuration Data Window (see page 126).

6. Activate change recording.

For more information, see Activate Change Recording Window (see page 128).

After all databases reflect the change in the table schema, the table is ready for replication on your system. ▣

## Optimizing Replicator Catalogs and Support Tables

Once the initial configuration has been defined, or following any significant changes, consider optimizing the Ingres Replicator system catalogs.

**Visual DBA:** You purge records from shadow and archive tables by expanding the Databases branch in the Database Object Manager, expanding the branch of the desired database, and highlighting Replication. Choose the Arcclean command from the Operations menu. For more information, see the online help topic Arcclean Operation dialog.

You modify the replicated system database by expanding the Databases branch in the Database Object Manager, expanding the branch of the desired database, and highlighting Replication. Choose the Repmod command from the Operations menu. For more information, see the online help topic Repmod Operation dialog box.

**Command Line:** Use the Arcclean command to purge unneeded records from the shadow and archive tables. See the *Command Reference Guide* for detailed information and procedures.

Use the Repmod command to modify the replicated system database to predetermined storage structures. For more information, see the *Command Reference Guide*.

**Note:** Arcclean remodifies the shadow and archive tables that it cleans up.

## Collision Resolution

A collision in a replicated database occurs when changes are made to the same record in different databases participating in the same CDDS. In a collision condition, Ingres Replicator cannot synchronize the databases without destroying information. The Replicator Servers can detect this collision condition when the data is transmitted between the two databases. For more information, see Collision Design (see page 52).

### Collision Detection

Suspected collisions in the Ingres Replicator system can be detected from these sources:

- Errors in the Replicator server log file (replicat.log)
- Collision Report
- E-mail from the Replicator Server

## Collisions in the Replication Server Log File

Because of the overhead involved, the Replicator Server by default does not look for collisions or attempt to resolve them. You can control collision detection and resolution through CDDS collision modes. If a collision does occur, the Replicator Server identifies the condition as an error. For example, the server issues errors such as:

The target record cannot be found in the target database or

or

A primary key was already in the target database for an insert transaction.

If you find these types of errors in the replicat.log file, this indicates a collision condition.

## Queue Collision Report

If you suspect that a collision has occurred, run the Queue Collision Report available through Visual DBA or the Resolver option of Replicator Manager. This option looks at the distribution queue of the source database to see if any collision conditions exist. The report displays problems that exist in the current distribution queue and its target databases. For more information, see Viewing Collision Conflicts in online help or Queue Collision Report (see page 188).

## Methods to Handle Collisions

There are two ways to handle collisions—automatic or manual. Each method has advantages and disadvantages. Automatic resolution (using the Ingres Replicator collision mode settings) takes less time but can produce unexpected results. Manual resolution gives you more control and, in some cases, is the only way to resolve a conflict.

To resolve the condition, you must restore consistency to the base tables of the source and target database as well as to the replicated transaction keys in the shadow tables. Replication operations (insert, update, delete) that caused the collision must also be removed from the distribution queue.

## How You Resolve Collisions Manually

You can resolve collisions manually in the following ways:

**Visual DBA:** You can resolve collisions manually by expanding the Replication branch and expanding the branch of the desired database and clicking the Collisions tab. For more information, see the online help topic Resolving Collisions Manually. ■

**Command Line:** To resolve a collision manually, follow these steps:

1. Determine the correct image of the record in collision that must be in the replicated system.

2. Determine the replicated transaction key for the record.

The key that you select must be the replicated transaction key of one of the records in collision.

3. To correct the record, you need to temporarily disable the Change Recorder. To do this, issue a set trace point DM32.

**Note:** This disables the Change Recorder for your session only.

4. Update the record to be the correct image.

5. Update the shadow record, giving it the selected replicated transaction key.

6. Remove any replication commands from the distribution queue that cause this record to be in collision.

7. Re-enable the Change Recorder by issuing a set notrace point DM32.

8. Repeat Steps 1-6 for each database in the replicated system that is affected by the collision. ■

## Automatic Resolution of Collisions

With automatic resolution, when two records collide, one record prevails over the other. If the operation was an insert or an update, a record survives a collision by overwriting the record in the target database (the target row is deleted and the prevailing source row is inserted in its place). If the transaction was a delete, the record in the target database is deleted. If a record does not survive a collision, its replication operation (insert, update, or delete) for that target is removed from the distribution queue.

**Note:** Automatic resolution overwrites the entire record and can overwrite columns that have correct information with columns that do not have correct information. You must not use automatic resolution if the information contained in the losing record is important. For example, if your database contains documents that are continually updated with new information, you can lose information with automatic resolution.



## Disaster Recovery

The normal disaster recovery methods that you use on standard databases also apply to replicated databases. Checkpoints, journaling, and rollforwards must be used on the Ingres Replicator system to bring a failed database to a consistent state. For a description of these recovery tools, see the *Database Administrator Guide*.

If these methods recover the failed database to the exact point of failure, the remote databases queue all the changes that are due to happen to the failed database, ready for the moment when the failed database becomes available. However, if journals, log files, or dump areas are lost, a gap of missing data exists on the failed database. You can recover part or this entire gap of missing data if the records still exist and reflect the data that was lost for the duration of the gap. For more information, see *Recovering Transactions from a Peer Database Using the Reconciler* (see page 185).

### Recovering Transactions from a Peer Database Using the Reconciler

In the event of a system failure in Ingres, it is standard procedure to recover the database from checkpoints. If journals, log files, or dump areas are lost, a gap of data on the failed database occurs. This same gap can still exist on one of the replicated databases. Use the Reconciler to attempt to recover the data gap using the shadow and archive tables for one or more peer databases.

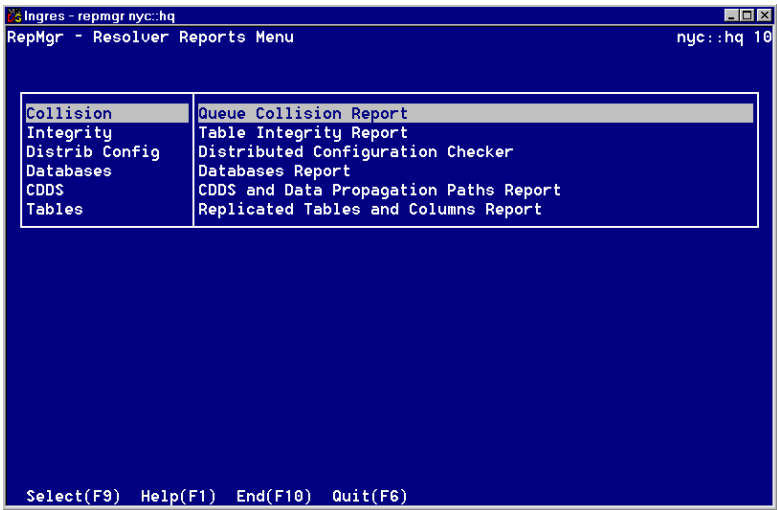
**Visual DBA:** You perform the reconcile procedure by choosing the Reconciler command from the Operations menu. For more information, see the online help topic *Reconciling a Replicated Database*. ■

**Command Line:** Use the `reconcil` command to recover the gap of data on the failed database using the shadow and archive tables from one or more full peer databases. For more information, see the *Command Reference Guide*. ■

## Resolver Reports Menu (Replicator Manager)

The Resolver Reports Menu appears when you select the Resolver option on the Replicator Manager main menu. From this window you can choose reports, which are useful as tools to maintain the consistency and integrity of your replicated databases.

**Note:** For the equivalent functionality using Visual DBA, see Replication Monitoring Using Visual Performance Monitor or Visual DBA (see page 156).



The Resolver Reports Menu has the following options:

**Collision**

Provides a list of collision conflicts between the local database and all the targets that have pending transactions in the distribution queue.

**Integrity**

Provides a comparison of a replicated table between the local database and another database.

**Distrib Config**

Provides a Distributed Configuration Checker Report that details any significant differences between the local database and remote database configurations. These database differences include registered tables and columns, database number, name, node, and so on.

**Databases**

Provides a listing of the replicated databases defined in the local database.

**CDDS**

Provides a listing of all CDDSs defined in the local database and their propagation paths.

**Tables**

Provides registration status of the tables and columns defined in the local database.

## Queue Collision Report

The Queue Collision Report lists collision conflicts that exist on the local database. By reviewing the report, you can decide if you want to manually or automatically resolve each collision.

The report lists all collisions found from the distribution queue. It reports the type of collision (insert, update, delete) and the values of the columns in the local database, followed by the remote database where the collision occurred (number, node, name) and the values of the columns in the remote database. The end of the report displays the total number of collisions found.

Following is an example of the Queue Collision Report:

```
Ingres Replicator
Queue Collision Report
Local Database: 10 hq

INSERT Collision for table rep_dba.book_list
Local database
Source DB: 10 Transaction ID: 811660429 Sequence No: 1
  *name: Doe, John
    manager: Jones, Ashley
    hourly_rate: 23.000
    title: Programmer
Remote Database Number 20 Node: lon, Name: europe
  *name: Doe, John
    manager: Wolfe, Neal
    hourly_rate: 25.000
    title: Programmer

INSERT Collision for table rep_dba.book_list
Local database
Source DB: 10 Transaction ID: 811660837 Sequence No: 1
  *name: Doe, Jane
    manager: Wolfe, Neal
    hourly_rate: 47.000
    title: Sr Programmer
Remote Database Number 20 Node: lon, Name: europe
  *name: Doe, Jane
    manager: Ashley, Jones
    hourly_rate: 45.000
    title: Sr Programmer

2 collision(s) found.
```

## How the Queue Collision Report Is Created

When you select the Collision option from the Resolver Reports Menu window, and there are records in the distribution queue, the Collision Report is created.

When you execute the report, the distribution queue of the local database is read and the following searches are conducted to determine if there are any collisions:

- For insert operations, the remote base table is searched using the table key. If found, both records are displayed in the report.
- For update operations, the remote shadow table is searched using the replicated transaction key of the previous operation that manipulated the row. If not found, the collision is displayed with the local replicated transaction key. If the table key exists in the remote table, both records are displayed.
- For delete operations, the remote shadow table is searched using the replicated transaction key of the previous operation that manipulated the row. If not found, the collision is displayed with the local replicated transaction key.

## Table Integrity Report Window

The Table Integrity Report window appears when you select the Integrity option from the Resolver Reports Menu window. This window allows you to define a report that compares a table in the local database against the same table in a different database.

Compare contents of table:  
for owner:  
in Consistent Distributed Data Set (optional):  
in Database Number 1:  
and Database number 2:  
Transaction Date/Times (Optional):  
Begin:  
End:

Order by:

Column Name	Order

Go(F9) ListChoices(SH-F2) Help(F1) End(F10)

You specify the criteria for the comparison based on transaction begin and end time, and if desired, a CDDS value to designate a particular horizontal partition. You can also designate the sort order of the report by replicated column or transaction time.

The Table Integrity Report window has the following items:

**Compare contents of table**

Specifies the replicated table for the first and second database that you want to compare.

**for owner**

Identifies the owner of the table. If only one user owns a table by that name, this field is automatically filled in. For tables with multiple owners, a pop-up table appears, which lets you select the appropriate owner.

**in Consistent Distributed Data Set (optional)**

(Optional) Identifies the value of the CDDS number to be reported. If the table is not horizontally partitioned, this field is automatically filled in with the CDDS number under which the table is registered.

For horizontally partitioned tables, enter a CDDS number from the CDDS lookup table. You can use ListChoices to view a list of acceptable values.

**in Database Number 1**

Identifies the number of the local database. This field is automatically filled in.

**and Database Number 2**

Identifies the number of the other database that you want to compare with the local database.

**Begin**

(Optional) Specifies the beginning of a date range in standard Ingres date and time format.

**End**

(Optional) Specifies the end of a date range in standard Ingres date and time format.

**Column Name**

Displays the names of all registered columns for the specified table. A trans\_time column also appears representing the transaction time.

**Order**

Lets you display rows in a specific order. Enter values (in numerical sequence starting with 1) beside each column name.

**Go**

Executes the report.

## How the Table Integrity Report Is Created

When you execute the report, the Integrity option checks whether the table structure or a secondary index force the table keys to be unique. If the keys are not unique, this information appears in the report. The Integrity option compares the shadow tables of the selected table and reports any rows that are present in one but not in the other.

This comparison is performed on the entire row. If there is a row that differs in only one column, it appears in the report as having one set of data missing from the second database, and the slightly different data is reported as missing from the local database. The report shows which database holds the data and all of the columns and values.

The following is an example of the Table Integrity Report:

Ingres Replicator

Table Integrity Report

For table 'rep\_dba.emp' in 'nyc::hq'

And table 'euro\_dba.emp' in 'lon::europe'

Row only in 'nyc::hq', not in 'lon::europe'

database no: 10, transaction id: 811660429, sequence no: 1.

```
*name: Doe, John
manager: Jones, Ashley
hourly_rate: 23.000
title: Programmer
```

Row only in 'nyc::hq', not in 'lon::europe'

database no: 10, transaction id: 811660837, sequence no: 1.

```
*name: Doe, Jane
manager: Wolfe, Neal
hourly_rate: 47.000
title: Sr Programmer
```

Row only in 'lon::europe', not in 'nyc::hq'

database no: 20, transaction id: 811660329, sequence no: 1.

```
*name: Doe, John
manager: Wolfe, Neal
hourly_rate: 25.000
title: Programmer
```

Row only in 'lon::europe', not in 'nyc::hq'

database no: 20, transaction id: 811660780, sequence no: 1.

```
*name: Doe, Jane
manager: Ashley, Jones
hourly_rate: 45.000
title: Sr Programmer
```

## Distributed Configuration Checker Report

The Distributed Configuration Checker Report verifies that the configuration information in a local database and its remote counterparts is identical where necessary. The report pinpoints any significant differences between the database objects and the configurations of those objects.

We recommend that you run this report after installing or reconfiguring Ingres Replicator as a way to ensure that you have no configuration errors.

The Distributed Configuration Checker Report checks the following items:

- Replicated database information
- CDDS information
- Propagation path information
- All tables registered for replication
- All registered columns

The report is created when you choose the DistribConfig option from the Resolver Reports Menu window.

Following is an example of the Distributed Configuration Checker Report:

```
Ingres Replicator
Distributed Configuration Checker Report
From local database 'hq'
28-feb-2003 09:24:00
Report on remote database sfo::west (11):
  DD_CDDS ERROR:
    Record for CDDS no 3, CDDS name 'Accounting'
    not found in remote database.
  DD_REGIST_TABLES ERROR:
    Support tables not created for local table no 4, table name 'tab1',
    table owner 'testenv'.
Report on remote database lon::europe (20):
  DD_CDDS ERROR:
    CDDS no 0 has local collision mode of 0,
    but remote collision mode of 2.
  DD_CDDS ERROR:
    Record for CDDS no 3, CDDS name 'Accounting'
    not found in remote database.
  DD_PATHS_ERROR:
    Record for CDDS no 0, localdb no 10, sourcedb no 10,
    targetdb no 11 not found in remote database.
  DD_PATHS_ERROR:
    Record for CDDS no 0, localdb no 10, sourcedb no 20,
    targetdb no 11 not found in remote database.
  DD_PATHS_ERROR:
    Record for CDDS no 0, localdb no 10, sourcedb no 30,
    targetdb no 11 not found in remote database.
  DD_REGIST_TABLES ERROR:
```



```
      Support tables not created for local table no 4, table name 'tab1',
      table owner 'testenv'.
Report on remote database hkg::asia (30):
DD_CDDS ERROR:
      CDDS no 0 has local collision mode of 0,
      but remote collision mode of 2.
DD_CDDS ERROR:
      Record for CDDS no 3, CDDS name 'Accounting'
      not found in remote database.
DD_REGIST_TABLES ERROR:
      Support tables not created for local table no 4, table name 'tab1',
      table owner 'testenv'.
```

## Replicated Databases Report

The Replicated Databases Report lists information about all replicated databases that are defined in the local database. It displays the Database Number, Virtual Node Name, Database Name, DBMS Type, Database Owner, and Remarks.

This report is equivalent to the contents of the Database Summary window (see page 108).

The report is created when you choose the Databases option from the Resolver Reports Menu window.

Following is an example of the Replicated Databases Report:

```

28-Feb-2003                                09:40:52

                                Ingres Replicator
                                Replicated Databases Report

hq 10

-----
Database # Virtual Node Name / Database Name          DBMS Type
          Database Owner / Remarks
-----
    10 nyc::hq
       rep_dba
       Headquarters database                          ingres

    11 sfo::west
       rep_dba
       U.S. Western Region database                  ingres

    12 dal::central
       rep_dba
       U.S. Central Region database                  ingres

    20 lon::europe
       euro_dba
       European Region database                      ingres

    30 hkg::asia
       rep_dba
       Asian Region database                          ingres

```

## CDDS Data Propagation Paths Report

The CDDS Data Propagation Paths Report displays a listing of the pathways for each CDDS. It displays the CDDS Number, Originating Database, Local Database, and Target Database.

This report is equivalent to the contents of the Propagation Path Definition window.

The report is created when you choose the CDDS option from the Resolver Reports Menu window.

Following is an example of the CDDS Data Propagation Paths Report:

```

28-Feb-2003                                     10:04:37
                                     Ingres Replicator
hq 10                                     CDDS Data Propagation Paths Report
-----
CDDS No:      1 Inventory
Originating Database
      Local Database
      Target Database
-----
10  nyc::hq
   10  nyc::hq
   11  sfo::west

10  nyc::hq
   10  nyc::hq

10  nyc::hq
   20  lon::europe
   30  hkg::asia

20  lon::europe
   10  nyc::hq
   11  sfo::west

20  lon::europe
   20  lon::europe
   10  nyc::hq

20  lon::europe
   20  lon::europe
   30  hkg::asia

30  hkg::asia
   10  nyc::hq
   20  lon::europe
   11  sfo::west

30  hkg::asia
   20  lon::europe

```

```

      10    nyc::hq
30    hkg::asia
      30    hkg::asia
      20    lon::europe
```

## Registered Tables Report

The Registered Table Report displays the status of all replicated tables. This report is created when you select Tables from the Resolver Reports Menu window.

Following is an example of the Registered Tables Report:

28-Feb-2003

11:35:16

### Ingres Replicator Registered Tables Report

hq 10

Table	C S A	Priority Lookup Table			
CDDS		CDDS Lookup Table			
Column	Data type		D	K	R
book_list	C S A				
1 Inventory					
book_no	integer not null with default		1	K	R
title	varchar(40) not null with default				R
author	varchar(25) not null with default				R
price	money not null with default				R
category	varchar(12) not null with default				R
stock	smallint not null with default				R
dist_no	smallint not null with default				R
book_orders	C S A				
0 Default CDDS					
order_no	integer not null with default		1	K	R
book_no	integer not null with default		2	K	R
sale_price	money with null				R
quantity	smallint with null				R
extension	money with null				R
cust_info	C S A				
0 Default CDDS					
cust_no	integer not null with default		1	K	R
name	varchar(25) not null with default				R
company	varchar(25) with null				R
street	varchar(20) not null with default				R
city	varchar(20) not null with default				R
state	char(2) not null with default				R
zip	char(10) not null with default				R
card_no	varchar(30) not null with default				R
ep_date	date not null with default				R
ship_to	char(1) not null with default				R
cust_orders	C S A				
0 Default CDDS					
order_no	integer not null with default		1	K	R
cust_no	integer not null with default				R
order_date	date not null with default				R
status	varchar(15) not null with default				R
order_total	money with null				R

C - Columns Registered  
S - Support Objects Created  
A - Change Recording Activated

D - Table Key Sequence  
K - Replicated Key Column  
R - Replicated Column

The Registered Tables Report has the following items:

**Table**

The name of the table.

**CDDS**

The number and name of the CDDS to which the table belongs.

**Column**

The name of the column.

**C (Columns Registered)**

Indicates that the columns are registered.

**S (Support Objects Created)**

Indicates that support objects are created.

**A (Change Recording Activated)**

Indicates that change recording has been activated.

**Data type**

The data type and declared length of the column.

**Priority Lookup Table**

The name of the priority lookup table assigned to the table, if defined.

**CDDS Lookup Table**

The name of the horizontal partitioning lookup table assigned to the table, if defined.

**D (Table Key Sequence)**

A number indicates that the column is a key column in the base table structure and indicates the column's sequence in the key.

**K (Replicated Key Column)**

Indicates that the column is a key column for replication.

**R (Replicated Column )**

Indicates that the column is registered for replication.

## How to Deactivate Replication

There are two ways to disable replication:

- Deactivate the Change Recorder for a CDDS or database.

This stops the recording of changes, but the configuration and the replicator objects remain in the database.

For details, see Deactivate the Change Recorder for a CDDS or Database (see page 199).

- Remove replication objects.

This removes the configuration and all the replicator objects from the database.

For details, see Removing Replication Objects (see page 200).

### Deactivate the Change Recorder for a CDDS or Database

You can deactivate individual tables, CDDSs, or databases from Change Recording. Doing so stops the recording of changes, but the configuration and the replicator objects remain in the database.

**Note:** Typically, it is not necessary or advisable to activate or deactivate individual tables. Rather, you would perform the operation on a CDDS or database.

#### To deactivate an object using Replicator Manager

1. Choose the Activate option from the Configuration Menu.

The Activate Change Recording window is displayed.

2. Place the cursor on the object to be deactivated, and choose a command from the menu line.

The menu line changes to:

Activate   DeActivate   Cancel

3. Choose DeActivate.

The selected object is deactivated.

The Stat column appears at the far right side of the Activate Change Recording window with the word "done" next to the items you deactivated.

#### To deactivate an object using VDBA

Refer to the VDBA online help topic Deactivating Replication.

## Removing Replication Objects

If replication is no longer needed on a database, you can remove the Ingres Replicator database objects from a replicated database. Doing so removes all the replicator catalogs (including the configuration) as well as the shadow and archive tables.

**Note:** Before removing replicator objects, you should disable the Change Recorder for the database.

**Visual DBA:** You remove replication objects by choosing the Dereplicate command from the Operations menu. For more information, see the VDBA online help topic Removing Replication Objects.

**Command Line:** Use the dereplic command to remove Ingres Replicator objects. For more information, see the *Command Reference Guide*.



# Chapter 8: Using Advanced Features

---

This section contains the following topics:

[Lookup Tables](#) (see page 201)

[Strategies for Avoiding Deadlock](#) (see page 210)

## Lookup Tables

You can create lookup tables to establish:

- Horizontal partitioning of a CDDS
- Priority collision resolution

### How You Set Up Horizontal Partitioning

A horizontally partitioned table uses a base CDDS, which contains the whole replicated base table, and partitioned CDDSs that each contain only those rows that are associated with the particular value assigned to that CDDS in the lookup table.

To set up horizontal partitioning, you must complete two main tasks:

- Create and populate the lookup table in every participating database.
- Implement horizontal partitioning through Visual DBA or the Replicator Manager.

## How You Create a Horizontal Partitioning Lookup Table

To set up horizontal partitioning, you must create and populate the lookup table in every participating database.

To create and populate a horizontal partitioning lookup table, follow these steps:

1. Create the horizontal partitioning lookup table with two types of columns:
  - Columns with the same name and data type as the columns in the base table that determines the partitioning. You can have one column of this sort, for example, a location column; or you can have several columns, for example, group code, division code, and region code.
  - A column called `cdds_no`, with a data type of `smallint` not null, that contains the CDDS number that corresponds to the partitioning value
2. Populate the horizontal partitioning lookup table using Visual DBA's SQL Scratchpad or SQL Assistant, the SQL Terminal Monitor, or Query-By-Forms; enter the partitioning values and their corresponding CDDS numbers. You do not need to add a row for the base CDDS if the base values do not need to be replicated elsewhere.
3. Once the table is populated in one database, use `copydb` to copy the lookup table to the other targets. Each lookup table must be exactly the same in all databases.

Alternatively, repeat Steps 1 and 2 for every database that participates in the horizontal replication.

**Note:** If you do not have a populated lookup table in every database that participates in horizontal partitioning, horizontal partitioning does not work properly.

## How You Implement Horizontal Partitioning Using Visual DBA

After you create and populate the lookup table in every participating database, you must implement horizontal partitioning using one of the appropriate Ingres tools.

To implement horizontal partitioning using Visual DBA, follow these steps:

1. Define all the databases that participate in the partitioning by highlighting Databases in the Database Object Manager and clicking the Add Object toolbar button or choosing the Create command from the Edit menu. For more information, see the online help topic *Creating a Database*.
2. Add new CDDs for horizontal partitioning with a different CDD for every partition. Expand the Replication branch and highlight CDD. For more information, see the online help topic *CDD Definition dialog box*. Add propagation paths for each new CDD in the expanded CDD branch (of the expanded Replication branch) by selecting the desired CDD number. For more information, see the online help topic *CDD Definition dialog box*.

If the base CDD is present only in a single database, it does not need any propagation paths.

3. Add database and Replicator server information for each new CDD in the expanded Replication branch, expanding the CDD branch, and selecting the desired CDD number. For more information, see the online help topic *CDD Definition dialog box*. Register the base table to be partitioned by clicking the check box next to its name in the CDD Definition window. For more information, see the online help topic *CDD Definition dialog box*.

The lookup table appears in the Tables list because it is a table in the database. However, it need not be registered for replication.

**Note:** The lookup tables are expected to be static; ordinarily they are not replicated. If desired, they can be registered. However, keep in mind that changes to the lookup table must be propagated ahead of rows in the base table or the partitioning scheme breaks down.

4. Attach the horizontal partitioning lookup table to the base table by specifying the Lookup Table in the CDD Definition dialog. For more information, see the online help topic *CDD Definition dialog box*.
5. Continue with configuration procedures—creating support objects, moving the configuration to other databases, and activating change recording—as explained in the chapter “Using Visual DBA for Configuration.” ■

## How You Implement Horizontal Partitioning Using Replicator Manager

After you create and populate the lookup table in every participating database, you must implement horizontal partitioning using one of the appropriate Ingres tools.

To implement horizontal partitioning using Replicator Manager, follow these steps:

1. Define all the databases that participate in the partitioning in the Database Summary window.

For more information, see [Add a Database to the Database Summary List](#) (see page 109).

2. Add new CDDSs for horizontal partitioning in the CDDS Summary window, with a different CDDS for every partition.

For more information, see [Add a CDDS](#) (see page 111).

3. Add database and Replicator server information for each new CDDS in the CDDS Databases and Servers window.

For more information, see [Add Database and Server Information](#) (see page 113).

4. Add propagation paths for each new CDDS in the Propagation Path Definition window. If the base CDDS is present in only a single database, it does not need any propagation paths.

For more information, see [Add Propagation Paths](#) (see page 115).

5. Register the base table to be partitioned from the Table Registration Summary window.

For more information, see [Register Tables](#) (see page 118).

The lookup table also appears in the Table Registration Summary window because it is a table in the database. However, it need not be registered for replication.

**Note:** The lookup tables are expected to be static; ordinarily they are not replicated. If desired, they can be registered. However, keep in mind that changes to the lookup table must be propagated ahead of rows in the base table or the partitioning scheme breaks down.

6. Choose Edit from the Table Registration Summary window, and enter the new CDDS number for the base table in the Table Registration Details window.

For instructions, see [Assign a Table to a Different CDDS](#) (see page 120).

7. Specify the horizontal partitioning lookup table to the base table in the Table Registration Details window.

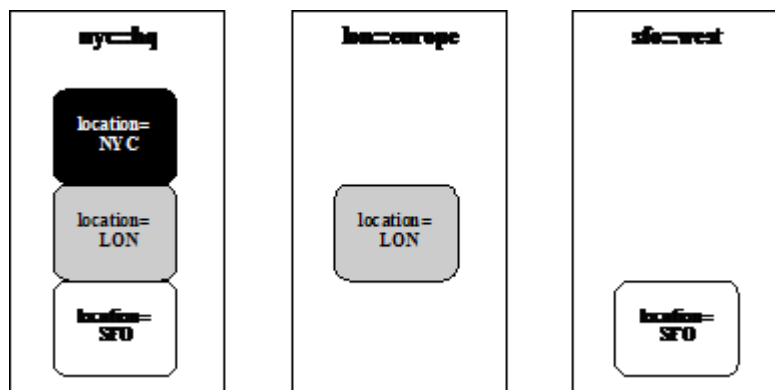
For more information, see [Assign Lookup Tables](#) (see page 121).

8. Continue with configuration procedures as explained in the chapter "Configuring Replication Using Replicator Manager."

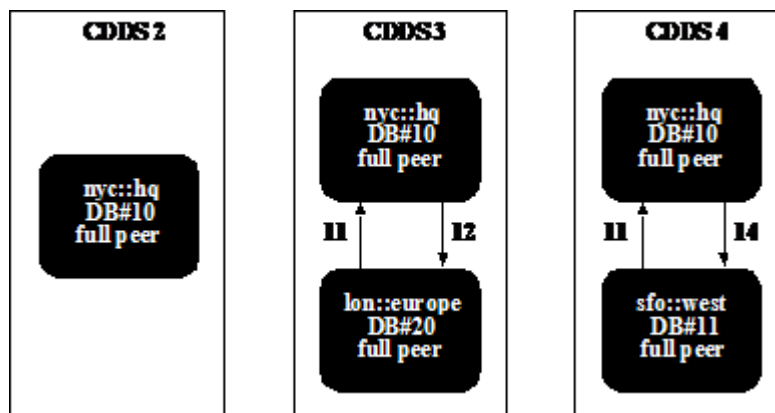
### Example: R.E.P.'s Horizontal Partitioning

The R.E.P. company is partitioning its warehouse information: the New York location must contain information for all warehouses, and the San Francisco and London locations must only contain information for their respective sites. This example is also discussed in CDDS Example: Horizontal Partitioning (see page 34).

The whse\_inventory table contains the location column used to determine horizontal partitioning. The new CDDSs and their contents are illustrated in the following figure:



The CDDS Diagrams portion of the CDDS Worksheet is illustrated in the following figure:



The R.E.P. DBA creates, populates, and distributes the lookup table `whse_inventory_cdds`.

1. Create the table by entering the following SQL statement with SQL Scratchpad in Visual DBA, or enter SQL/ISQL from the command line:

```
create table whse_inventory_cdds (  
    location    char(3)    not null,  
    cdds_no     smallint   not null)
```

The location column contains the location values that are used to partition the data.

2. The DBA populates `whse_inventory_cdds` with the following values:

location	cdds_no
LON	3
SFO	4

No value is added for NYC because the New York data is not replicated to other sites.

3. The DBA copies `whse_inventory_cdds` to the `lon::europe` and `sfo::west` databases.

The R.E.P. DBA implements horizontal partitioning by completing the following tasks on `nyc::hq`:

1. The DBA does not need to add any database information because all three databases participate in other CDDSs that were already configured.
2. The DBA adds the following CDDSs to the CDDS branch under Replication in Visual DBA or to the CDDS Summary window in Replicator Manager:
  - CDDS 2, NYC Warehouse
  - CDDS 3, LON Warehouse
  - CDDS 4, SFO Warehouse

The DBA uses the default collision and error modes.

3. The DBA adds database and Replicator Server information for each CDDS in the CDDS Definition window in Visual DBA or in the CDDS Databases and Server window in Replicator Manager.

**Note:** If you are using Visual DBA, Step 4 is performed before Step 3.

CDDS 2 does not need any propagation paths because the NYC data is not replicated to any other locations.

The Database Summary portion of the CDDS Worksheet for each CDDS is as follows:

CDDS No.	Database No./Name	Target Type	Server Number
2 (NYC)	10, nyc::hq	full peer	11
3 (LON)	10, nyc::hq	full peer	11
	20, lon::europe	full peer	12
4 (SFO)	10, nyc::hq	full peer	
	11, sfo::west	full peer	

- The DBA adds the propagation paths for each CDDS in the CDDS Definition dialog in Visual DBA or the Propagation Path Definition window in Replicator Manager.

The Propagation Paths portion of the CDDS Worksheet for CDDS 3 is as follows:

Originator	Local	Target	Comment
10 (nyc::hq)	10 (nyc::hq)	20 (lon::europe)	NYC to LON
20 (lon::europe)	20 (lon::europe)	10 (nyc::hq)	LON to NYC

The Propagation Paths portion of the CDDS Worksheet for CDDS 4 is as follows:

Originator	Local	Target	Comment
10 (nyc::hq)	10 (nyc::hq)	11 (sfo::west)	NYC to SFO
11 (sfo::west)	11 (sfo::west)	10 (nyc::hq)	SFO to NYC

- The DBA registers the whse\_inventory table in the CDDS Definition window in Visual DBA or in the Table Registration Summary window in Replicator Manager. In Replicator Manager, at this point, the CDDS assigned to whse\_inventory is still the Default CDDS.
- The DBA accesses the registration information for whse\_inventory in the Table Registration Details window and changes the CDDS number to 2.
- The DBA specifies the whse\_inventory\_cdds lookup table:

**Replicator Manager:** The DBA specifies the whse\_inventory\_cdds lookup table to whse\_inventory in the Table Registration Details window by choosing the ListChoices option while the cursor is in the Horizontal Partition Lookup field.

**Visual DBA:** The DBA specifies the whse\_inventory\_cdds lookup table to whse\_inventory in the CDDS Definition window by typing the lookup table name into the CDDS Lookup field.

- Continue with configuration procedures—creating support objects, moving the configuration to other databases, and activating change recording—as explained in the chapters “Configuring Replication Using Visual DBA” and “Configuring Replication Using Replicator Manager.”

## Priority Collision Resolution

In CDDs that use PriorityResolution collision mode, collisions are resolved by comparing the database numbers of the colliding databases; the database with the lowest number prevails. However, you can alter the priority of individual rows by creating and assigning a priority collision resolution lookup table that contains priority numbers; the row with the highest number prevails.

### How You Create a Priority Collision Resolution Lookup Table

A priority collision resolution lookup table allows you to specify priority according to data values in a registered table. Follow these steps to create this table based on data values:

1. Create the priority collision resolution lookup table with two types of columns:
  - Columns with the same name and data type as the columns in the base table that determines priority. You can have one column of this sort, for example, a location column; or you can have several columns, for example, group code, division code, and region code.
  - A column called `dd_priority`, with a data type of `smallint` not null, that contains the priority number; `dd_priority` must be an integer with the range 0-32767.

**Note:** If many levels of priority are required, it is useful to establish a primary key on those lookup columns.

2. Populate the priority lookup table, assigning a priority number to each value so that the record with the highest priority has the largest priority number.
3. Once the table is populated in one database, use `copydb` to copy the lookup table to the other targets. Every database that contains the base table that is the basis for priority collision resolution must include the lookup table.

Alternatively, repeat the preceding steps for every database that contains the CDDs to have priority collision resolution.

4. Register the base table for priority collision resolution in the CDDs Definition window in Visual DBA or from the Table Registration Summary window in the Replicator Manager.

The lookup table appears in the Tables list in Visual DBA or in the Table Registration Summary window in Replicator Manager because it is a table in the database. However, it need not be registered for replication because it is not replicated.



5. Specify the priority collision resolution lookup table:

**Visual DBA:** Specify the priority collision resolution lookup table to the base table in the CDDS Definition dialog. For more information, see the online help topics, Altering a CDDS and CDDS Definition dialog box.

**Replicator Manager:** Specify the priority collision resolution lookup table to the base table in the Table Registration Detail window. For instructions, see Assign Lookup Tables (see page 121).

### Example: R.E.P.'s Priority Collision Resolution

The R.E.P. company wants to use PriorityResolution collision mode on their Inventory CDDS. The Inventory CDDS can be updated from each of the three full peer databases: New York (database 10), San Francisco (database 11), and London (database 20). Without a priority collision resolution lookup table, New York has priority over San Francisco and London, and San Francisco has priority over London. R.E.P. decides, however, that London must have a higher priority than San Francisco.

The Inventory CDDS has a table called `book_inventory`, which has a `prim_whse_location` column that indicates location of the inventory item and, presumably, the database from which its row originated.

The R.E.P. DBA implements priority collision resolution by completing the following tasks:

1. The DBA creates a priority collision resolution lookup table named `book_inventory_priority` in the `nyc::hq` database.

Create the table by entering the following SQL statement with SQL Scratchpad in Visual DBA or, in Replicator Manager, with SQL/ISQL from the command line:

```
create table book_inventory_priority (
    prim_whse_location  char(3)    not null,
    dd_priority         smallint   not null)
```

2. The DBA populates the table as follows:

prim_whse_location	dd_priority
NYC	10
LON	8
SFO	5

3. The DBA copies the table to the `lon::europe` and `sfo::west` databases.
4. The DBA registers the `book_inventory` table in the `nyc::hq` database by clicking the check box in the Tables list in the CDDS Definition window in Visual DBA or in the Table Registration Summary window in Replicator Manager.

5. The DBA specifies the book\_inventory\_priority lookup table:

**Visual DBA:** The DBA specifies the book\_inventory\_priority lookup table to book\_inventory in the Priority Lookup Table field in the CDDS Definition window.

**Replicator Manager:** The DBA specifies the book\_inventory\_priority lookup table to book\_inventory in the Table Registration Details window by choosing the ListChoices option while the cursor is in the Priority Collision Resolution Lookup field.

6. Continue with configuration procedures—creating support objects, moving the configuration to other databases, and activating change recording—as explained in the chapters “Configuring Replication Using Visual DBA” and “Configuring Replication Using Replicator Manager.”

## Strategies for Avoiding Deadlock

You can use the following strategies to avoid deadlock:

- Using row locking
- Overriding default locking
- Setting an appropriate maxlocks value
- Understanding deadlock in the distribution threads

You can also use storage structures to avoid deadlock. For more information, see Storage Structures (see page 60).

## Row Locking Option

Row locking can be used to avoid deadlock in situations where a nightly modification of the shadow and archive tables is not possible. Row locking uses more resources than hash tables.

Row locking avoids lock contention on the shadow and archive tables because no users touch the same row at the same time. All operations on the base replicated table (insert, update, or delete) cause an insert to be made into the shadow and, sometimes, into the archive table. Because all these inserts add new rows and the rows added by one user do not have the same `transaction_id` value (the primary part of the key) as the rows added by other users, the locks never contend.

This is a very different situation than for page locking, where a continually increasing `transaction_id` value causes all inserts to go to the same far right B-tree leaf page. The locking characteristics of the shadow and archive tables (and inserts to the input queue table) follow the locking characteristics of the update of the base replicated table if possible.

The primary requirement for row locking is that the page size of the shadow and archive tables as well as the base table must be at least 4 KB.

**Note:** The help table statement in the SQL terminal monitor shows the page size of the relevant tables.

The base replicated table must also have a 4 KB page size, and row locking must be enabled for it. For more information, see the `SET LOCKMODE` statement. If row locking is used to update the base replicated table, the DBMS attempts to open the shadow and archive tables and the input queue table using row locking. If a page size smaller than 4 KB makes this impossible, page locking is used.

## rep\_xx\_lockmode Parameter—Override Default Locking

The default locking characteristics for the shadow, archive, and input queue tables follow those of the original update. You can override this default for every table in every database for a specific DBMS Server or for all DBMS servers.

To override the default, add the following resources to the config.dat file in the II\_CONFIG directory:

```
ii.nodename.dbms.server_name.rep_sa_lockmode
ii.nodename.dbms.server_name.rep_iq_lockmode
ii.nodename.dbms.server_name.rep_dq_lockmode
```

where:

### ***nodename***

Is the name of the node and *server\_name* is the server name (use an asterisk [\*] for all servers).

### **rep\_sa\_lockmode**

Determines the lock level used for the Ingres Replicator shadow and archive tables when a user update is performed.

### **rep\_iq\_lockmode**

Determines the lock level used for the input queue table when a user update is performed and when the distribution threads move records from the input queue.

### **rep\_dq\_lockmode**

Determines the lock level used by the Ingres Replicator distribution threads when moving records from the input queue to the distribution queue.

Possible values for these parameters are:

### **ROW**

Always use row locking.

### **PAGE**

Always use page locking.

This is the default for the rep\_dq\_lockmode parameter.

### **TABLE**

Always use table locking.

### **USER**

Locking strategy is determined by the user update (not valid for rep\_dq\_lockmode).

This is the default for the `rep_sa_lockmode` and `rep_iq_lockmode` parameters if they are not set, which means that the locking strategy is taken from the user update.

All these locking strategies describe the initial lock level at the time the update is started. If the `maxlocks` value for any table is exceeded, the locking escalates in the normal manner to table locking. For more information, see `Maxlocks Values` (see page 214). You can override the initial lock level used on the input queue table for a specified database session using `set lockmode` on `dd_input_queue` where `level = level`.

To summarize, the level set by `set lockmode` on the *tablename* statement overrides all other settings for that table. Otherwise, the level is determined by the `config.dat` parameters described above. If these parameters are not set, the level is determined by the level used when updating the base user table. Hence, the user is given the ability to determine the exact level used for each update to every table involved in replication.

**Note:** The `set lockmode` statement does not affect the updates performed by the distribution threads because they are not performed in the context of a user.

## Maxlocks Values

For user updates to the shadow, archive, and input queue tables, the value of maxlocks (in the set lockmode statement) is taken from the value inherited from the system\_maxlocks value or set with the set lockmode statement. For shadow and archive tables, this value cannot be overridden.

The input queue table is updated when any update to any base replicated table occurs. Therefore, it is important to avoid lock escalation on this table. For this table, the set lockmode statement works for maxlocks and lets you set the system\_maxlocks for the input queue separately. If set lockmode is not used on the input queue, the maxlocks value is inherited from the initial user update.

For distribution threads (that cannot inherit user session attributes), the maxlocks value can be set using the following config parameter:

```
ii.nodename.dbms.server_name.rep_dt_maxlocks
```

This parameter must be set to an integer value indicating the maximum number of locks taken before escalation. The distribution threads use this value of maxlocks for all table opens, but only if it exceeds 50. If this value is not set, the threads use the generic system maxlocks value as set in the config.dat parameter:

```
ii.nodename.dbms.server_name.system_maxlocks
```

This value is used only if it exceeds 100. If neither parameter is set, the default value is 100 locks.

## Deadlock in the Distribution Threads

If the distribution threads receive a deadlock error, they roll back the transaction and try again immediately. If the deadlock problem persists after three tries, a warning is logged to the error log, and the distribution transaction is requeued to the internal pending list for later distribution. This occurs a few seconds or a few minutes later but is always in the current wakeup cycle of one of the distribution threads.

# Appendix A: Data Dictionary Tables

---

This section contains the following topics:

[System Catalogs](#) (see page 215)

[Base Table](#) (see page 223)

[Archive Table](#) (see page 223)

[Shadow Table](#) (see page 224)

[Internal Tables](#) (see page 225)

This appendix lists the tables in the data dictionary.

## System Catalogs

The system catalog tables in the data dictionary as as follows:

- dd\_cdds
- dd\_databases
- dd\_db\_cdds
- dd\_distrib\_queue
- dd\_input\_queue
- dd\_paths
- dd\_mail\_aler
- dd\_regist\_columns
- dd\_regist\_paths
- dd\_servers

### dd\_cdds Table

The dd\_cdds table defines each Consistent Distributed Data Set (CDDS) in the replicated environment:

Column Name	Data Type	Description
cdds_no	smallint not null	Numeric identifier assigned by the user to the CDDS. The range is 0-32,767.
cdds_name	char(32) not null	The name assigned by the user to uniquely identify the CDDS.

Column Name	Data Type	Description
collision_mode	smallint not null with default	Indicates the method used to handle collision conflicts in the CDDS. Values are: 0 = PassiveDetection 1 = ActiveDetection 2 = BenignResolution 3 = PriorityResolution 4 = LastWriteWins
error_mode	smallint not null with default	Indicates the method used to handle errors detected while transmitting replicated information within the CDDS. Values are: 0 = SkipTransaction 1 = SkipRow 2 = QuietCDDS 3 = QuietDatabase 4 = QuietServer
control_db	smallint not null with default	Reserved for future use.

## dd\_databases Table

The dd\_databases table defines each database in the replicated environment:

Column Name	Data Type	Description
database_no	smallint not null	Numeric identifier assigned by the user to uniquely identify the database in the replicated environment. The range is 1-32,767.
vnode_name	char(32) not null	Ingres Net virtual node where the database is located.
database_name	char(32) not null	The name of the database.
database_owner	char(32) not null with default	The name of the owner of the database.
dbms_type	char(8) not null with default	The type of database management system servicing



Column Name	Data Type	Description
		the database, for example, Ingres, Datacom/DB, DB2.
security_level	char(2) not null with default	The security level implemented in the database. Values are Blank, C2, and B1.
local_db	smallint not null with default	Indicates this is the local database. Only one row in this table can have a 1 in this column.
config_changed	char(25) not null with default	The last date and time that a configuration change was made that affected this database.
remark	varchar(80) not null with default	Comments about the database.

## dd\_db\_cdds Table

The dd\_db\_cdds table defines which CDDSs are present in which database:

Column Name	Data Type	Description
database_no	smallint not null	Identifies a database in the replicated environment.
cdds_no	smallint not null	Identifies a CDDS present in the database.
target_type	smallint not null	Indicates the expected replication behavior of the CDDS in the database. Values are: 1 = Full peer 2 = Protected read-only 3 = Unprotected read-only
is_quiet	smallint not null with default	Indicates that transmission of replicated information to the CDDS or the database is temporarily disabled, typically because of collisions or errors encountered.
server_no	smallint not null with default	Identifies the Replicator Server, if any, that propagates changes to the CDDS at the database.

## dd\_distrib\_queue Table

The dd\_distrib\_queue table contains the queue of operations (insert, update, delete) that are to be replicated to target databases. The queue is populated by distribution threads and manipulated by one or more Replicator Servers:

Column Name	Data Type	Description
targetdb	smallint not null	Identifies the database to which the transaction must be replicated.
sourcedb	smallint not null	Identifies the database where the transaction originated.
transaction_id	integer not null	Identifier given by the DBMS Server to the original transaction.
sequence_no	integer not null	The sequence of the operation (insert, update, delete) within the original transaction.
trans_type	smallint not null	Indicates whether this operation is an: 1 = insert 2 = update 3 = delete
table_no	smallint not null	Identifies the table changed by the original transaction.
old_sourcedb	smallint not null	Identifies the database that originated the transaction that manipulated the row before this transaction.
old_transaction_id	integer not null	Identifier given by the DBMS Server to the transaction that manipulated the row before this transaction.
old_sequence_no	integer not null	The sequence of the operation (insert, update, delete) within the transaction that manipulated the row before this transaction.
trans_time	date with null	The date and time when the transaction that manipulated the row was committed in the originating database.
cdds_no	smallint not null with default	The numeric identifier assigned to the CDDS. This can come directly from dd_regist_tables or indirectly

Column Name	Data Type	Description
		from a lookup table.
dd_priority	smallint not null with default	The user-assigned priority to resolve collisions. This can be 0 or have been derived from a lookup table.

## dd\_input\_queue Table

The dd\_input\_queue table contains the queue of operations (insert, update, delete) that are entering this database. The operations can come from a user interface on this database or from a Replicator server on another database:

Column Name	Data Type	Description
sourcedb	smallint not null	Identifies the database where the transaction originated.
transaction_id	integer not null	Identifier given by the DBMS Server to the original transaction.
sequence_no	integer not null	The sequence of the operation (insert, update, delete) within the original transaction.
table_no	smallint not null	Identifies the table changed by the original transaction.
old_sourcedb	smallint not null	Identifies the database that originated the transaction that manipulated the row before this transaction.
old_transaction_id	integer not null	Identifier given by the DBMS Server to the transaction that manipulated the row before this transaction.
old_sequence_no	integer not null	The sequence of the operation (insert, update, delete) within the transaction that manipulated the row before this transaction.
trans_time	date with null	The date and time when the transaction that manipulated the row was committed in the originating database.
trans_type	smallint not null	Indicates whether this operation is an:

Column Name	Data Type	Description
		1 = insert 2 = update 3 = delete
cdds_no	smallint not null with default	The numeric identifier assigned to the CDDS. This can come directly from dd_regist_tables or indirectly from a lookup table.
dd_priority	smallint not null with default	The user-assigned priority to resolve collisions. This can be 0 or can have been derived from a lookup table.

## dd\_mail\_alert Table

The dd\_mail\_alert table lists the users to be alerted by e-mail if a Replicator server in this database reports an error:

Column Name	Data Type	Description
mail_username	varchar(80) not null	The e-mail address of a user that needs to be notified by e-mail.

## dd\_paths Table

The dd\_paths table defines the paths to be taken by replicated information in a CDDS from the originating (source) database to a final target database. On each database, this table specifies where to propagate a change, based on the database where it originated:

Column Name	Data Type	Description
cdds_no	smallint not null	Identifies the CDDS.
localdb	smallint not null	Identifies the current database in the propagation path.
sourcedb	smallint not null	Identifies the database where the change originated.
targetdb	smallint not null	Identifies the next database to which the change must be propagated.
final_target	smallint not null	Indicates that the target database is the

Column Name	Data Type	Description
	null	final target in this path. This column is reserved for future use.

## dd\_regist\_columns Table

The dd\_regist\_columns table defines the columns of the tables that have been registered for replication:

Column Name	Data Type	Description
table_no	smallint not null	Numeric identifier assigned to the table by the configuration process.
column_name	char(32) not null	The name of the column.
column_sequence	integer not null	Indicates the sequence of the column in the base table. If the column is not replicated, its sequence is 0 (zero).
key_sequence	smallint not null with default	The order of the column in the unique key used for replication. If the value is 0, this column is not part of the key.

## dd\_regist\_tables Table

The dd\_regist\_tables table shows the tables that have been registered with the local database:

Column Name	Data Type	Description
table_no	integer not null	Numeric identifier assigned to the table by the configuration process.
table_name	char(32) not null	The name of the user table.
table_owner	char(32) not null	The table owner's user name.
columns_registered	char(25) not null with default	The last date and time the column or key registration was changed.

Column Name	Data Type	Description
supp_objs_created	char(25) not null with default	The last date and time support objects were created.
rules_created	char(25) not null with default	The last date and time that change recording was activated.
cdds_no	smallint with null	Identifies the CDDS to which this table belongs.
cdds_lookup_table	char(32) not null with default	The name of the lookup table to be used for partitioning this table horizontally.
prio_lookup_table	char(32) not null with default	The name of the priority lookup table to be used for resolving collisions by priority.
index_used	char(32) not null with default	The name of the storage structure (primary or secondary index) containing the unique key columns used for replication.

## dd\_servers Table

The dd\_servers table defines the Replicator Servers:

Column Name	Data Type	Description
server_no	smallint not null	The numeric identifier for a server.
server_name	varchar(24) not null	The name of the server.
pid	varchar(12) not null with default	The operating system process identifier for a running server.

## Base Table

The base table is an example of a replicated table that documents the replicated fields:

Column Name	Data Type	Description
column1	integer not null	Primary key of a replicated table.
column2	varchar(20) not null	Other attribute of a replicated table.

## Archive Table

The archive table contains all the replicated columns in the base table, and the replicated transaction key, which is made up of the following information:

- Database number of the originating database
- Transaction ID
- Transaction sequence number

Each row in the archive table reflects a row that was in the shadow table.

This is an example of a replicated archive table:

Column Name	Data Type	Description
column1	integer not null	Primary key of a replicated table.
column2	varchar(20) not null	Other attribute of a replicated table.
sourcedb	smallint not null	Identifies the database where the transaction originated.
transaction_id	integer not null	Identifier given by the DBMS Server to the original transaction.
sequence_no	integer not null	The sequence of the operation (insert, update, delete) within the original transaction.

## Shadow Table

A shadow table has all the primary key columns of the base replicated table plus other Ingres Replicator columns. This table contains information that is needed for the replication—the timestamp indicating when the row was manipulated and a priority number indicating the processing priority of the row in the event of collision:

Column Name	Data Type	Description
column1	integer not null	Primary key of a replicated table.
sourcedb	smallint not null	Identifies the database where the transaction originated.
transaction_id	integer not null	Identifier given by the DBMS Server to the original transaction.
sequence_no	integer not null	The sequence of the operation (insert, update, delete) within the original transaction.
trans_time	date with null	The date and time when the transaction that manipulated the row was committed in the originating database.
distribution_time	date with null	The time of final distribution to a local database.
in_archive	integer1 not null with default	Indicates whether this row is in the archive table.
cdds_no	smallint not null with default	Identifies the CDDS.
trans_type	smallint not null	Indicates if this operation is an: 1 = insert 2 = update 3 = delete
dd_priority	smallint not null with default	The user-assigned priority to resolve collisions. This can be 0 or have been derived from a lookup table.
new_key	smallint not null with default	This field is set to one for an update or delete if a key had to be created for the original record.
old_sourcedb	smallint not null with default	Identifies the database that originated the transaction that manipulated the row before this



Column Name	Data Type	Description
		transaction.
old_transaction_id	integer not null with default	Identifier given by the DBMS Server to the transaction that manipulated the row before this transaction.
old_sequence_no	integer not null with default	The sequence of the operation (insert, update, delete) within the transaction that manipulated the row before this transaction.

## Internal Tables

Ingres Replicator currently uses the following tables internally:

- dd\_dbms\_types
- dd\_flag\_values
- dd\_events
- dd\_last\_number
- dd\_option\_values
- dd\_server\_flags
- dd\_support\_tables



# Appendix B: Cluster Support

---

This section contains the following topics:

[Replicator in a Cluster Environment](#) (see page 227)

[-TPC Flag—Control Use of Two-phase Commit](#) (see page 227)

## Replicator in a Cluster Environment

Two-phase commit must be turned off for Ingres Replicator to work properly in a clustered environment.

## -TPC Flag—Control Use of Two-phase Commit

Ingres Replicator cannot propagate replications in a cluster environment in the same way that it does in noncluster environments. By default, Replicator Server uses two-phase commit, which is not supported at sites running Ingres Cluster Solution.

Replicator Server has a flag, -TPC, that allows all installations (including Ingres Cluster Solution) to control whether Replicator Server uses two-phase commit.

The -TPC flag can have the following values:

### **-TPC0**

Does not use two-phase commit.

### **-TPC1**

(Default) Uses two phase commit.

**Important!** *The lack of two-phase commit increases the possibility of loss of data integrity in uncompleted transactions.*



# Index

---

## A

- Activate (menu option)
  - Configuration Menu screen • 128
- Activate Change Recorders screen
  - activating • 128
  - deactivating • 128
  - described • 128
  - options for activation • 128
- activating
  - CDDS • 128
  - database • 128
- Activation (menu option)
  - Table Registration Summary screen • 123
- Active Detection mode • 55
- activity display, on Replication Monitor screen • 158
- adding
  - CDDS to CDDS Summary window • 111
  - database to the Database Summary list • 109
  - support objects to tables • 119
- alter table (statement) • 177
- archive table
  - defined • 223
  - full peer CDDS • 37
  - role in replication • 22

## B

- base table
  - defined • 223
- Benign Resolution mode • 55
- Both Queue and Shadow Table (option) • 91
- BothQueue&ShadowTable (menu option)
  - Table Registration Summary screen • 124
- B-tree structure
  - avoiding deadlock • 60

## C

- CDDS
  - adding • 111
  - deleting • 112
  - editing • 111
- CDDS (menu option)

- Configuration Menu window • 110
- Resolver Reports Menu screen • 195
- CDDS Data Propagation Paths Report • 195
- CDDS Databases and Servers window • 112
- CDDS Summary window • 110
  - Propagation Path Definition window • 114
  - Table Name pop-up • 116
  - viewing table registration • 116
- CDDS target type • 20, 36, 72
- CDDS Worksheet • 73
  - displayed • 73
  - filling in • 70
- Change Recorder
  - activating for a CDDS or database • 128
  - activating for a table • 88, 123
  - deactivating for a CDDS or database • 128
  - deactivating for a table • 88, 123
  - definition • 22
- Check for Keys at Server Startup (-NSR) • 139
- checking
  - existence of primary keys • 139
  - status of queues • 165
- checkpoints • 185
- Clear Quiet Server (-CLQ) • 141
- CLQ (Clear Quiet Server) • 141
- collision • 55
  - Active Detection (collision mode) • 55
  - assigning priority resolution lookup table • 88
  - Benign Resolution (collision mode) • 55
  - counted as an error • 54
  - creating priority collision resolution lookup table • 208
  - defined • 20
  - design • 52
  - detecting • 182
  - Last Write Wins (collision mode) • 55
  - Passive Detection (collision mode) • 55
  - preventing • 182
  - Priority Resolution (collision mode) • 55
  - Queue Collision Report • 188
  - resolving manually • 183, 184
- Collision (menu option)
  - Resolver Reports Menu screen • 188
- column rules • 63
- commands

---

- copydb • 179
- dd\_server • 170
- dereplic • 200
- issuing outside of Replicator • 171
- reconcil • 185
- repmgr • 96
- commands (Operations menu)
  - Activate Replication • 93
  - Dereplicate • 200
  - Propagate • 92
- commit.log (file) • 137
- configuration
  - checking • 192
  - information propagating • 92
  - of Replicator • 84, 101
- Configuration (menu option)
  - Replicator Manager main menu window • 104
- Configuration Menu screen
  - Activate Change Recorders screen • 128
  - Mail Notification List screen • 129
  - Move Configuration Data screen • 126
  - Table Registration Summary screen • 117
- Configuration Menu window
  - CDDS Summary window • 110
  - Database Summary window • 108
- Consistent Distributed Data Set (CDDS) • 20, 56, 71
  - activating • 128
  - adding to CDDS Summary window • 111
  - CDDS Data Propagation Paths Report • 195
  - collision handling • 54
  - controlling status • 141
  - deactivating • 128
  - defined • 32
  - deleting • 112
  - editing • 111
  - error handling • 56
  - example • 78
  - horizontal partitioning • 201
  - identifying • 70
  - showing server assignments • 163
  - specifying information for • 88
  - viewing table information • 116
- Consistent Distributed Data Set window • 111
- constraints, integrity • 177
- conventions
  - for Replicator database objects • 63
- copydb (command) • 179
- copying

- data into an existing table • 176
- replicated tables • 175
- Create Keys (menu option) • 91
- create table (statement) • 177
- CreateKeys (menu option)
  - Table Registration Summary screen • 124
- creating
  - error mail objects • 93
  - Ingres Net entries • 65
  - lookup tables • 201
  - mail notification list • 130
  - replicated transaction keys with the CreateKeys option • 124
  - replication objects • 85

## D

- data aggregation
  - replication design • 52
- data dictionary
  - archive table • 223
  - base table • 223
  - internal tables • 225
  - shadow table • 224
  - system catalog tables • 215
- data ownership
  - application design • 51
- data types
  - date • 63
  - float • 63
  - long byte • 63
  - long varchar • 63
  - money • 63
  - not supported for Enterprise Access • 63
  - not supported in replication key • 63
  - object\_key • 63
  - table\_key • 63
  - user defined • 63
- database • 55
  - activating • 128
  - adding to the Database Summary list • 109
  - assigning Replicator Server numbers • 58
  - CDDS target types • 36
  - checking for primary keys • 139
  - controlling status • 141
  - copying replicated tables • 175
  - Database Summary window • 108
  - deactivating • 128
  - defining the local database • 103

- 
- deleting from the Database Summary window • 110
  - disaster recovery • 185
  - editing information • 109
  - integrity constraints • 177
  - maintaining • 175
  - maximum allowed • 63
  - moving configuration data • 126
  - preparing for replication • 63
  - Replicated Databases Report • 194
  - showing server assignments • 163
  - Database Administrator
    - privileges needed • 64
    - role of • 45
  - database availability
    - maximizing • 40
  - database events
    - creating • 102
    - dd\_go\_server • 172
    - dd\_ping • 172
    - dd\_set\_server • 172
    - dd\_stop\_server • 172
    - selecting a server • 166
    - sending • 166
    - sending from an application • 172
    - sending from ISQL Terminal Monitor • 172
    - sending through Replication Monitor • 166
  - database information, on Replication Monitor screen • 158
  - database name, assigning • 68
  - database number, assigning • 68
  - database objects
    - rules • 63
  - Database Summary window
    - adding databases • 109
    - deleting databases • 110
    - editing a database • 109
  - database type • 68
  - Database Worksheet
    - displayed • 68
    - filling in • 68
  - databases
    - synchronizing with the Create Keys option • 91
  - Databases (menu option)
    - CDDS Summary window • 112
    - Configuration Menu window • 108
    - Resolver Reports Menu screen • 194
  - date (data type) • 63
  - DBMS type • 68
  - dd\_cdds (table)
    - defined • 215
  - dd\_databases (table)
    - defined • 216
  - dd\_db\_cdds (table)
    - defined • 217
  - dd\_distrib\_queue (table) • 158
    - checking status • 165
    - defined • 218
    - reporting collisions in • 188
    - role in replication • 24
  - dd\_go\_server (database event) • 172
  - dd\_input\_queue (table) • 158
    - checking status • 165
    - defined • 219
    - role in replication • 22
  - dd\_mail\_alert (table)
    - defined • 220
  - dd\_paths (table)
    - defined • 220
  - dd\_ping (database event) • 172
  - dd\_regist\_columns (table)
    - defined • 221
  - dd\_regist\_tables (table)
    - defined • 221
  - dd\_servers (table)
    - defined • 222
  - dd\_set\_server (database event) • 172
  - dd\_stop\_server (database event) • 172
  - dd\_transaction\_id (table)
    - defined • 223
  - DeActivate (menu option)
    - Activate Change Recorders screen • 128
    - Table Registration Summary screen • 123
  - deactivating
    - CDDS or database • 128
    - table • 123
  - deadlock
    - avoiding • 60
    - avoiding with B-tree structure • 60
    - avoiding with hash structure • 61
  - Default (menu option)
    - runrepl.opt file screen • 164
  - Define a Replicated Database window
    - adding databases • 109
    - editing database information • 109
  - Define Local Database window
    - described • 103
  - defining
    - Ingres Net entries • 65
-

---

- deleting
  - CDDS • 112
  - databases from the Database Summary list
    - 110
  - mail notification list entries • 130
- DeRegister (menu option)
  - Table Registration Summary screen • 118
- deregistering
  - tables • 118
- dereplic (command) • 200
- DerePLICATE (command) • 200
- derived information
  - replication design • 48
- DescribeFlag (menu option)
  - runrepl.opt file screen • 164
- designing
  - replication scheme • 66
- disaster recovery • 185
- DistribConfig (menu option)
  - Resolver Reports Menu screen • 192
- Distributed Configuration Checker Report
  - described • 192
  - example • 192
- Distributed Database Administrator
  - grants needed • 64
  - role of • 45
- Distribution Server • 56
  - definition • 22
  - error handling • 56

## E

- EditConfig (menu option)
  - Replication Monitor screen • 164
- editing
  - CDDS information • 111
  - database information • 109
  - runrepl.opt file • 164
  - table registration • 119
- e-mail
  - creating a mail notification list • 130
  - notifying users of errors • 134
- Enterprise Access
  - data types not supported • 63
- Enterprise Access databases
  - unprotected read-only target type • 38
- error
  - creating a mail notification list • 130
  - DBMS Server • 137
  - handling • 56, 139

- messages • 137
- primary key checking • 139
- role of servers • 56
- error count, on Replication Monitor screen • 158
- Error Mail (-MLE) • 134
- error mail objects, creating • 93
- Event Timeout (-EVT) • 171
- events
  - raising at the database level • 157
  - raising at the server level • 157
- Events (menu option)
  - Replication Monitor screen • 166

## F

- files
  - commit.log • 137
  - print.log • 137
  - replicat.log • 137
  - runrepl.opt • 134
- filling in the CDDS Worksheet • 72
- flags
  - descriptions from runrepl.opt file screen • 164
  - descriptions of server parameters • 144
  - EMX (Maximum Error) • 135, 140
  - EVT (Event Timeout) • 171
  - IDB (Local Database Name) • 144
  - LGL (Log Level) • 134, 137
  - MLE (Error Mail) • 134
  - NML (No Error Mail) • 134
  - OWN (Local Database Owner) • 144
  - PTL (Print Logging Level) • 137
  - QBM (Queue Read Limit) • 143
  - QBT (Transaction Break Limit) • 143
  - QIT (Quiet Server) • 141, 170
  - selecting flags in runrepl.opt file screen • 164
  - Server Startup Settings Worksheet • 152
  - setting defaults • 164
  - SGL (Single Run) • 170
  - SVR (Server Number) • 144
  - TOT (Lock Timeout) • 143
  - TPC • 227
- float (data type) • 63

## G

- granting



---

operations to Distributed DBA • 64

## H

Hash structure

avoiding deadlock • 61

horizontal partitioning

assigning a lookup to a table • 88

creating a lookup table • 201

Table Worksheet • 77

## I

incoming records, on Replication Monitor  
screen • 158

Ingres Cluster Solution • 14, 28

Ingres Net

creating entries • 65

Replicator requirement • 18

role in replication • 25

Ingres Replicator

activate table • 88

application design issues • 47

CDDS lookup table • 88

Change Recorder • 22

compared to Ingres Star • 17

configuring using Replicator Manager • 95

configuring using Visual DBA • 83

conventions for database objects • 63

creating catalogs • 102

design concepts • 31

design issues • 52

design schemes • 38

designing your replication scheme • 66

disaster recovery • 185

Distribution Server • 22

illustrated configuration • 18

key terminology • 20

overview • 20, 22, 131

processing tables • 30

reconfiguring • 177

replication scheme examples • 78

Replicator Manager • 95

Replicator Server • 131

repmgr command • 96

resolving collisions • 182

security • 17

starting the Replicator Server • 133

Ingres Star

compared to Replicator • 17

integrity

create integrity (statement) • 177

defining constraints • 177

Table Integrity Report • 189

Integrity (menu option)

Resolver Reports Menu screen • 189

## J

journaling

disaster recovery • 185

## L

lartool (utility) • 28

Last Write Wins mode • 55

Local Database Name (-IDB) • 144

Local Database Owner (-OWN) • 144

Lock Timeout (-TOT) • 143

log files

commit.log • 137

print.log • 137

replicat.log • 137

Log Level (-LGL) • 134, 137

logging levels, setting • 145

long byte (data type) • 63

long varchar (data type) • 63

lookup table

horizontal partitioning • 201

priority collision resolution • 208

## M

Mail Notification List screen

creating a list • 130

deleting entries • 130

described • 129

MailList (menu option)

Configuration Menu screen • 129

Maximum Error (-EMX) • 135, 140

maximum error count • 145

menu items

selecting in Replicator Manager • 97

menu maps

Replication Monitor • 162

messages

informational • 137

levels • 138

two-phase commit • 137

warning • 137

---

- money (data type) • 63
- Monitor (menu option)
  - Replicator Manager main menu screen • 158
- monitoring
  - incoming records • 158
  - outgoing records • 158
  - queues • 158
  - replicated databases • 158
  - replication collisions • 157
  - replication events, raising at the database level • 157
  - replication events, raising at the server level • 157
  - Replicator Server statuses • 157
  - server status • 158
  - table integrity after replication • 157
- Move Configuration Data screen
  - data moved • 126
  - data not moved • 126
  - described • 126
- MoveConfig (menu option)
  - Configuration Menu screen • 126
- moving
  - configuration data • 126

**N**

- naming guidelines • 71
- naming guidelines for database • 68
- No Error Mail (-NML) • 134
- No Key Checking (-SCR)
  - flags • 139
- NQT (Unquiet Server) • 141
- NSR (Check for Keys at Server Startup) • 139
- numbering guidelines • 71
- numbering guidelines for database • 68

## O

- object\_key (data type) • 63
- OpenSQL
  - standard for table names • 63
- outgoing records, on Replication Monitor screen
  - 158

## P

- parameters
  - documenting via worksheet • 152

- Replicator Server • 145
  - setting • 164
- Passive Detection mode • 55
- Paths (menu option)
  - CDDS Summary window • 114
- ping
  - sending • 165
- Ping (menu option)
  - Replication Monitor screen • 163
- ping count, on Replication Monitor screen • 158
- primary keys
  - checking for existence • 139
- Print Logging Level (-PTL) • 137
- print.log (file) • 137
- priority collision resolution
  - creating lookup table • 208
- Priority Resolution mode • 55
- privileges
  - Database Administrators • 64
  - using Replicator Manager • 96
- propagating
  - configuration information • 92
- Propagation Path Definition window
  - described • 114
- propagation paths • 20
  - CDDS Propagation Paths Report • 195
  - defined • 34
  - example • 78

## Q

- QCD (Quiet CDDS) • 141
- QDB (Quiet Database) • 141
- QIT (Quiet Server) • 141
- Queue Collision Report
  - described • 188
- queue information, on Replication Monitor screen • 158
- Queue Read Limit (-QBM) • 143
- Queues (menu option)
  - Replication Monitor screen • 158, 165
- Quiet CDDS (error mode)
  - error • 56
- Quiet CDDS (-QCD) • 141
- Quiet Database (error mode)
  - error • 56
- Quiet Database (-QDB) • 141
- Quiet Server (error mode)
  - error • 56
- Quiet Server (-QIT) • 141, 170

---

quieting due to error • 56

## R

### records

- incoming • 158
- outgoing • 158

### Register (menu option)

- Table Registration Summary screen • 118

### Registered Tables Report

- described • 197

### registering

- tables • 118

### removing objects

- from replicated database • 200

### repcat (utility) • 102

### repinst (command) • 131

### repicat.log (file) • 137

### Replicated Databases Report

- described • 194
- example • 194

### replicated transaction key • 20

- using the CreateKeys option • 124

### replicated transaction keys

- using the Create Keys option • 91

### replication

- activating • 93
- adding databases • 86
- choosing columns • 88
- defining error mail destinations • 93
- installing objects • 85
- monitoring collisions • 157
- monitoring replicated databases • 158
- monitoring server statuses • 157
- monitoring table integrity • 157
- monitoring using Replicator Manager • 158
- monitoring using Visual DBA • 156
- monitoring using Visual Performance Monitor • 156
- propagating information • 92
- raising events at the database level • 157
- raising events at the server level • 157
- registering tables • 88
- removing objects • 200

### replication design

- application issues • 47
- data aggregation • 52
- data ownership • 51
- derived information • 48
- preventing collisions • 52

- sample configurations • 38

### replication key

- data types not supported • 63

### Replication Monitor

- menu map • 162

### Replication Monitor screen

- changing startup parameters • 164
- described • 158
- editing the runrepl.opt file • 164
- runrepl.opt file screen • 164
- Send Database Event screen • 166
- Server Assignments screen • 163
- setting startup parameters • 164

### replication scheme

- designing • 66
- example configurations • 38
- examples • 78

### Replicator Manager

- Define Local Database window • 103
- definition • 29
- operating • 97
- parameters • 96
- privileges to use • 96
- Resolver reports • 186
- running the first time • 103
- selecting menu items • 97
- starting • 96

### Replicator Server • 56

- assigning example • 80
- assigning numbers • 58
- config and log file locations • 152
- configuration options for monitoring • 134
- defined • 22
- editing the runrepl.opt file • 164
- error handling • 56, 139
- flags • 145
- managing memory • 143
- parameter settings • 144
- parameters • 145
- replication cycle • 144
- replication cycle duration • 171
- role in collision handling • 54
- role in replication • 131
- showing assignments • 163
- starting • 133

### Replicator service, installing • 131

### repmgr (command) • 96

### reports

- CDDS Data Propagation Paths • 195
- Distributed Configuration Checker • 192

- Queue Collision • 188
- Registered Tables • 197
- Replicated Databases • 194
- Resolver option • 186
- Table Integrity • 189
- requirements
  - Ingres Net • 18
- Resolver (menu option)
  - Replicator Manager main menu screen • 186
- Resolver Reports Menu screen
  - CDDs Data Propagation Paths Report • 195 described • 186
  - Distributed Configuration Checker Report • 192
  - Queue Collision Report • 188
  - Registered Tables Report • 197
  - Replicated Databases Report • 194
  - Table Integrity Report • 189
- role of database number in resolution • 55
- role of number in collision resolution • 55
- rules
  - role in replication • 22
- runrepl.opt (file)
  - described • 134
  - editing configuration file • 164
  - saving • 164
- runrepl.opt file screen
  - choosing defaults • 164
  - editing server parameters • 164
  - saving the file • 164
  - selecting flags • 164
  - viewing flag descriptions • 164

## S

- Save (menu option)
  - runrepl.opt file screen • 164
- scheduling
  - server activity • 170
- SCR (No Key Checking) • 139
- security • 17
- Select Server Number to Receive Event screen
  - selecting a server • 166
- SelectFlag (menu option)
  - runrepl.opt file screen • 164
- Selecting
  - server to receive database event • 166
- Send Database Event screen
  - described • 166

- Select Server Number to Receive Event screen • 166
- sending an event • 167
- sending
  - database events • 166
- Server Assignments screen
  - described • 163
- Server Number (-SVR) • 144
- Server Startup Settings Worksheet
  - described • 152
  - example • 152
- server status, on Replication Monitor screen • 158
- servers • 141
  - changing startup parameters • 166
  - changing status • 141
  - controlling status • 141
  - pinging • 165
  - processing activity • 140
  - scheduling • 170
  - scheduling with event timeout • 171
  - starting from Replication Monitor screen • 165
  - starting with events • 170
  - stopping • 168
- SGL (Single Run) • 141
- shadow table
  - defined • 224
  - full peer CDDs • 37
  - populating with Create Keys option • 91
  - populating with CreateKeys option • 124
  - protected read-only target • 38
  - role in replication • 22
- Shadow Table Only (option) • 91
- ShadowTableOnly (menu option)
  - Table Registration Summary screen • 124
- ShowAssign (menu option)
  - Replication Monitor screen) • 163
- Single Run (-SGL) • 141, 170
- single run processing • 141
- Skip Row (error mode)
  - error • 56
- Skip Transaction (error mode)
  - error • 56
- Start (menu option)
  - Replication Monitor screen) • 165
- starting
  - Replicator Server • 133
  - servers from Replication Monitor screen • 165

---

- statements
  - alter table • 177
  - bulk deletions • 177
  - create integrity • 177
  - create table • 177
  - delete • 177
  - modify • 177
- Stop (menu option)
  - Replication Monitor screen • 168
- stopping servers • 168
- Support (menu option)
  - Table Registration Summary screen • 122
- synchronizing
  - databases with the CreateKeys option • 124

## T

- table
  - OpenSQL standard length • 56, 63
  - rules for names • 63
- Table Integrity Report screen
  - Resolver Reports Menu screen • 189
- Table Name window
  - described • 116
- Table Registration Details screen
  - described • 119
- Table Registration Summary screen
  - activating tables • 123
  - adding support objects • 119
  - creating replicated transaction keys • 124
  - deactivating tables • 123
  - deregistering tables • 118
  - described • 117
  - editing table registration • 119
  - registering tables • 118
  - Table Registration Details screen • 119
- Table Worksheet
  - displayed • 77
  - filling out • 77
- table\_key (data type) • 63
- tables
  - adding support objects • 119
  - assigning horizontal partition lookup • 88
  - assigning priority collision resolution lookup • 88
  - copying between databases • 175
  - creating lookup • 201
  - deregistering • 118
  - editing registration • 119
  - maintaining • 175

- Registered Tables Report • 197
- registering • 118
- registering for replication • 88
- Table Integrity Report • 189
- viewing table information for a CDDS • 90, 116
- Tables (menu option)
  - Configuration Menu screen • 117
  - Resolver Reports Menu screen • 197
- TPC (flag) • 227
- Transaction Break Limit (-QBT) • 143
- turning off two-phase commit • 227
- two-phase commit
  - commit.log file • 137
  - messages • 137
  - turning off • 227

## U

- u (flag) • 96
- UCD (Unquiet CDDS) • 141
- UDB (Unquiet Database) • 141
- Unquiet CDDS (-UCD) • 141
- Unquiet Database (-UDB) • 141
- Unquiet Server (-NQT)
  - flags • 141
- user-defined data types • 63
- utilities
  - lartool • 28
  - repcat • 102

## V

- vertical partitioning
  - Table Worksheet • 77
- Visual DBA
  - using to monitor replication • 131
- Visual Performance Monitor
  - using to monitor replication • 156
- verses VDBA • 131

## W

- worksheets
  - CDDS • 70
  - database • 68
  - Server Startup Settings • 152
  - Table • 77