SYBASE®

Administration Guide

# Mirror Activator™ for Oracle

12.6

[ Linux, Microsoft Windows, and UNIX ]

DOCUMENT ID: DC00264-01-1260-01

LAST REVISED: March 2006

# Contents

# About This Book

Mirror Activator for Oracle™ is a Sybase® software solution that allows you to combine the benefits of transaction replication and disk replication, thereby eliminating the disadvantages of using either system alone in a disaster recovery solution.

The Mirror Activator for Oracle solution includes the following Sybase software products:

- Mirror Replication Agent™
- Replication Server®
- Enterprise Connect™ Data Access Option for Oracle (DirectConnect)

**Audience**

This book is for anyone who needs to manage or administer a Mirror Activator for Oracle system. This may include:

- System Administrators
- Database Administrators
- Network Administrators

**How to use this book**

This book is organized as follows:

Chapter 1, "Overview and Introduction," provides an overview of replication solutions for disaster recovery, and introduces the Mirror Activator for Oracle solution and the Mirror Replication Agent component.

Chapter 2, "Setup and Configuration," describes how to set up and configure the Mirror Replication Agent, and other software components of the Mirror Activator for Oracle system.

Chapter 3, "Administering Mirror Replication Agent," describes administrative tasks and procedures for the Mirror Replication Agent.

Chapter 4, "Troubleshooting Mirror Replication Agent," describes basic troubleshooting and system recovery procedures.

Appendix A, "Materializing Databases," describes how to materialize the databases in a Mirror Activator for Oracle system.

Appendix B, “Failover and Failback with Mirror Activator for Oracle,” describes procedures for failover and failback in a Mirror Activator for Oracle system.

Appendix C, “Mirror Replication Agent and Oracle Databases,” describes general issues and considerations that are specific to using Mirror Replication Agent version 12.6 with the Oracle data server.

**Related documents**

If you are not familiar with Sybase transaction replication technology, refer to the following documents for more information:

- Replication Server *Design Guide* for an introduction to basic transaction replication concepts and Sybase transaction replication systems
- Replication Server *Administration Guide* for an introduction to Replication Server support for warm standby applications
- Replication Server *Heterogeneous Replication Guide* for detailed information about configuring Replication Server and implementing a Sybase replication system with non-Sybase databases

To learn more about the Mirror Activator for Oracle solution and the Mirror Replication Agent component, refer to the following documents:

- Mirror Activator for Oracle *Mirror Replication Agent Reference Manual* for information about all Mirror Replication Agent commands and configuration properties, including syntax, examples, and detailed command usage notes
- Mirror Activator for Oracle *Installation Guide* for information about installing the Mirror Replication Agent software
- The Mirror Activator for Oracle *Release Bulletin* for last-minute information that was too late to be included in the books

**Note** A more recent version of the Mirror Activator for Oracle *Release Bulletin* may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Technical Library Web site.

Make sure you have appropriate documentation for the Oracle version used with the Mirror Activator for Oracle system.

**Java environment**

The Mirror Replication Agent component requires a Java Runtime Environment (JRE) on the Mirror Replication Agent host machine.

- The Mirror Activator for Oracle *Release Bulletin* contains the most up-to-date information about Java and JRE requirements.

- Java documentation available from your operating system vendor describes how to set up and manage the Java environment on your platform.

Further information about Java environments can be found at the following URL:

```
http://java.sun.com
```

**Other sources of information**

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

**Sybase certifications on the Web**

Technical documentation at the Sybase Web site is updated frequently.

❖ **To find the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.

3 In the Certification Report filter select a product, platform, and timeframe, and then click Go.

4 Click a Certification Report title to display the report.

❖ **To find the latest information on component certifications**

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product, or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **To create a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

## Sybase EBFs and software maintenance

❖ **To find the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

5. Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

**Style conventions**

The following style conventions are used in this book:

- In a sample screen display, commands that you should enter exactly as shown appear like this:

  ```
  pdb_init
  ```

- In the regular text of this document, variables or user-supplied words appear like this:

  Specify the *value* option to change the setting of the configuration parameter.

- In a sample screen display, variables or words that you should replace with the appropriate value for your site appear like this:

  ```
  resume connection to pds.pdb
  ```

  where *pds* and *pdb* are the variables you should replace.

- In the regular text of this document, names of programs, utilities, procedures, and commands appear like this:

  Use the pdb_init command to initialize the primary database.

- In the regular text of this document, names of database objects (tables, columns, stored procedures, etc.) appear like this:

  Check the price column in the widgets table.

- In the regular text of this document, names of datatypes appear like this:

  Use the date or datetime datatype.

- In the regular text of this document, names of files and directories appear like this:

  Log files are located in the *$SYBASE/MRO-12_6/inst_name/log* directory.

**Syntax conventions**

The following syntax conventions are used in this book:

***Table 1: Syntax conventions***

| Key | Definition |
|---|---|
| { } | Curly braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you enter the command. |
| [ ] | Brackets mean that choosing one or more of the enclosed options is optional. Do not type the brackets when you enter the command. |
| ( ) | Parentheses are to be typed as part of the command. |
| \| | The vertical bar means you can select only one of the options shown. |
| , | The comma means you can choose as many of the options shown as you like, separating your choices with commas that you type as part of the command. |

In reference sections of this document, statements that show the syntax of commands appear like this:

ra_config [*param*[, *value*]]

The words *param* and *value* in the syntax are variables or user-supplied words.

**Character case conventions**

The following character case conventions are used in this book:

- All command syntax and command examples are shown in lowercase. However, Mirror Replication Agent command names are *not* case sensitive. For example, PDB_INIT, Pdb_Init, and pdb_init are equivalent.
- Names of configuration parameters are case sensitive. For example, Scan_Sleep_Max is not the same as scan_sleep_max, and the former would be interpreted as an invalid parameter name.
- Database object names are *not* case sensitive in Mirror Replication Agent commands. However, if you need to use a mixed-case object name in a command (to match a mixed-case object name in the primary database), you must delimit the object name with quote characters. For example:

```
pdb_get_tables "TableName"
```

**Accessibility features**

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Mirror Activator for Oracle and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

The online help for this product is also provided in HTML, which you can navigate using a screen reader.

---

**Note** You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

---

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

For a Section 508 compliance statement for Mirror Activator for Oracle, see Sybase Accessibility at http://www.sybase.com/detail?id=1028493.

**If you need help**

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

# Overview and Introduction

This chapter provides an overview of replication solutions for disaster recovery, and introduces the Mirror Activator for Oracle solution and the Mirror Replication Agent component.

## Disaster recovery options

Unfortunately, effective disaster recovery solutions can be viewed as “expensive insurance” because of the high total cost of ownership (TCO) and low return on investment (ROI) that is typically associated with them.

Upper-tier disaster recovery solutions require a means to replicate mission-critical data from a primary site to a remote standby site. The following sections describe two replication alternatives for disaster recovery: *disk* replication and *transaction* replication.

### Disk replication

Disk replication (or disk mirroring) systems replicate the contents of a disk volume or file system from a primary site to a standby site. The main benefits of disk replication are:

- Redundancy of primary disk devices at a remote standby site
- Zero or near-zero data loss, particularly with synchronous replication

The benefits of disk replication come at a relatively high cost, and there are significant disadvantages and gaps in the protection that disk replication provides:

- A substantial cost of disk replication results from forcing standby system resources to go offline for periodic updates.

  Disk replication requires exclusive write control of mirror devices at the standby site, which conflicts with the DBMS requirement for exclusive device write control, even when client applications cannot change data.

  To avoid keeping standby systems completely idle or offline until a failover, local disk replication must be implemented at the standby site to make periodic "snapshots" of mirror devices available to the standby systems.

- Standby system resources are hardware-dependent.

  Disk replication is device-based, sending data from a primary device in blocks (or pages) to a mirror device. Hardware and operating systems at the primary site must be duplicated at the standby site, and they must be identical to the primary site configuration.

- Data integrity (transactional consistency) between primary and standby databases cannot be guaranteed.

  Disk replication updates mirror devices by block (or page) boundaries, with no knowledge of transaction boundaries. Because a single transaction can change several blocks, on several devices, the standby database can be corrupted if transmission from the primary site is interrupted before all of the affected blocks (on all of the affected devices) are received at the standby site.

- Disk replication provides no protection from disk corruptions at the primary site, which are replicated block-for-block to mirror devices at the standby site.

- Disk replication requires high network bandwidth for synchronous replication with acceptable application response time, particularly for databases with high transaction volumes.

## Transaction replication

Transaction replication systems read the transaction log of the primary database, convert the log records into equivalent SQL commands, and then apply the SQL to a standby database.

Benefits of transaction replication

The main benefits of transaction replication are:

- Data integrity (transactional consistency) between primary and standby databases is guaranteed.

  Transaction replication is *logical* replication. It is based on the transaction log of the primary database, so it follows all of the data integrity "rules" of the database.

- Standby databases are always available for decision support and reporting applications, with no downtime required by the transaction replication system.

  Transaction replication requires the standby database to be continuously online, so it can apply replicated transactions as they arrive in the primary database log. Transaction replication does not require control of any database devices or log devices.

  **Note** Availability of standby system resources increases the ROI on both hardware and the DBMS software required to support disaster recovery.

- Standby system resources are hardware-independent because transaction replication is logical and not device-specific.
- The standby site is protected from disk corruption because transaction replication is based on the primary database transaction log, which is managed at the device level by the primary DBMS.
- Network bandwidth requirements are reduced because:
  - Transactions rolled back in the primary database do not need to be replicated.
  - Transaction replication can encode the replicated transactions and data in a more compact form than the actual log records or raw SQL. This results in more efficient network communication.

  **Note** High network bandwidth can be a substantial operating cost, so reducing the bandwidth requirement lowers the TCO of a disaster recovery solution.

## Disadvantages of transaction replication

There are also some disadvantages of transaction replication when used alone for disaster recovery:

- Disaster recovery is provided only for databases, and not for applications or other data stores at the primary site.
- Transaction replication is asynchronous, so:

- Network congestion can increase latency, which can increase the recovery time.

  Latency is the time elapsed between committing a transaction at the primary database and committing that transaction at the standby database.

- Some data loss can occur in certain situations.

  If the connection to the primary site goes down after a transaction is committed, but before the replication system reads the primary log record, the transaction cannot be replicated.

## The Mirror Activator for Oracle solution

Mirror Activator for Oracle is a Sybase software solution that allows you to combine the benefits of transaction replication and disk replication, thus eliminating the disadvantages of using either system alone in a disaster recovery situation.

### Mirror Activator for Oracle software

The Mirror Activator for Oracle solution includes the following Sybase software products:

- Mirror Replication Agent
- Replication Server
- DirectConnect™ for Oracle (Enterprise Connect Data Access Option for Oracle)

#### Mirror Replication Agent

Mirror Replication Agent reads primary database transaction log devices to acquire transactions to replicate. Mirror Replication Agent is specifically designed to read remote (or mirror) log devices that reside in a separate location from the primary data server and its devices. Mirror Replication Agent requires only read access to the mirror log devices. After acquiring transaction data from the mirror log devices, Mirror Replication Agent sends replicated transactions to Replication Server, for distribution to the standby database.

**Replication Server**

Replication Server provides the infrastructure for a Sybase transaction replication system. It manages the standby database, applying replicated transactions, and it maintains its own device-based queues to provide guaranteed transaction delivery. Replication Server also maintains its own system database, called the Replication Server System Database (RSSD), where it stores replication system data and metadata.

**DirectConnect for Oracle**

DirectConnect for Oracle is a Sybase product that provides access to Oracle databases through a Sybase Open Client interface.

## Mirror Activator for Oracle advantages

Compared to disaster recovery solutions that are based on either disk replication or transaction replication alone, advantages of the Mirror Activator for Oracle solution are:

- Lower total cost of ownership (TCO)
- Enhanced return on investment (ROI)
- An integrated disaster recovery solution, with the "best of both worlds"

**Lower TCO**

The Mirror Activator for Oracle solution provides lower TCO with:

- Reduced network bandwidth requirements, because only the primary log devices need to be mirrored (primary data devices need not be mirrored)
- Hardware-independence for the standby DBMS (both hardware and operating system flexibility)

**Enhanced ROI**

The Mirror Activator for Oracle solution provides enhanced and immediate ROI with:

- No downtime required for standby databases (always online and available for client applications, always up-to-date)
- No idle time required for standby hardware and infrastructure (always available when not used for recovery)

**Integrated solution benefits**

Mirror Activator for Oracle provides an integrated disaster recovery solution with:

- Standby databases protected from disk corruption (by logical, not literal, replication)
- Synchronous replication, with zero data loss *and* guaranteed data integrity (transactional consistency)

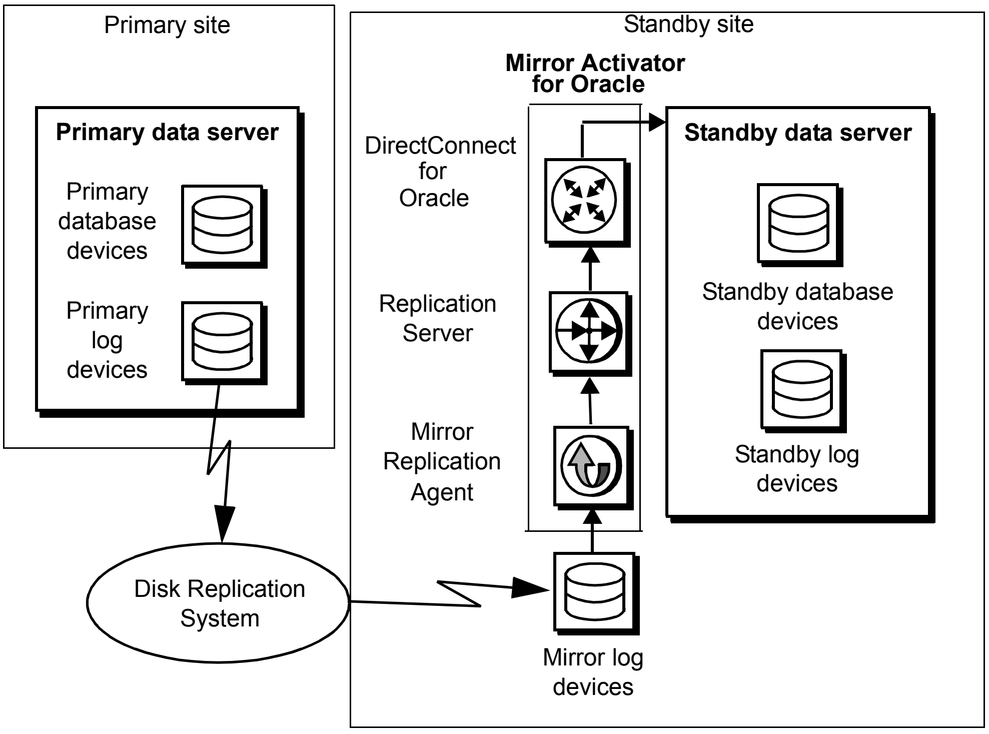
- Complete coverage for databases, as well as non-database systems

## How Mirror Activator for Oracle works

A Mirror Activator for Oracle system consists of a special variant of the Sybase transaction replication system, which is integrated with a disk replication system (provided by another vendor).

Figure 1-1 shows a typical Mirror Activator for Oracle system configuration. Arrows illustrate the flow of mirrored log device data and replicated transactions during normal Mirror Activator for Oracle system operation, that is, while replicating transactions from the primary database to the standby database.

***Figure 1-1: Mirror Activator for Oracle system***



**Note** The disk replication system is the Mirror Activator for Oracle system component that moves log data from the primary site to the standby site. Components of the disk replication system reside at both primary and standby sites.

## Mirror Activator for Oracle system components

As shown in Figure 1-1, the main components of a Mirror Activator for Oracle system are:

- Primary database
- Disk replication system
- Mirror log devices
- Mirror Replication Agent
- Replication Server
- DirectConnect for Oracle
- Standby database

### Primary database

The primary database is the source of transactions that are replicated to the standby database. The primary data server maintains the primary database transaction log, and it controls the primary database log devices.

### Disk replication system

The disk replication system replicates data at the device level. It may include a storage area network (SAN), network attached storage (NAS), or disk mirroring/synchronization mechanism, and it may incorporate both hardware and proprietary system software.

### Mirror log devices

The mirror log devices are off-site copies of the primary database transaction log devices. They are managed by the disk replication system, which requires exclusive write control of the mirror devices, so they are accessible on a read-only basis.

### Mirror Replication Agent

Mirror Replication Agent reads primary database transactions from mirror log devices, and then sends those transactions to Replication Server for distribution to the standby database. The Mirror Replication Agent requires only read access to mirror log devices.

Replication Server

Replication Server receives the replicated transactions from Mirror Replication Agent. Replication Server processes the transactions and converts them into SQL, which it sends to the standby database for processing. When the replicated transactions are processed successfully in the standby database, the standby database is synchronized with the primary database.

DirectConnect for Oracle

DirectConnect for Oracle consists of a database gateway server that allows you to use the Sybase Open Client and Open Server protocol (such as Replication Server) to connect with non-Sybase data servers, using either the data server's native communications protocol or the standard, ODBC protocol.

Standby database

The standby database is the destination of transactions that are replicated from the primary database. During normal Mirror Activator for Oracle system operation, the standby database is always online, processing the replicated transactions it receives from the Replication Server.

During normal system operation, the Replication Server is the only client allowed to send data-changing transactions to the standby database. All other clients are restricted to read-only database access.

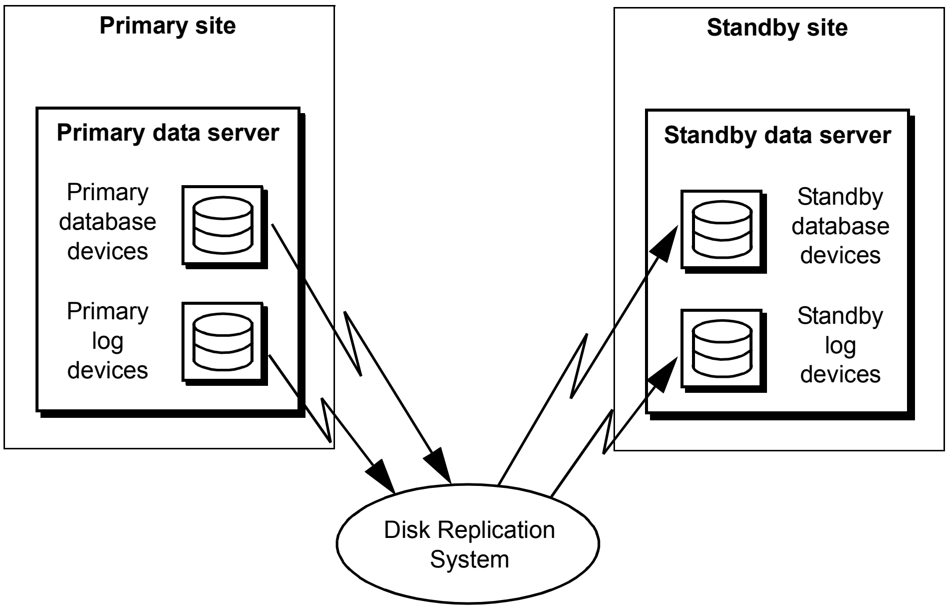
## Materializing the standby database

Transaction replication requires a "starting point," where the primary and standby databases are identical, with the same data and schema. If the standby database is not identical to the primary database, it must be *materialized* before transaction replication begins.

With integrated disk replication, you can materialize the standby database by copying a *snapshot* of the primary data and log devices to the standby site. After the snapshot is received at the standby site, the standby database can be brought online.

Figure 1-2 illustrates how snapshot data moves from devices at the primary site to devices at the standby site, to materialize the standby database.

***Figure 1-2: Standby database materialization***



To materialize the standby database, the primary database must be *quiesced*, so that all client applications are locked out of the database, any transactions in progress are frozen, and the primary data and log devices are stabilized.

While the primary database is quiesced, the disk replication system captures a snapshot image of the primary data and log devices. The disk replication system copies the snapshot to the standby data and log devices, which loads the standby database with data and schema that are identical to the primary database.

After materializing the standby database, the disk replication system disconnects from the standby devices, and sends an additional copy of the primary log device snapshot to materialize the mirror log devices. Then, the disk replication system is set to mirror (or synchronously replicate) all subsequent changes on the primary log devices to the mirror log devices, which are the source of all transactions replicated by the Mirror Activator for Oracle system.

After the snapshot is captured at the primary site, and before primary database access is restored to other client applications, the Mirror Activator for Oracle system executes a procedure in the primary database to place a *marker* in the transaction log. The Mirror Activator for Oracle system uses the log marker to identify its "starting point" for transaction replication.

After the marker is recorded in the primary database transaction log, primary database access for client applications can be restored.

**Note** You can use a similar materialization process to restore the primary database after failover to the standby.

## Fault tolerance and automatic recovery

The Mirror Activator for Oracle system is highly fault-tolerant. It is designed to avoid any single point of failure, and to provide automatic and graceful recovery from any system failure.

During normal Mirror Activator for Oracle system operation, the disk replication system mirrors the primary database log devices synchronously, which guarantees zero data loss—that is, no data for a completed transaction will be lost.

Device-based queues that the Replication Server maintains guarantee transaction delivery to the standby database, even if network failures occur at the standby site, or if the standby database itself fails.

The Mirror Activator for Oracle system maintains a *queue ID* in both the Mirror Replication Agent and the Replication Server, which it uses to keep track of the most recent confirmed transaction (successfully replicated) in the standby database. The queue ID allows the Mirror Activator for Oracle system to automatically recover from system faults, and it prevents either the Mirror Replication Agent or the Replication Server component from becoming a single point of failure.

## Introduction to Mirror Replication Agent

Mirror Replication Agent is the Mirror Activator for Oracle system component that reads primary database transactions from mirror log devices, and sends those transactions to Replication Server for distribution to the standby database.

Mirror Replication Agent is a Java application that runs as a standalone process, independent of any other Mirror Activator for Oracle system component. It requires a Java Runtime Environment (JRE) on its host platform.

Mirror Replication Agent can reside on the same host as the Replication Server or the standby data server, or it can reside on a machine separate from any other Mirror Activator for Oracle system component. The Mirror Replication Agent host must have local access to the mirror log devices.

## Mirror Replication Agent instances

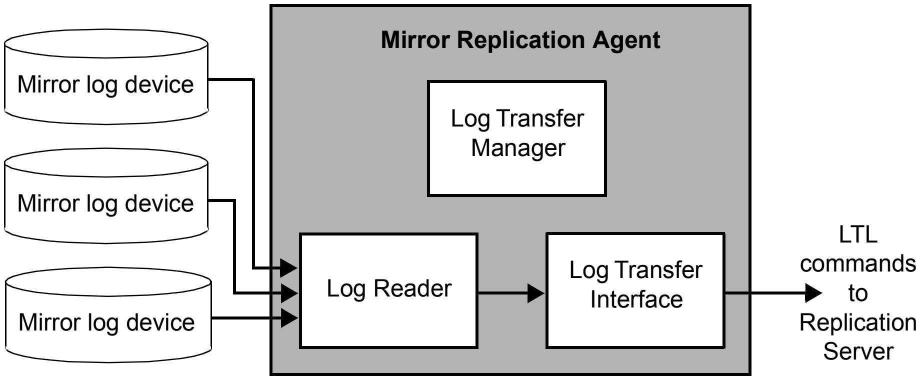
You must create one instance of the Mirror Replication Agent for each primary database that you want to replicate transactions from. Each Mirror Replication Agent instance is an independent process, with its own instance directories to house its configuration file, system log files, and system data repository.

## Mirror Replication Agent components

Mirror Replication Agent consists of a set of components that work together to propagate transactions from the mirror log devices to the Replication Server.

Figure 1-3 shows the flow of transaction data through a Mirror Replication Agent during normal (transaction replicating) operation.

***Figure 1-3: Mirror Replication Agent transaction data flow***



The main Mirror Replication Agent components are:

- Log Reader – reads the primary database transaction log on mirror log devices to retrieve transactions for replication, generates change-set data, and passes change sets to the Log Transfer Interface component.

- Log Transfer Interface (LTI) – processes change-set data from the Log Reader, generates Log Transfer Language (LTL) commands, and sends the LTL commands and data to the Replication Server.
- Log Transfer Manager (LTM) – manages all other components, coordinates their operations and interactions, and processes any errors reported by other components.

## Replication Agent System Database

Each Mirror Replication Agent instance uses an embedded database (the Replication Agent System Database, or RASD) to manage its system data repository, which stores information about the primary database schema, mirror log devices, and transaction log metadata.

The system data repository allows Mirror Replication Agent to continue processing transactions from the mirror log devices if the primary database goes offline. The RASD provides database recovery features (such as backup and restore, and software-mirrored devices) to support Mirror Activator for Oracle system recovery and fault tolerance.

When you create a Mirror Replication Agent instance, the RASD is created automatically. The system data repository is populated with the information that the Mirror Replication Agent needs when you *initialize* the Mirror Replication Agent instance.

## Mirror Replication Agent communications

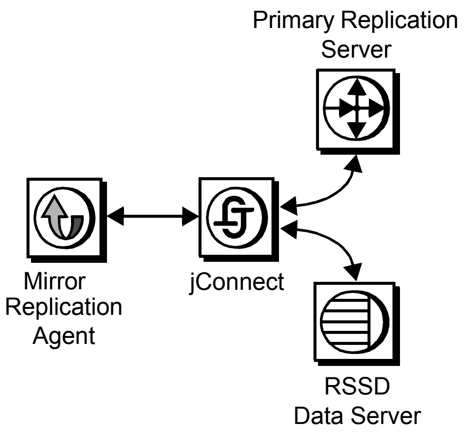
Mirror Replication Agent uses an Oracle-supplied JDBC driver for all communications with Oracle.

Each Mirror Replication Agent instance uses a single instance of the JDBC driver to communicate with an Oracle primary data server. The Sybase jConnect JDBC driver is used to communicate to all Open Client™ and Open Server™ applications.

**Note** Mirror Replication Agent uses file or device I/O for access to the mirror log devices.

Figure 1-4 shows how Mirror Replication Agent communicates with other Mirror Activator for Oracle system components.

***Figure 1-4: Mirror Replication Agent communication***



During normal operation (while replicating transactions), Mirror Replication Agent maintains continuous connections with the following Mirror Activator for Oracle system components:

- Primary database
- Primary Replication Server

Mirror Replication Agent maintains a connection with the primary data server to perform primary database log truncation activities.

Depending on its configuration, Mirror Replication Agent may occasionally connect to the RSSD to retrieve replication definitions, which it can use to process the replicated transactions more efficiently.

## Managing and monitoring Mirror Replication Agent

You can manage and monitor a Mirror Replication Agent instance by logging in to its administration port, using any Open Client application that implements the Sybase Tabular Data Stream™ (TDS) protocol, such as the interactive SQL utility isql, or SQL Advantage®.

# CHAPTER 2 Setup and Configuration

This chapter describes how to set up and configure the Mirror Replication Agent component, and other software components of the Mirror Activator for Oracle system.

**Note** The procedures in this chapter assume you have already installed Mirror Activator for Oracle, DirectConnect for Oracle, and Replication Server software, as described in the Mirror Activator for Oracle installation guide, the DirectConnect for Oracle installation guide, and the Replication Server installation and configuration guides for your platform.

## Setting up the Mirror Activator for Oracle system

To set up the Mirror Activator for Oracle system, you must complete the following high-level tasks:

- Install and configure Mirror Activator for Oracle system components

- Materialize and set up synchronous replication to devices at the standby site:
    - For a new Mirror Activator for Oracle system, you must materialize the standby database (data and log devices) and the mirror log devices, and you must set up synchronous replication to the mirror log devices.

When these tasks are complete, you can start replication with the Mirror Activator for Oracle system.

**Note** The setup tasks for some components of the Mirror Activator for Oracle system require the primary database to be quiesced. To minimize primary database downtime, refer to the appropriate checklist and plan your setup procedures carefully. If you are setting up a new Mirror Activator for Oracle system, see Table 2-1 on page 21.

## Installation and basic configuration

For information about installing Mirror Activator for Oracle system components and setting them up with a rudimentary configuration, refer to the documentation for each system component.

### Sybase software

Sybase provides the following documentation for Mirror Activator for Oracle system components:

- Mirror Activator for Oracle *Installation Guide*
- Replication Server installation and configuration guides for your platform
- Enterprise Connect Data Access *Installation Guide* (Oracle installation)

The Replication Server configuration guide for each platform describes the basic configuration required for all Replication Server installations, which is not covered in this document.

### Disk replication system and devices

Sybase does not provide hardware or software for disk replication systems. For information about installing and configuring the disk replication system and its related devices, refer to the documentation provided by the disk replication system and/or device component vendor(s).

**Data servers and databases**

An existing data server and database (the primary database) is obviously a prerequisite of the Mirror Activator for Oracle system. Therefore, the tasks associated with installing, configuring, and managing data servers, and designing, creating, and managing databases are not covered in this document.

For information about data servers and databases, refer to Oracle documentation.

## Mirror Activator for Oracle configuration requirements

This section describes general configuration requirements, for each component, for systems.

For detailed setup and configuration procedures:

- See “Setting up Mirror Replication Agent connectivity” on page 68 to configure the Mirror Replication Agent connection parameters.
- See “Setting up a new Mirror Activator for Oracle system” on page 19 to set up the primary and standby databases, Mirror Replication Agent, and Replication Server in a *new* Mirror Activator for Oracle system.
- Refer to the documentation provided by your disk replication system vendor (and/or device vendor) to set up and configure the disk replication system and mirror log devices.

**Primary database**

The primary database must be configured as follows:

- Mirror Replication Agent user login name added to the primary database, and the user must be granted appropriate permission to be able to perform task necessary to support replication. This user must also have read access to the mirrored redo log device
- Maintenance User login name (as specified in the Replication Server create connection command) added to the primary database
- Maintenance User login name added to the primary database
- Configure to archive the log with automatic log archiving disabled
- Configure to enable supplemental logging

**Disk replication system**

The disk replication system must be configured as follows:

- Ready to copy a snapshot image of all primary database data and log devices to the standby database data and log devices, if disk replication will be used to materialize the standby database for a new Mirror Activator for Oracle system

- Ready to copy a snapshot image of all standby database data and log devices to the primary database devices, if disk replication will be used to materialize the primary database for failback
- Ready to begin synchronous replication of all primary log devices to the mirror log devices at the standby site

#### Mirror log devices

The mirror log devices at the standby site must be configured as follows:

- Ready to receive a snapshot image from the disk replication system to materialize the mirror log devices from the primary database log devices
- Ready to receive synchronous replication (or mirroring) of the primary log devices from the disk replication system
- Mirror Replication Agent allowed local, read-only access at the standby site during synchronous replication from primary log devices

#### Mirror Replication Agent

The Mirror Replication Agent instance must be configured as follows:

- Connection configuration set correctly for network communications with the primary database, Replication Server, and RSSD
- Initialized using the pdb_init command to validate the primary database is prepared for replication.
- Initialized using the ra_init command to set up the RASD
- Paths correctly defined for access to all mirror log devices

The pdb_init command will validate the primary database as well as setup the Replication System Agent Object table in the primary to support procedure replication.

#### Replication Server

The Replication Server must be configured as follows:

- Mirror Replication Agent user login name, with connect source permission granted
- Mirror Replication Agent user login name for the RSSD identified or created.
- Database Replication Definition and Subscription defined for the primary and standby database.
- The Heterogeneous Datatype Support Scripts must be applied.

#### Standby database

The following list describes the *minimum* standby database configuration required to participate in the Mirror Activator for Oracle system:

- All data server options and configuration parameters *exactly match* those of the primary data server.
- The database name, data and log device configuration, and all database options *exactly match* those of the primary database.
- Maintenance User login name (as specified in the Replication Server create connection command) added to the standby database, with the required permissions granted.
- Replication Server database objects (tables and procedures) created and set up in the standby database, with appropriate permissions granted to the Maintenance User name in the standby database.
- DDL user login name added to the standby database and granted permissions required to execute replicated DDL commands.

Materialization populates the standby database with all of the contents of the primary database, including the Maintenance User login name and Replication Server database objects (which are required in the primary database). If you use snapshot materialization for the standby database, its name, device configuration, and options must exactly match those of the primary database.

## Setting up a new Mirror Activator for Oracle system

This section describes setup and configuration tasks for the following Mirror Activator for Oracle system components:

- Primary database
- Mirror Replication Agent
- Replication Server
- ECDA for Oracle
- Standby database

Table 2-1 provides a checklist of the tasks required to configure all of the software components and set up a new Mirror Activator for Oracle system for replication.

The checklist in Table 2-1 assumes that:

- You will perform a snapshot materialization of the standby database, using the disk replication system facilities. See Appendix A, “Materializing Databases,” for more information.

- You have installed all Mirror Activator for Oracle components.
- You have already completed all of the tasks described in “Setting up Mirror Replication Agent connectivity” on page 68.

When setting up a new Mirror Activator for Oracle system, you must perform the tasks in Table 2-1 in the order they are shown. Deviating from this sequence may produce undesired results, requiring the process to be repeated.

***Table 2-1: Setup and configuration for a new Mirror Activator for Oracle system***

| Task | Description |
|---|---|
| 1 | Set up and enable the disk replication system for synchronous replication to the mirror log devices. |
| 2 | Set up the disk replication system for snapshot replication to support materialization. |
| 3 | Set up the Mirror Activator for Oracle system Maintenance user in the primary data server and in the primary database. |
| 4 | Set up the Mirror Activator for Oracle DDL user in the primary data server. |
| 5 | Set up the Replication Server database objects in the primary database. |
| 6 | Set up the Replication Server database objects in the RSSD. |
| 7 | Set up the ECDA for Oracle to the standby database. |
| 8 | Add database connections and Database Replication Definitions and Subscriptions for the primary and standby databases. |
| 9 | Create and configure a Mirror Replication Agent instance. |
| 10 | Initialize the primary database using the Mirror Replication Agent pdb_init command. |
| 11 | Shut down the standby data server.<br><br>**Note** This step is required only when you use snapshot materialization for a standby database. |
| 12 | Quiesce the primary database to suspend update activity. |
| 13 | Materialize the standby database data and log devices and the mirror log devices at the standby site. Retain synchronous replication to the mirror log devices. |
| 14 | Initialize the Mirror Replication Agent using the ra_init command, and set the paths to the mirror log devices.<br><br>**Note** You can initialize the Mirror Replication Agent concurrently with materialization. |
| 15 | Resume update activity on the primary database, after the device materialization and Mirror Replication Agent initialization are complete. |
| 16 | If you use snapshot materialization for a standby database, start the standby data server. |
| 17 | Resume the Mirror Replication Agent to put it in Replicating state, and resume the standby database connection in the Replication Server. |

The following sections contain detailed procedures for each setup and configuration task.

## Set up synchronous replication of log devices

To provide continuous synchronous mirroring of your primary database logs to the standby site, work with your disk replication vendor. The Mirror Replication Agent will require read access to the mirror files while synchronous replication is occurring.

**Note** Not all disk replication vendors support mounting and access to the mirror files during synchronous replication. Ensure your vendor provides this capability.

## Set up snapshot replication for materialization

If using disk replication to materialize your standby database, work with your disk replication vendor to provide on-demand or "snapshot" copies of your primary database and log files. You will also need to work with your primary database vendor to determine exactly what files and steps must be taken to successfully execute the standby database from snapshot file copies.

**Note** The snapshot log file copies *cannot* be the same target files as the synchronous mirror of the log files used for on-going replication. Once copied, the snapshot files at the standby site will be owned, and written to, by the standby database. The synchronous mirror of the log files must continue to be active copies of the primary database log files.

## Set up the Maintenance User in the primary database

Setting up the Maintenance User involves:

- Adding the Maintenance User login to the primary data server
- Grant appropriate permissions to the Maintenance User login

**Note** Setting up the maintenance user in the primary database assumes snapshot replication copies this user to the standby database along with the data. If you are using a different materialization technique, the maintenance user may need to be added directly to the standby database. Regardless of the materialization technique, the maintenance user must exist and have the appropriate permissions at the standby database after materialization occurs.

To set up the Maintenance User in the primary database repeat the steps for creating a primary database user and contact your DBA or refer to the Primary database vendor documentation.

## Set up the DDL User in the primary database

❖ **To set up the DDL user in the primary database**

1. Add the DDL login to the primary data server
2. Grant appropriate permissions to the DDL User login to execute any DDL command to be replicated.
3. The DDL user must be a different user than the maintenance user

---

**Note** Setting up the DDL user in the Primary database assumes snapshot replication will copy this user to the standby database along with the data. If you are using a different materialization technique, the DDL user may need to be added directly to the standby database. Regardless of materialization technique, the DDL user must exist and have the appropriate permissions at the standby database after materialization occurs.

---

## Set up Replication Server database objects

Replication Server requires a few database objects (procedures and tables) in the primary and standby databases, so that it can:

- Place markers in the database transaction log
- Keep track of successfully replicated transactions
- Replicate stored procedures
- Replicate sequences

You need not set up Replication Server objects in the standby database, because when you materialize the standby database, Replication Server objects in the primary database are copied to the standby database.

To set up Replication Server objects in the primary database, use the Replication Server setup for replicate script for the database you are replicating to. The primary database installation script is a SQL script named as follows: *$SYBASE/REP-12_6/scripts/hds_oracle_setup_for_replicate.sql*

**Note** The scripts in the Replication Server directory have not been updated for this release. Apply the following scripts from the Mirror Replication Agent installation instead:
*$SYBASE/MRO-12_6/scripts \hds_oracle_new_setup_for_replicate.sql*
*$SYBASE/MRO-12_6/scripts\oracle_create_replicate_sequence_proc.sql*

### ❖ To set up Replication Server objects in the primary database

1. Open an operating system command prompt window on the primary data server host machine.
2. At the operating system command prompt, invoke the isql utility to execute the primary database installation script in the primary database.
   - On Microsoft Windows platforms, enter:

     ```
     isql -Usa -Ppwd -Spds -Dpdb -i rsinspri.sql
     ```

     where:

     - *sa* is the System Administrator user login on the primary data server.
     - *pwd* is the password for the System Administrator user login.
     - *pds* is the name of the primary data server.
     - *pdb* is the name of the primary database.
   - On UNIX platforms, enter:

```
isql -Usa -Ppwd -Spds -Dpdb -i rs_install_primary.sql
```

     where:

     - *sa* is the System Administrator user login on the primary data server.
     - *pwd* is the password for the System Administrator user login.
     - *pds* is the name of the primary data server.
     - *pdb* is the name of the primary database.

3 Log in to the primary database with system administration privileges.

4 Grant permissions on the Replication Server objects in the primary database:

```
grant all on rs_lastcommit to mro_maint
grant execute on rs_get_lastcommit to mro_maint
grant execute on rs_update_lastcommit to public
grant execute on rs_check_repl_stat to public
grant execute on rs_marker to public
```

where:

- *mro_maint* is the name of the Maintenance User in the primary database.

## Set up the Replication Server database objects in the RSSD

Replication Server requires changes to the RSSD to support Oracle datatypes.

To set up Replication Server changes in the RSSD, execute the following

Replication Server scripts against the RSSD database:

```
$SYBASE/REP-12_6/scripts/hds_oracle_udds.sql

$SYBASE/REP-12_6/scripts/hds_oracle_funcstrings.sql

$SYBASE/REP-12_6/scripts/hds_clt_ase_to_oracle.sql

$SYBASE/REP-12_6/scripts/hds_clt_oracle_to_ase.sql
```

**Note** In each script, you must edit the "use RSSD" statement to point to the correct RSSD for your Replication Server.

The scripts in the Replication Server directory have not been updated for this release. Apply the following additional scripts from the Mirror Replication Agent installation to your RSSD:

```
$SYBASE/MRO-12_6/scripts/hds_oracle_new_udds.sql.
```

## Set up the ECDA for Oracle to the standby database

If you have not done so already, install ECDA for Oracle and configure a service to the standby database.

## Add databases to Replication Server

You must create connections in the Replication Server for the primary and standby databases.

**Note** You must have "sa" permission in the Replication Server to perform this procedure.

### ❖ To create database connections in Replication Server

1. Log in to the Replication Server with a user login that has "sa" permission.
2. Create a database connection for the primary database:

   ```
   create connection to pds.pdb
      set error class oracle_error_class
      set function string class
         rs_oracle_function_class
      set username "mro_maint"
      set password  "mro_maint_pwd"
      with log transfer on, dsi_suspended
   ```

   where:

   - *pds* is the name of the primary data server.
   - *pdb* is the name of the primary database.
   - *mro_maint* is the Maintenance User login for the primary database.
   - *mro_maint_pwd* is the password for the Maintenance User.

   **Note** The pds and pdb values are usually server host name and Oracle SID.

   If the oracle_error_class does not exist in your Replication Server, you can create it using the following scripts from the Mirror Replication Agent installation:

   Apply script *$SYBASE/MRO-12_6/scripts /oracle_create_error_class_1_rs.sql* to Replication Server

   Apply script *$SYBASE/MRO-12_6 /scripts /oracle_create_error_class_2_rssd.sql* to the RSSD

   Apply script *$SYBASE/MRO-12_6 /scripts /oracle_create_error_class_3_rs.sql* to Replication Server

3 Create a database connection for the standby database:

```
create connection to sds.sdb
   set error class oracle_error_class
   set function string class
      rs_oracle_function_class
   set username "mro_maint"
   set password  "mro_maint_pwd"
```

where:

- *sds* is the name of the standby data server.
- *sdb* is the name of the standby database.
- *mro_maint* is the Maintenance User login for the standby database.
- *mro_maint_pwd* is the password for the Maintenance User.

---

**Note** NOTE: If the oracle_error_class does not exist in your Replication Server, you can create it using the following scripts from the Mirror Replication Agent installation instead:

Apply script *$SYBASE/MRO-12_6 /scripts /oracle_create_error_class_1_rs.sql* to Replication Server

Apply script *$SYBASE/MRO-12_6 /scripts /oracle_create_error_class_2_rssd.sql* to the RSSD

Apply script *$SYBASE/MRO-12_6 /scripts /oracle_create_error_class_3_rs.sql* to Replication Server

---

4 Add Database Replication Definition and Subscription. Replication Server requires a Database Replication Definition to identify the database objects that are to be replicated from the Primary database. A Database Subscription identifies the standby database that will receive the replicated transactions.

To create database replication definitions in Replication Server:

1 Log in to the Replication Server with a user login that has “sa” permission.

2 Create a database replication definition for the primary database:

```
create database replication definition {pds}_repdef

    with primary at {pds}.{pdb}
```

```
        replicate DDL
go
```

where:

- pds is the name of the primary data server.
- pdb is the name of the primary database.

To create database replication subscription in Replication Server:

1. Log in to the Replication Server with a user login that has “sa” permission.
2. Create a database replication definition for the standby database:

```
create subscription {pds}_sub
        for database replication definition {pds}_repdef
        with primary at {pds}.{pdb}
        with replicate at {sds}.{sdb}
        without materialization
go
```

where:

- *pds* is the name of the primary data server.
- *pdb* is the name of the primary database.
- *sds* is the name of the standby data server.
- *sdb* is the name of the standby database.

**Note**

- Replication Server requires a unique function replication definition to support Oracle sequence replication.
- The scripts in the Replication Server directory have not been updated for this release. Apply the following additional script from the Mirror Replication Agent installation to your Replication Server, after editing the script to include your primary database connection name:

```
$SYBASE/MRO-12_6/scripts
/oracle_create_rs_sequence_repdef.sql
```

## Create and configure a Mirror Replication Agent instance

Using the sections in the remainder of this chapter and those in Chapter 3, create and configure a Mirror Replication Agent at the standby site.

## Initializing the Mirror Replication Agent transaction log

Mirror Replication Agent uses the native transaction log maintained by the primary database to obtain transactions.

**Note** The Mirror Replication Agent also creates a few user tables in the primary database to support its operation.

Before you initialize a Mirror Replication Agent, the primary database must be quiesced. The following procedure includes that quiesing.

### Specifying the transaction log object name prefix

**Note** You must have set up connectivity between the Mirror Replication Agent instance and the following Mirror Activator for Oracle system components:

- Primary data server
- Replication Server
- RSSD

In addition, primary databases require you to perform specific setup tasks *before* you can initialize the Mirror Replication Agent transaction log. See Appendix C, “Mirror Replication Agent and Oracle Databases,” to verify that the required setup tasks have been performed for your primary database.

Before you create the Mirror Replication Agent transaction log base objects, you can specify the object name *prefix* string that will be used to name transaction log objects. You can set this prefix string to avoid conflicts with the names of existing database objects in your primary database.

The value of the pdb_xlog_prefix parameter is the prefix string used in all Mirror Replication Agent transaction log component names. Use the ra_config command to change the value of the pdb_xlog_prefix parameter.

**Note** Mirror Replication Agent uses the value of pdb_xlog_prefix to find its transaction log objects in the primary database. If you change the value of pdb_xlog_prefix after you create the transaction log, Mirror Replication Agent will not find the transaction log objects that use the old prefix.

### ❖ To initialize a Mirror Replication Agent transaction log

1. Log in to the Mirror Replication Agent administration port.
2. To define a prefix that uniquely identifies the Mirror Replication Agent transaction log objects that you are creating, use the following command:

   ```
   ra_config pdb_xlog_prefix, string
   ```

   where *string* is a character string of one to three characters that will be used as a prefix for all database object names of the Mirror Replication Agent transaction log components created in the primary database.

   **Note** The default value of the pdb_xlog_prefix parameter is ra_. Unless this string poses a conflict with existing database object names in your primary database, you should use the default value.

3. To initialize the transaction log, use the following command:

   ```
   pdb_init
   ```

   **Note** By default, ra_init will load the timezone information from Oracle's Home *installation/oracore/zoneinfo/timezone.dat*. If the file is not accessed by Mirror Replication Agent, you will need to configure the *pdb_timezone_file* configuration property and specify the location of the *timezone.dat file*. For more information, see Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

   When you invoke the pdb_init command, the Mirror Replication Agent validates required settings in the primary database are correctly set to support replication, and creates objects in the primary database.

   The pdb_init command validates that the following settings are true in the primary database:

   - Archiving of redo logs is enabled.
   - Automatic archiving of redo logs is disabled.
   - Supplemental logging of primary key and unique indexes is enabled.

   If any of these settings is not set, the pdb_init command returns an error. The following Oracle SQL commands can be used to manually validate the settings:

   - The result from the following query should be "ARCHIVELOG" when archiving of redo logs is enabled:

```
select log_mode v$database
```

- The result from the following query should be "FALSE" when automatic archiving of redo logs is disabled:

  ```
  select value from v$system_parameter where name
  = "log_archive_start"
  ```

- The results from the following query should both be "TRUE" when supplemental logging of primary key and unique indexes is enabled:

  ```
  select supplemental_log_data_pk,
  supplemental_log_data_ui from v$database
  ```

If any adjustment to these settings is required, follow the instructions in the Oracle documentation for your database version on the proper commands to make these changes permanent in your environment.

**Note** Transaction log base objects must be created before any objects can be marked for replication in the primary database.

4 To verify that the Mirror Replication Agent transaction log was created, use the following command:

```
ra_helpsysinfo
```

When you invoke the ra_helpsysinfo command, Mirror Replication Agent returns a list of the transaction log base objects in the primary database if initialization completed successfully. If no information is returned, the transaction log does not exist in the primary database.

## Shut down the standby data server

See Appendix A, "Materializing Databases," for more information about materialization procedures for the standby database.

**Note** You must have a System Administrator user role in the standby database to perform this procedure.

❖ **To shut down the standby data server**

1 Log in to the standby database with a System Administrator user role.

2 Shut down the standby data server:

```
shutdown
```

## Quiesce the primary database

You must quiesce the primary database to suspend update activities until the standby database is materialized and the Mirror Replication Agent is initialized.

Any activity that would change the schema in the Primary database should not be allowed to take place while Replication Agent is initializing. Select a time when there are no users are on the system or quiesce the primary database.

## Materialize the standby database

Use the disk replication system facilities to perform the following operations:

- Materialize the standby data devices with a snapshot of the primary data devices
- Materialize the standby log devices with a snapshot of the primary log devices
- Materialize the mirror log devices with a snapshot of the primary log devices
- Configure the disk replication system to mirror (synchronously replicate) all changes on the primary log devices to the mirror log devices

See Appendix A, "Materializing Databases," for more information.

Refer to the documentation provided by your disk replication system vendor (and/or device vendor) for information about configuring the disk replication system and mirror log devices.

## Initialize the Mirror Replication Agent

You must initialize the Mirror Replication Agent instance to populate the RASD with the information it needs about the primary database schema and transaction log devices.

**Note** This procedure requires the primary database to be quiesced. You can initialize the Mirror Replication Agent concurrently with the standby database materialization.

❖ **To initialize the Mirror Replication Agent instance**

1 Log in to the Mirror Replication Agent administration port.

2 Initialize the Mirror Replication Agent instance:

```
ra_init
```

After you initialize the Mirror Replication Agent instance, you may need to alter the log device path(s) returned by the primary data server during initialization, so that the Mirror Replication Agent can access the mirror log devices.

To determine if you need to alter any default log device path, compare the path returned by the primary data server for each primary log device with the path for the corresponding mirror log device:

- Use the Mirror Replication Agent ra_helpdevice command to view the log device path(s) returned by the primary data server during initialization.
- If necessary, use the Mirror Replication Agent ra_devicepath command to alter the default log device path to point to the corresponding mirror log device.

See the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual* for more information about the ra_devicepath and ra_helpdevice commands.

When the Mirror Replication Agent is initialized, you can put the Mirror Replication Agent instance in Replicating state. For more information about putting the Mirror Replication Agent in Replicating state see Chapter 3, “Administering Mirror Replication Agent,” section “Changing the Mirror Replication Agent state.”

## Resume update activity on the primary database

After the standby database materialization and Mirror Replication Agent initialization are complete, release the quiesce to resume update activity on the primary database.

You must *not* resume update activity on the primary database until all of the following operations are complete:

- All standby database data and log devices are materialized.
- All mirror log devices are materialized.
- The disk replication system is configured for synchronous replication from the primary log devices to the mirror log devices.
- The Mirror Replication Agent is initialized, with correct paths defined for all mirror log devices.

## Start the standby database

**Note** Issue the appropriate startup command for their specific database instance. For example, startup after logging in with a sysdba role.

See Appendix A, "Materializing Databases," for more information about materialization procedures for the standby database.

## Resume the Mirror Replication Agent

You must resume the Mirror Replication Agent instance to put it in *Replicating* state, so that it can read the mirror log devices and send replicated transactions to the Replication Server.

❖ **To resume the Mirror Replication Agent**

1 Log in to the Mirror Replication Agent administration port.

2 Start replication in the Mirror Replication Agent:

```
resume
```

3 Verify that the Mirror Replication Agent instance is in *Replicating* state:

```
ra_status
```

If the Mirror Replication Agent instance is not in *Replicating* state after you invoke the resume command, see Chapter 4, "Troubleshooting Mirror Replication Agent," for more information.

## Resume the standby database connection

To start Mirror Activator for Oracle replication, you must resume the Replication Server connection to the standby database.

**Note** You must have "sa" permission in the Replication Server to perform this procedure.

❖ **To resume the Replication Server standby database connection**

1 Log in to the Replication Server with "sa" permission.

2 Resume the standby database connection to start replication:

```
resume connection to sds.sdb
```

where:

- *sds* is the name of the standby data server.
- *sdb* is the name of the standby database.

If Replication Server drops the standby database connection or fails to begin replication after you invoke the resume connection command, see Chapter 4, “Troubleshooting Mirror Replication Agent,” or refer to the Replication Server *Troubleshooting Guide* for more information.

**Note** The Mirror Replication Agent instance will go to the *Replicating* state only if a connection for the primary database has been created in the primary Replication Server. For more information on creating the primary database connection in Replication Server, see Appendix C, “Mirror Replication Agent and Oracle Databases.”

## The Mirror Replication Agent instance

After you install the Mirror Replication Agent software, you must create one instance of the Mirror Replication Agent for each primary database that you want to replicate transactions from.

Each Mirror Replication Agent instance is an independent process, with its own instance directories to house its configuration file, system log files, and Replication Agent System Database (RASD). Each Mirror Replication Agent instance manages its own connections to the primary data server, mirror log devices, Replication Server, and RSSD.

When you create a Mirror Replication Agent instance, you must specify:

- A unique instance (server) name
- A unique client socket port number for its administration port

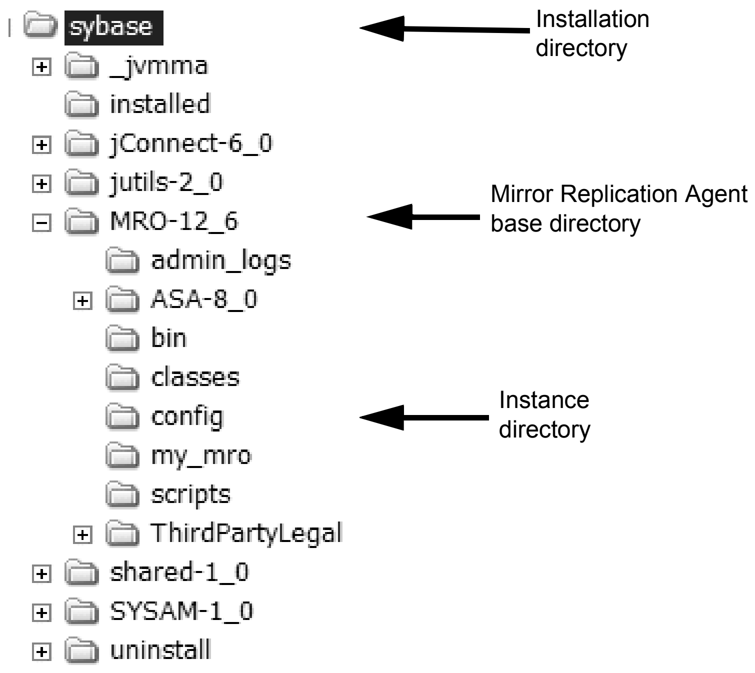
You can create and run more than one Mirror Replication Agent instance on a single host machine, but each instance must have a unique name and a unique port number.

For more information see “Creating a Mirror Replication Agent instance” on page 42.

## Mirror Replication Agent instance directories

Figure 2-1 shows an example of the Mirror Replication Agent instance directories, which are created under the Mirror Replication Agent base directory when you create a Mirror Replication Agent instance.

***Figure 2-1: Mirror Replication Agent directories***



The Mirror Replication Agent base directory (*MRO-12_6*) and the installation directory (*sybase*) are created when you install the Mirror Replication Agent software.

---

**Note** A single installation (on a single host machine) can support multiple Mirror Replication Agent instances. Each instance directory resides under the Mirror Replication Agent base directory created when you install the software.

---

## Using Mirror Replication Agent utilities

Three utilities are provided with the Mirror Replication Agent:

- **Note** On Windows platforms, when you execute a *run* script, you can omit the extension: mro -i my_mro. However, on UNIX, you must always include the extension: mro.sh -i my_mro.

- mro – starts a Mirror Replication Agent instance, or returns the Mirror Replication Agent software version number.
- mro_admin – allows you to create, copy, verify, and delete Mirror Replication Agent instances, or list all verifiable installed Mirror Replication Agent instances on a machine.

Mirror Replication Agent utilities are supplied as batch files for Microsoft Windows platforms and shell scripts for UNIX platforms. The utility files reside in the *bin* subdirectory, under the Mirror Replication Agent base directory.

You can use the -help option with either the mro_admin or mro command line utility to obtain information about that utility.

For more information about the command line utilities, see "Using the command line interface" on page 38.

### Setting the SYBASE environment

Before you can invoke a Mirror Replication Agent utility, you must:

- Log in to the operating system on the Mirror Replication Agent host machine with a user login that has execute permission in the Mirror Replication Agent installation directory and all subdirectories (for example, the "sybase" user)
- Use the *SYBASE* environment script to verify that the Sybase environment variables are set

The *SYBASE* environment script is supplied as a batch file for Microsoft Windows platforms (*SYBASE.bat*) and a shell script for UNIX platforms (*SYBASE.sh*).

❖ **To set the SYBASE environment**

1. Log in to the operating system on the Mirror Replication Agent host machine with a user login that has the appropriate permissions.
2. Open an operating system command window.
3. At the operating system prompt, navigate to the Mirror Replication Agent installation directory.
   - On Microsoft Windows platforms, enter:

     ```
     cd c:\sybase
     ```

     where *c:\sybase* is the path to the Mirror Replication Agent installation directory.
   - On UNIX platforms, enter:

     ```
     cd /opt/sybase
     ```

     where */opt/sybase* is the path to the Mirror Replication Agent installation directory.
4. In the Mirror Replication Agent installation directory, invoke the *SYBASE* environment script.
   - On Microsoft Windows platforms:

     ```
     SYBASE
     ```
   - On UNIX platforms, for Bourne and Korn shells:

     ```
     . SYBASE.sh
     ```
   - On UNIX platforms for C-shell:

     ```
     source SYBASE.csh
     ```

**Note** On UNIX platforms, you can insert the source SYBASE.sh command in the *.login* file for the Mirror Replication Agent administrator (or “sybase” user), so that the SYBASE environment is set automatically when you log in to the Mirror Replication Agent host machine.

## Using the command line interface

This section describes how to administer a Mirror Replication Agent instance using the command line interface.

## Using the mro utility

The Mirror Replication Agent mro utility provides the following functions:

- Starts a specified Mirror Replication Agent instance
- Returns the Mirror Replication Agent software version number

For information about creating a Mirror Replication Agent instance, see "Creating a Mirror Replication Agent instance" on page 42.

To run the mro utility, invoke it as a command at the operating system prompt.

Syntax

mro [-help | -i *inst_name* [-*state*] | -v]

Parameters

-help

The option that returns command usage information.

**Note** You can also invoke mro with no option specified to return command usage information.

-i *inst_name*

The option that specifies a Mirror Replication Agent instance to start, where *inst_name* is the name of an existing Mirror Replication Agent instance.

-*state*

The keyword that specifies a start-up state for the Mirror Replication Agent instance.

Valid -*state* values are:

- -admin – starts the Mirror Replication Agent instance in *Admin* state. (This is the default start-up state.)
- -replicate – starts the Mirror Replication Agent instance in *Replicating* state.

-v

The option that returns the Mirror Replication Agent software version number.

Example

To start a Mirror Replication Agent instance named "my_mro" in *Replicating* state, enter the following command at the operating system prompt:

```
mro -i my_mro -replicate
```

For more information about starting a Mirror Replication Agent instance, see "Starting the Mirror Replication Agent" on page 58.

For more information about *Admin* and *Replicating* states, see "Understanding Mirror Replication Agent states" on page 86.

**Start-up errors**

If the Mirror Replication Agent instance encounters start-up errors:

- On Microsoft Windows platforms, start-up errors are displayed in the operating system command window.
- On UNIX platforms, start-up errors are displayed in the operating system command window and recorded in the Mirror Replication Agent system log.

For more information, see Chapter 4, "Troubleshooting Mirror Replication Agent."

## Using the mro_admin utility

The Mirror Replication Agent mro_admin utility provides the following functions:

- Creates, copies, deletes, and verifies Mirror Replication Agent instances
- Lists all valid Mirror Replication Agent instances on the Mirror Replication Agent host machine
- Returns the path of the Mirror Replication Agent installation directory
- Creates Mirror Replication instances from parameters in a response file.

To run the mro_admin utility, invoke it as a command at the operating system prompt.

Syntax

mro_admin [option [create options]] [*inst_name*]

**Note** You can also invoke mro_admin with no option specified to return command usage information.

Parameters

-b

The option that returns the complete path of the Mirror Replication Agent installation directory.

-c *inst_name*

The option that creates a new Mirror Replication Agent instance using the specified name (*inst_name*).

The *inst_name* string must be a valid server name, and unique on the host machine.

When you use the -c option, the following options are required:

- -p, or
- -p and -f

When you use the -f option the primary database type specified for the existing Mirror Replication Agent instance is copied to the configuration of the new Mirror Replication Agent instance.

-p *port_num*

The option that specifies a client socket port number for the administration port of the Mirror Replication Agent instance.

The *port_num* must be a valid port number, and unique on the Mirror Replication Agent host machine.

When the -c option is used, you also have the option of specifying that the configuration of the new Mirror Replication Agent instance should be based on the configuration file for an existing Mirror Replication Agent instance. To do this, use the -f option.

-f *old_inst*

The option that copies the configuration of an existing Mirror Replication Agent instance for a new Mirror Replication Agent instance.

The *old_inst* string is the name of the existing Mirror Replication Agent instance whose configuration you want to copy for the new Mirror Replication Agent instance.

**Note** When you use the -f option, some configuration parameters are set to default values. For more information, see "Copying a Mirror Replication Agent configuration" on page 52.

-d *inst_name*

The option that deletes a specified Mirror Replication Agent instance.

The *inst_name* string must be the name of an existing Mirror Replication Agent instance.

When you invoke mro_admin with the -d option, the utility deletes all of the subdirectories associated with the specified instance from the Mirror Replication Agent installation directory.

---

**Note** On Microsoft Windows platforms, if any application is accessing a file or directory associated with a Mirror Replication Agent instance when you delete the instance, the open file or directory is *not* deleted. An error message informs you of the file or directory not deleted.

---

To finish deleting a Mirror Replication Agent instance after a file or directory access conflict on a Microsoft Windows platform, you must:

- Verify that the file or directory is not open in any application
- Manually delete the file or directory

-h

The option that returns command usage information.

-l (lowercase *L*)

The option that lists all verifiable Mirror Replication Agent instances.

-v *inst_name*

The option that verifies the complete directory structure for a specified Mirror Replication Agent instance.

The *inst_name* string must be the name of an existing Mirror Replication Agent instance.

-v *inst_name*

The option that verifies the complete directory structure for a specified Mirror Replication Agent instance.

The *inst_name* string must be the name of an existing Mirror Replication Agent instance.

## Creating a Mirror Replication Agent instance

You can create a Mirror Replication Agent instance at any time after the Mirror Replication Agent software is installed by invoking mro_admin with the -c option or using the Administrator GUI utility.

The complete syntax is:

```
mro_admin -c new_inst -p port_num {-f old_inst}
```

where:

- *new_inst* is the name of the new Mirror Replication Agent instance you are creating.
- *port_num* is the client socket port number for the administration port of the new Mirror Replication Agent instance.
- *old_inst* is the name of an existing Mirror Replication Agent instance whose configuration you want to duplicate for the new Mirror Replication Agent instance.

You must specify the -f option to copy an existing configuration.

For information about creating a Mirror Replication Agent instance based on the configuration of an existing instance, see "Copying a Mirror Replication Agent configuration" on page 52.

## Creating Mirror Replication instances using resource files

The mro_admin utility provides two command-line parameters that support creating a Mirror Replication Agent instance using a resource file, and validating resource files.

Syntax

```
mro_admin [-vr res_file | -r res_file]
```

Parameters

-vr *res_file*

Validates the specified resource file (*res_file*), without creating a Mirror Replication Agent instance or making any change in the environment.

-r *res_file*

Creates a Mirror Replication Agent instance, based on the contents of the specified resource file (*res_file*).

A *resource file* is an ASCII text file that contains configuration information for the Mirror Replication Agent instance to be created by the mro_admin utility.

The mro_admin parameters in the resource file allow you to specify the following options, in addition to creating a Mirror Replication Agent instance:

- Create the instance user login in the primary data server, and grant all required permissions.
- Start the new instance after it is created.
- Initialize the new instance after it starts.

- Record mirror log device information in the log device repository after the instance is initialized.

---

**Note** When you *validate* a resource file with mro_admin -vr, no other action is taken, and no Mirror Replication Agent instance is created.

---

The following sections describe how to use the new mro_admin features:

- Creating a new resource file
- Editing a resource file
- Validating a resource file
- Creating an instance with a resource file

## Creating a new resource file

A resource file template, named *mro.rs*, is provided in the *init* subdirectory of the Mirror Replication Agent installation directory. For example:

```
C:\sybase\MRO-12_6\init\mro.rs
```

The resource file template contains comments that describe each configuration parameter and its value.

---

**Note** Sybase recommends that you validate each resource file *before* you create a Mirror Replication Agent instance using that resource file.

---

### ❖ To create a resource file

1. Copy the resource file template *mro.rs* to another file that you will edit to create the new resource file. For example:

   ```
   cp mro.rs pubs2.rs
   ```

   where *pubs2.rs* is the name of the new resource file you want to create.

   If you have an existing resource file, you can copy that file to create a new resource file, instead of copying the template.

2. Use your preferred text editor to edit the resource file copy that you created.

After you create a new resource file, you should validate it. For more information, see “Validating a resource file” on page 45.

### Editing a resource file

The mro_admin resource file is an ASCII text file, that you can edit using any standard text editor.

Resource file contents must conform to the following:

- Configuration parameters for both the Mirror Replication Agent and the mro_admin utility must use the following format:

  ```
  param=value
  ```

  where:

  - *param* is the name of the configuration parameter.
  - *value* is the value of the configuration parameter.

  **Note** Spaces are not allowed before or after the = symbol, or within the *value* string.

- Each ***param=value*** statement must occur on a separate line.
- If a default value exists for a configuration parameter, you can specify the default value with the string `USE_DEFAULT`, as follows:

  ```
  param=USE_DEFAULT
  ```

  where *param* is the name of the configuration parameter.

- The following mro_admin configuration parameters require a value of yes or no:

  - create_pds_username
  - start_instance
  - initialize_instance

  The yes value is not case-insensitive. Any string other than `[y|Y][e|E][s|S]` is interpreted as no.

**Note** Blank lines and lines that begin with the # symbol are ignored in the resource file.

### Validating a resource file

When you invoke the mro_admin utility with the -vr option, the utility validates the specified resource file and returns information about the validation process.

The mro_admin utility validates resource files by:

- Verifying uniqueness of the Mirror Replication Agent administration port number and instance name.
- Verifying access to the primary data server, Replication Server®, and RSSD.
- Verifying the host name, port number, database name, user login, and password on each server.
- Verifying the Replication Server database connection for the primary database.
- Verifying that the *pds_username* user has all the required permissions at the primary database.
- Verifying that the primary database redo logs are correctly configured.
- Verifying access to mirror log devices, if specified in the resource file.

If any validation fails, the mro_admin utility returns an error message and information about the failure.

You can repeat the validation process as many times as necessary. No entities are changed or created as a result of this process.

**Note** Sybase recommends that you validate a new resource file *before* you create a Mirror Replication Agent instance using the new resource file.

## ❖ To validate a resource file

1. Invoke the mro_admin utility, specifying the -vr option and the name of the resource file:

   ```
   mro_admin -vr res_file
   ```

   where *res_file* is the name of the resource file you want to validate.

   For example, if the resource file is named *pubs2.rs*, enter the following at the command prompt:

   ```
   mro_admin -vr pubs2.rs
   ```

   Validation results are returned as either:

   - `Response_file processing completed.`

     or

   - `Response_file processing completed with errors.`

If the validation is successful, you can skip step 2, and use the resource file to create a Mirror Replication Agent instance. For more information, see “Creating an instance with a resource file” on page 47.

If the validation encounters errors, continue to step 2.

2. Use the following procedure to correct validation errors:

   a. Review the error messages to determine the cause of the failure.

   b. Edit the resource file to correct the appropriate values.

   c. Invoke mro_admin -vr again, specifying the name of the resource file.

   Repeat this step until the resource file is successfully validated.

### Creating an instance with a resource file

When you invoke the mro_admin utility with the -r option, the utility first validates the specified resource file, as described in “Validating a resource file” on page 45, except:

- If the Mirror Replication Agent primary database user login does not exist in the primary data server, the utility creates it, if specified in the resource file (`create_pds_username=yes`). If the user login does exist in the primary data server but does not have all the required privileges, set create to *yes*, to have the utility grant all required permissions.

  If the Mirror Replication Agent primary database user login does exist in the primary data server, has all the required privileges, and the resource file specifies that it should be created, the utility returns an error message and does not create the instance. (This error would be caught in the validation process described in “Validating a resource file” on page 45.)

- If the resource file specifies that the new Mirror Replication Agent instance should be initialized (`initialize_instance=yes`), then:

  - The Mirror Replication Agent primary database user login must either exist in the primary data server, or be created by the mro_admin utility (`create_pds_username=yes`)

  - The Response file must specify that the Mirror Replication Agent instance should be started (*start_instance=yes*).

  Otherwise, the utility returns an error message and does not create the instance.

After validating the resource file successfully, the mro_admin utility does the following:

- Creates and configures a Mirror Replication Agent instance, based on the contents of the specified resource file.
- Creates or grants all required privileges for the instance user, if specified in the *resource* file.
- Starts the new Mirror Replication Agent instance, if specified in the resource file.
- Initializes the new Mirror Replication Agent instance, if specified in the resource file.
- Records mirror log device information in the log device repository, if specified in the resource file.

The utility also returns information about the instance created and the result.

If instance creation fails, the mro_admin utility returns an error message and information about the failure.

---

**Note** Sybase recommends that you validate a new resource file *before* you create a Mirror Replication Agent instance using the new resource file. For more information, see "Validating a resource file" on page 45.

---

❖ **To create a Mirror Replication Agent instance**

- Invoke the mro_admin utility, specifying the -r option and the name of the resource file:

  ```
  mro_admin -r res_file
  ```

  where *res_file* is the name of the resource file.

  For example, if the resource file is named *pubs2.rs*, enter the following at the command prompt:

  ```
  mro_admin -r pubs2.rs
  ```

  Results are returned as either:

  - `Response_file processing completed.`

    or

  - `Response_file processing completed with errors.`

If the instance creation is successful, you can begin using the new Mirror Replication Agent instance.

If the instance creation fails, you may have to:

- Drop the Mirror Replication Agent user from the primary database.
- Delete all files and subdirectories in the instance directory, and delete the instance directory from the Mirror Replication Agent installation directory.
- Edit the resource file to correct the appropriate values.

---

**Note** If the instance creation fails, use the following recovery procedure *before* you attempt to create the instance again.

---

❖ **To recover from instance creation errors**

1 If the resource file does *not* specify that the instance user login be created in the primary data server, skip this step and continue with step 2.

  If the resource file specifies that the instance user login be created in the primary data server (that is, `create_pds_username=yes`), then:

  a Check the primary database to determine if the instance user was added:

  ```
  select*from dba_users username
  ```

  where username is the *PDS_USERNAME* of the instance user in the primary database.

  - If the instance user was added to the primary database, skip step 1b and continue with step 1c.
  - If the instance user was *not* added to the primary database, continue with step 1b.

  b Check that the *pds_sa_username* has sufficient privileges to create the instance login at the primary database, and has the authority to grant access to various “dollar” tables such as *sys.seq$*.

  c Edit the resource file to specify that the instance user login should not be created in the primary data server (`create_pds_username=no`).

---

**Note** If the Mirror Replication Agent primary database user login is successfully created before the instance creation fails, you must either:

- Edit the resource file to set the value of the create_pds_username parameter to no, or
- Log in to the primary data server and drop the instance login.

---

2 Check the Mirror Replication Agent base directory on the Mirror Replication Agent host to determine if a new instance directory was created. The Mirror Replication Agent base directory is:

```
%SYBASE%\MRO-12_6
```

where *%SYBASE%* is the Mirror Replication Agent installation directory.

If you do *not* find a new instance directory in the Mirror Replication Agent base directory, skip step 3 and continue with step 4.

If you find a new instance directory in the Mirror Replication Agent base directory, continue with step 3.

3 To delete the new instance directory, you have two options:

- Use the mro_admin utility to delete the instance:

  ```
  mro_admin -d inst_name
  ```

  where *inst_name* is the name of the instance you want to delete.

  or

- Use operating system commands to delete all of the files and subdirectories in the new instance directory, and then delete the new instance directory.

4 Review the error messages to find the cause of the instance creation failure, and if necessary, edit the resource file to correct the appropriate values.

After editing the resource file, use mro_admin to validate the resource file:

```
mro_admin -vr res_file
```

where *res_file* is the name of the resource file.

See “Validating a resource file” on page 45 for more information.

After you complete the recovery procedure, you can retry creating the Mirror Replication Agent instance.

## Creating a Mirror Replication Agent using the command line

Use the following procedure to create a Mirror Replication Agent instance using the command line.

**Note** You must set the SYBASE environment before you invoke the Mirror Replication Agent mro_admin utility. For more information see "Setting the SYBASE environment" on page 37.

❖ **To create a Mirror Replication Agent instance using the command line**

1 Open an operating system command window on the Mirror Replication Agent host machine.

2 At the operating system prompt, navigate to the Mirror Replication Agent *bin* directory.

- On Microsoft Windows platforms, enter:

  ```
  cd %SYBASE%\MRO-12_6\bin
  ```

  where *%SYBASE%* is the path to the Mirror Replication Agent installation directory.

- On UNIX platforms, enter:

  ```
  cd $SYBASE/MRO-12_6/bin
  ```

  where *$SYBASE* is the path to the Mirror Replication Agent installation directory.

3 In the Mirror Replication Agent *bin* directory, invoke the mro_admin utility to create a new Mirror Replication Agent instance:

```
mro_admin -c new_inst -p port_num
```

where:

- *new_inst* is the name of the Mirror Replication Agent instance.
- *port_num* is the client socket port number for the administration port of the new instance.

After you invoke mro_admin, the operating system prompt returns when the new Mirror Replication Agent instance is created.

4 Verify that the Mirror Replication Agent instance was created properly using one of the following methods:

- Invoke mro_admin with the -v option, and specify the name of the new Mirror Replication Agent instance:

  ```
  mro_admin -v new_inst
  ```

  where *new_inst* is the name of the new Mirror Replication Agent instance.

  When you verify a Mirror Replication Agent instance with the -v option, the utility verifies the instance by checking for an instance directory with the specified instance name under the Mirror Replication Agent base directory, and checking all of the subdirectories under the Mirror Replication Agent instance directory.

- Invoke mro_admin with the -l option:

  ```
  mro_admin -l
  ```

  The -l option lists all verifiable Mirror Replication Agent instances, which should include the new one you just created.

- As an alternative to using the mro_admin utility, you can use operating system commands to verify that the Mirror Replication Agent instance directories were created correctly (as shown in Figure 2-1).

After you create a Mirror Replication Agent instance, you can use the mro utility to start the instance so that you can administer and configure it. For more information, see “Starting the Mirror Replication Agent” on page 58.

**Note** Sybase recommends that you create a user login name and password to replace the default “sa” login and secure access to the administration port, immediately after you create a Mirror Replication Agent instance. For more information, see “Creating the Mirror Replication Agent administrator login” on page 67.

## Copying a Mirror Replication Agent configuration

When you create a new Mirror Replication Agent instance, you can copy the configuration of an existing instance by invoking mro_admin with the -c option and -f option.

The complete syntax is:

mro_admin -c *new_inst* -p *port_num* [-f *old_inst*]

where:

- *new_inst* is the name of the new Mirror Replication Agent instance.
- *port_num* is the client socket port number for the administration port of the new instance.
- *old_inst* is the name of an existing Mirror Replication Agent instance whose configuration you want to copy for the new instance.

If you do not specify the -f option, the new Mirror Replication Agent instance is created with a default configuration.

For information about creating a Mirror Replication Agent instance with the default configuration, see "Creating a Mirror Replication Agent instance" on page 42.

Use the following procedure to create a new Mirror Replication Agent instance, based on the configuration of an existing instance.

---

**Note** You must set the SYBASE environment before you invoke the Mirror Replication Agent mro_admin utility. For more information, see "Setting the SYBASE environment" on page 37.

---

❖ **To copy an existing Mirror Replication Agent instance configuration to a new instance**

1 Open an operating system command window on the Mirror Replication Agent host machine.

2 At the operating system prompt, navigate to the Mirror Replication Agent *bin* directory.

- On Microsoft Windows platforms:

```
cd %SYBASE%\MRO-12_6\bin
```

  where *%SYBASE%* is the path to the Mirror Replication Agent installation directory.

- On UNIX platforms:

```
cd $SYBASE/MRO-12_6/bin
```

  where *$SYBASE* is the path to the Mirror Replication Agent installation directory.

3 In the Mirror Replication Agent *bin* directory, invoke the mro_admin utility to create a new Mirror Replication Agent instance whose configuration is based on the configuration of an existing instance:

```
mro_admin -c new_inst -p port_num -f old_inst
```

where:

- *new_inst* is the name of the new Mirror Replication Agent instance.
- *port_num* is the client socket port number for the administration port of the new instance.
- *old_inst* is the name of an existing Mirror Replication Agent instance whose configuration you want to copy for the new instance.

After you invoke mro_admin, the operating system prompt returns when the new Mirror Replication Agent instance is created.

4 Verify that the Mirror Replication Agent instance was created properly using one of the following methods:

- Invoke mro_admin with the -v option, and specify the name of the new Mirror Replication Agent instance:

  ```
  mro_admin -v new_inst
  ```

  where *new_inst* is the name of the new Mirror Replication Agent instance.

  When you verify a Mirror Replication Agent instance with the -v option, the utility verifies the instance by checking for an instance directory with the specified instance name under the Mirror Replication Agent base directory, and checking all of the subdirectories under the Mirror Replication Agent instance directory.

- Invoke mro_admin with the -l (lowercase L) option:

  ```
  mro_admin -l
  ```

  The -l option lists all verifiable Mirror Replication Agent instances, which should include the one you just created.

- As an alternative to using the mro_admin utility, you can use operating system commands to verify that the Mirror Replication Agent instance directories were created correctly (as shown in Figure 2-1 on page 36).

---

**Note** When you create a new Mirror Replication Agent instance and copy the configuration of an existing instance, some configuration parameters are set to default values, and they are not copied from the existing configuration.

---

The values of the following configuration parameters are not copied from an existing configuration:

```
admin_port
log_directory
pds_database_name
pds_datasource_name
pds_host_name
pds_password
pds_port_number
pds_retry_count
pds_retry_timeout
pds_server_name
pds_username
rs_source_db
rs_source_ds
rasd_backup_dir
rasd_database
rasd_mirror_tran_log
rasd_trace_log_dir
rasd_tran_log
rasd_tran_log_mirror
asa_port
```

For more information about Mirror Replication Agent configuration parameters, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

After you create a Mirror Replication Agent instance, you can use the mro utility to start the instance so that it can be administered and configured.

---

**Note** Sybase recommends that you create a user login name and password to replace the default “sa” login and secure access to the administration port, immediately after you create a Mirror Replication Agent instance. For more information see “Creating the Mirror Replication Agent administrator login” on page 67.

---

## Deleting a Mirror Replication Agent instance

You can delete a Mirror Replication Agent instance at any time by invoking mro_admin with the -d option.

Before you delete a Mirror Replication Agent instance, you should:

- Shut down the Mirror Replication Agent instance, if it is running. For more information, see “Shutting down the Mirror Replication Agent instance” on page 89.
- If the Mirror Replication Agent software is installed on a Microsoft Windows platform, verify that none of the files in the instance subdirectories are open, and that no application or window is accessing the instance subdirectories.

---

**Note** You must set the SYBASE environment before you invoke the Mirror Replication Agent mro_admin utility. For more information, see “Setting the SYBASE environment” on page 37.

---

❖ **To delete a Mirror Replication Agent instance**

1 Open an operating system command window on the Mirror Replication Agent host machine.

2 At the operating system prompt, navigate to the Mirror Replication Agent *bin* directory.

- On Microsoft Windows platforms, enter:

  ```
  cd %SYBASE%\MRO-12_6\bin
  ```

  where *%SYBASE%* is the path to the Mirror Replication Agent installation directory.

- On UNIX platforms, enter:

  ```
  cd $SYBASE/MRO-12_6/bin
  ```

  where *$SYBASE* is the path to the Mirror Replication Agent installation directory.

3 In the Mirror Replication Agent *bin* directory, invoke the mro_admin utility with the -d option to delete a Mirror Replication Agent instance:

```
mro_admin -d inst_name
```

where *inst_name* is the name of the Mirror Replication Agent instance you want to delete.

The following message appears:

```
Are you sure you want to delete the Mirror
Replication Agent instance inst_name? [y/n]
```

4 Enter y to delete the Mirror Replication Agent instance.

After the instance is deleted, the operating system prompt returns.

If the instance is running when you invoke mro_admin with the -d option, the utility returns an error message:

```
Cannot delete Mirror Replication Agent instance
'inst_name' because it is currently running.
```

To shut down a Mirror Replication Agent instance, log in to its administrative port, and use the shutdown command. For more information, see "Shutting down the Mirror Replication Agent instance" on page 89.

5 Verify that the Mirror Replication Agent instance was deleted properly using one of the following methods:

- Invoke the mro_admin utility with the -v option, and specify the name of the deleted Mirror Replication Agent instance:

  ```
  mro_admin -v inst_name
  ```

  where *inst_name* is the name of the deleted Mirror Replication Agent instance.

  When you verify a Mirror Replication Agent instance with the -v option, the utility looks for an instance directory with the specified instance name under the Mirror Replication Agent base directory, and looks for the correct subdirectories under the Mirror Replication Agent instance directory.

- Invoke the mro_admin utility with the -l option:

  ```
  mro_admin -l
  ```

  The -l option lists all verifiable Mirror Replication Agent instances, which should *not* include the one you just deleted.

- As an alternative to using the mro_admin utility, you can use operating system commands to verify that the Mirror Replication Agent instance directories were deleted correctly. (The Mirror Replication Agent instance directories are shown in Figure 2-1.)

**Note** On Microsoft Windows platforms, if any application is accessing a file or directory associated with a Mirror Replication Agent instance when you delete the instance, the open file or directory is *not* deleted. An error message informs you of the file or directory not deleted.

To finish deleting a Mirror Replication Agent instance after a file or directory access conflict on a Microsoft Windows platform, you must:

- Verify that the file or directory is not open in any application
- Manually delete the file or directory

**Note** If you delete a Mirror Replication Agent instance, Mirror Replication Agent does not unmark any primary database objects marked for replication, nor does Mirror Replication Agent delete its transaction log objects. Before you shut down and delete a Mirror Replication Agent instance, you must unmark primary database objects and delete the transaction log.

## Starting the Mirror Replication Agent

To start a Mirror Replication Agent instance, you must log in to the Mirror Replication Agent host machine with a user name that has execute permission in the Mirror Replication Agent installation directory and all subdirectories (for example, the “sybase” user).

Following are ways you can start a Mirror Replication Agent instance:

- Invoke the mro utility and specify the instance that you want to start.
- Invoke the *RUN* script for the instance that you want to start.

The mro utility and the *RUN* script are batch files on Microsoft Windows platforms and shell scripts on UNIX platforms.

### Start-up requirements

Before you can start a Mirror Replication Agent instance and connect to the primary data server, you must set all required variables.

For more information about connectivity requirements specific to your primary database, see Appendix C, “Mirror Replication Agent and Oracle Databases.”

Setting the CLASSPATH environment variable

Add the location of the JDBC driver for the primary database to the CLASSPATH environment variable.

See Appendix C, "Mirror Replication Agent and Oracle Databases," for more information about installing and setting up the JDBC driver for the primary database and setting up Mirror Replication Agent connectivity.

Setting the Mirror Replication Agent character set to match the primary database

If the character set on your Mirror Replication Agent is different than the one on your primary database, you need to set the RA_JAVA_DFLT_CHARSET environment variable. The Mirror Replication Agent character set must be the same as that of the primary database. For more information, see "Setting character sets" on page 59.

## Setting character sets

In a Mirror Activator for Oracle system, character set conversions might be required if system components reside on more than one type of platform. For example, if the primary data server host does not support all of the same character sets as the Replication Server host, then character set conversion (between compatible character sets on the two platforms) is required.

Character set problems can produce data inconsistencies between the primary database and the standby database. To avoid character set problems, you must either:

- Use the same character set on all servers and platforms in the Mirror Activator for Oracle system, or
- Use compatible character sets on all servers and platforms in the Mirror Activator for Oracle system, and configure the Mirror Activator for Oracle system components to perform the appropriate character set conversions.

Using character set conversions slows performance.

**Note** Sybase recommends that you use the same character set on *all* servers and platforms in a Mirror Activator for Oracle system.

### Configuring your environment's character set

By default, the Java Virtual Machine (JVM) under which a Mirror Replication Agent instance is running finds your system's default character set. The type of character data that Mirror Replication Agent can handle is determined by the character set, also known as the encoding. Unless you want to override the default character set that the JVM finds on your system, you do *not* need to explicitly set the character set-related environment variable.

To support overriding the default character set, all of the executable scripts (or batch files) in the Mirror Replication Agent */bin* directory refer to an environment variable named RA_JAVA_DFLT_CHARSET. You can set this environment variable to use the character set you want. The character set you specify must be the character set configured on the primary database. For a list of valid Java character sets, see Supported Encodings on the Internationalization page under Documentation for the J2SE 1.4.2 JDK at http://java.sun.com/j2se/corejava/intl/index.jsp.

All Mirror Replication Agent instance RUN scripts also reference the RA_JAVA_DFLT_CHARSET environment variable.

**Note** If you are using Replication Server to replicate a number of different character sets, you must configure it for UTF8.

You can override the system default character set by either:

- Setting the value of a system variable named RA_JAVA_DFLT_CHARSET in your environment and using the ra utility to start the Mirror Replication Agent instance, or
- Setting the value of the RA_JAVA_DFLT_CHARSET variable in the Mirror Replication Agent instance RUN script and using the RUN script to start the Mirror Replication Agent instance.

If you start a Mirror Replication Agent instance by invoking the ra utility, you can override the value of the RA_JAVA_DFLT_CHARSET system variable in your environment to specify the character set.

If you start a Mirror Replication Agent instance by invoking the instance RUN script (or batch file), you can edit the instance RUN script to specify the default value of RA_JAVA_DFLT_CHARSET and specify the character set you want to use.

### ❖ To override the system default character set for all instances

1. Enter a character set value in the *ra* script:
   - For Windows, edit the *%SYBASE%\MRO-12_6\bin\ra.bat* file.
   - For UNIX, edit the *$SYBASE/MRO-12_6/bin/ra.sh* file:

     ```
     RA_JAVA_DFLT_CHARSET=charset
     ```

   where *charset* is the Java-supported encoding.

For example: `ISO8859_1` or `Cp1252` for ISO-1 (also known as Latin-1), and `ISO8859_8` or `Cp1255` for Hebrew.

---

**Note** In UNIX, spaces are *not* allowed on either side of the equals sign.

For a list of valid Java character sets, see Supported Encodings on the Internationalization page under Documentation for the J2SE 1.4.2 JDK at http://java.sun.com/j2se/corejava/intl/index.jsp.

---

2 Uncomment the following lines of code:

For Windows:

```
set RA_JAVA_DFLT_CHARSET=charset
```

For UNIX:

```
RA_JAVA_DFLT_CHARSET=charset
export RA_JAVA_DFLT_CHARSET
```

❖ **To override the system default character set for a Mirror Replication Agent instance**

- Enter a character set value in the RUN script:

For Windows, edit the *%SYBASE%\MRO-12_6\<instance>\RUN_<instance>.bat* script:

```
set RA_JAVA_DFLT_CHARSET=charset
```

For UNIX, edit the *$SYBASE/MRO-12_6/<instance>/RUN_<instance>.sh* batch file:

```
RA_JAVA_DFLT_CHARSET=charset
export RA_JAVA_DFLT_CHARSET
```

where *charset* is the Java-supported encoding.

For example: `ISO8859_1` or `Cp1252` for ISO-1 (also known as Latin-1), and `ISO8859_8` or `Cp1255` for Hebrew.

---

**Note** In UNIX, spaces are *not* allowed on either side of the equals sign.

For a list of valid Java character sets, see Supported Encodings on the Internationalization page under Documentation for the J2SE 1.4.2 JDK at http://java.sun.com/j2se/corejava/intl/index.jsp.

---

## Starting an instance with the mro utility

When you start the Mirror Replication Agent with the mro utility, you can specify the instance start-up state. If you do *not* specify a start-up state when you invoke the mro utility, the Mirror Replication Agent instance starts in its default *Admin* state.

---

**Note** You must set the SYBASE environment before you invoke the Mirror Replication Agent mro utility. See "Setting the SYBASE environment" on page 37 for more information.

---

### ❖ To start Mirror Replication Agent with the *mro* utility

1. Open an operating system command window on the Mirror Replication Agent host machine.
2. At the operating system prompt, navigate to the Mirror Replication Agent *bin* directory.
   - On Microsoft Windows platforms, enter:

     ```
     cd %SYBASE%\MRO-12_6\bin
     ```

     where *%SYBASE%* is the path to the Mirror Replication Agent installation directory.
   - On UNIX platforms, enter:

     ```
     cd $SYBASE/MRO-12_6/bin
     ```

     where *$SYBASE* is the path to the Mirror Replication Agent installation directory.
3. In the Mirror Replication Agent *bin* directory, invoke the mro utility to start the Mirror Replication Agent instance:

   ```
   mro -i inst_name
   ```

   or

   ```
   mro -i inst_name -state
   ```

   where:

   - *inst_name* is the server name of the Mirror Replication Agent instance.
   - *state* is the optional keyword for the start-up state:

- admin – starts the Mirror Replication Agent instance in *Admin* state.
- replicate – starts the Mirror Replication Agent instance in *Replicating* state.

For example, to start the Mirror Replication Agent instance named "my_mro" in *Replicating* state, enter:

```
mro -i my_mro -replicate
```

After you start the Mirror Replication Agent instance, you must open another operating system command window to log in to its administration port.

## Starting an instance with the RUN script

The *RUN* script is named *RUN_inst_name*, where *inst_name* is the name of the Mirror Replication Agent instance. It is created automatically when the Mirror Replication Agent instance is created.

The *RUN* script invokes the mro utility with the appropriate parameter values to start the Mirror Replication Agent instance. You can edit the *RUN* script to specify the start-up state.

**Note** You do not need to set the SYBASE environment variable before you invoke the *RUN* script, because the *RUN* script sets the Sybase environment variable before it starts the Mirror Replication Agent instance.

### ❖ To start with the *RUN* script

1 Open an operating system command window on the Mirror Replication Agent host machine.

2 At the operating system prompt, navigate to the Mirror Replication Agent instance directory:

- On Microsoft Windows platforms, enter:

  ```
  cd %SYBASE%\MRO-12_6\inst_name
  ```

  where:

  - *%SYBASE%* is the path to the Mirror Replication Agent installation directory.
  - *inst_name* is the name of the Mirror Replication Agent instance.

- On UNIX platforms, enter:

  ```
  cd $SYBASE/MRO-12_6/inst_name
  ```

  where:

  - *$SYBASE* is the path to the Mirror Replication Agent installation directory.
  - *inst_name* is the name of the Mirror Replication Agent instance.

3 In the Mirror Replication Agent instance directory, invoke the *RUN* script to start the Mirror Replication Agent instance:

```
RUN_inst_name
```

where *inst_name* is the server name of the Mirror Replication Agent instance.

For example, to start the Mirror Replication Agent instance named "my_mro," enter:

```
RUN_my_mro
```

**Note** Because this *RUN* script is generated at the time that the instance is created, the UNIX version does not have the *.sh* extension.

After you start the Mirror Replication Agent instance, you must open another operating system command window to log in to its administration port.

## Using the Mirror Replication Agent administration port

When you create a Mirror Replication Agent instance, you specify a client socket port number for its administration port. Client applications use this port to connect to the Mirror Replication Agent.

The administration port allows Open Client (or Open Client-compatible) applications to log in and execute Mirror Replication Agent commands. You can use any Sybase Open Client interface utility (such as isql or SQL Advantage) to connect to the Mirror Replication Agent administration port.

**Note** Client applications are *not* provided with the Mirror Replication Agent software. The isql utility is provided with the Replication Server.

## Creating an entry in the interfaces file

In general, Open Client applications (such as isql) require an interfaces file to identify available servers, host machines, and client ports. On Microsoft Windows platforms, the interfaces file is named *sql.ini*. On UNIX platforms, the interfaces file is named *interfaces*.

If you want Open Client applications to be able to connect to the Mirror Replication Agent administration port, as they would to any other Open Server application, you must create a server entry for the Mirror Replication Agent in the interfaces file on the Open Client application host machine.

A server entry for a Mirror Replication Agent administration port in an interfaces file appears as follows:

```
[inst_name]
 query=protocol,host_name,port_num
```

where:

- *inst_name* is the name of the Mirror Replication Agent instance.
- *protocol* is the network protocol used for the connection.
- *host_name* is the name of the Mirror Replication Agent host machine.
- *port_num* is the client socket port number of the administration port.

For example, to specify an interfaces file entry for a Mirror Replication Agent instance named "my_mro," using the Microsoft Windows socket protocol, on a host named "my_host," with client socket port number 10002, you would add the following lines to the interfaces file:

```
[my_mro]
query=NLWNSCK,my_host,10002
```

Some systems require the interfaces file to be in the TLI form. If your system requires that, you must use a utility (such as sybtli or dsedit) that edits the interfaces file and saves the result in a form compatible with TLI.

After you create an entry for the Mirror Replication Agent instance in the interfaces file, you can connect to the administration port using any Open Client application that uses that interfaces file.

## Logging in to the Mirror Replication Agent

This section describes how to use the isql interactive SQL utility to log in to the Mirror Replication Agent administration port.

Before you can log in to the Mirror Replication Agent administration port with an Open Client application (such as isql), you must first create a server entry for the Mirror Replication Agent instance in the interfaces file. See "Creating an entry in the interfaces file" on page 65 for more information.

---

**Note** The first time you log in to a newly created Mirror Replication Agent instance, use the default administrator login, "sa", with no password.

---

### ❖ To log in to a Mirror Replication Agent instance

1. Open an operating system command window.
2. At the operating system prompt, enter the following command:

   ```
   isql -Uusername -Ppassword -Sinst_name
   ```

   where:

   - *username* is the Mirror Replication Agent administrator login.
   - *password* is the corresponding password.
   - *inst_name* is the name of the Mirror Replication Agent instance.

   For example, to log in to a new Mirror Replication Agent instance named "my_mro," enter:

   ```
   isql -Usa -P -Smy_mro
   ```

---

**Note** Sybase recommends that you create a new administrator login and password to replace the default "sa" login and secure access to the administration port immediately after you create a Mirror Replication Agent instance. See "Creating the Mirror Replication Agent administrator login" on page 67 for more information.

---

After you have successfully logged in to the administration port, you can use Mirror Replication Agent commands to administer the Mirror Replication Agent instance.

## Creating the Mirror Replication Agent administrator login

Each Mirror Replication Agent instance has only one administrator login. The default administrator login (`sa`, with no password) is created when the Mirror Replication Agent instance is created.

---

**Note** Sybase recommends that you create a user login name and password to replace the default “sa” login and secure access to the administration port immediately after you create a Mirror Replication Agent instance.

---

You can use ra_set_login to create (or change) the administrator login for a Mirror Replication Agent instance.

❖ **To create or change the Mirror Replication Agent administrator login**

1. Log in to the Mirror Replication Agent instance with the administrator login.

   When you log in to the Mirror Replication Agent instance for the first time, use the default administrator login.

2. After you log in, enter the following command:

   ```
   ra_set_login admin_user,admin_pw
   ```

   where:

   - *admin_user* is the new administrator login name you want to use for this Mirror Replication Agent instance.
   - *admin_pw* is the password for the new administrator login.

   ---

   **Note** Use the values from items 1e and 1f on the “Installation and Setup Worksheet” in the Mirror Activator for Oracle *Installation Guide* to specify the Mirror Replication Agent administrator login name and password.

   ---

The new login name replaces the current administrator login. The next time you log in to the Mirror Replication Agent instance, you must use the new administrator login name and password.

## Setting up Mirror Replication Agent connectivity

You must set up connectivity between the Mirror Replication Agent instance and the following Mirror Activator for Oracle system components:

- Primary data server
- Replication Server
- RSSD

Primary databases require you to perform specific setup tasks *before* you can set up connectivity between the Mirror Replication Agent and a primary database. See Appendix C, “Mirror Replication Agent and Oracle Databases,” to verify that the required setup tasks have been performed for your primary database.

**Note** The term “RSSD” in this document refers to both RSSD and ERSSD, any difference will be noted.

Setting up connectivity for the Mirror Replication Agent requires:

- Creating a user login name, with the appropriate authority in the primary data server and the primary database, for the Mirror Replication Agent
- Creating a user login name, with connect source permission in the Replication Server, for the Mirror Replication Agent
- Creating a user login name, with the appropriate authority in the RSSD data server and the RSSD, for the Mirror Replication Agent
- Setting values for the Mirror Replication Agent connection configuration parameters

You can use the “Installation and Setup Worksheet” in the Mirror Activator for Oracle *Installation Guide*, to record the values of connection configuration parameters for each Mirror Replication Agent instance.

### Creating the primary database user login name

Mirror Replication Agent requires client access to the primary database to acquire information about the database schema and database log devices.

Use the following procedure to set up a user login name in the primary data server and the primary database for the Mirror Replication Agent instance.

---

**Note** You must have a System Administrator user role in the primary data server to perform this procedure.

---

❖ **To create a primary database user login for Mirror Replication Agent**

1. Log in to the primary data server with a System Administrator user role.
2. Add the Mirror Replication Agent login name to the primary data server, and if necessary, to the primary database.
   - Refer to the Appendix C, "Mirror Replication Agent and Oracle Databases" for information about the permissions and authorities required in each type of primary data server and primary database.
   - Refer to the documentation provided by your primary data server vendor for information about the specific commands you need to execute to create the Mirror Replication Agent login name in the primary data server (and, if necessary, in the primary database).

After you set up the Mirror Replication Agent user login in the primary data server, verify that the new user login name is valid (it can log in to the primary data server and access the primary database).

## Creating the Replication Server user login name

Mirror Replication Agent requires client access to the Replication Server to send replicated transactions.

Use the following procedure to set up a Replication Server user login name for the Mirror Replication Agent instance.

---

**Note** You must have "sa" permission in the Replication Server to perform this procedure.

---

❖ **To create a Replication Server user login for Mirror Replication Agent**

1. Log in to the Replication Server with a login that has "sa" permission.
2. Create the Mirror Replication Agent user login name in the Replication Server:

```
create user mro_rs_user set password mro_rs_pwd
```

where:

- *mro_rs_user* is the Mirror Replication Agent user login name.
- *mro_rs_pwd* is the password for the user login name.

3 Grant **connect source** permission to the Mirror Replication Agent login name:

```
grant connect source to mro_rs_user
```

where *mro_rs_user* is the Mirror Replication Agent user login name.

After you set up the Mirror Replication Agent user login in the Replication Server, verify that the new user login name is valid (it can log in to the Replication Server).

## Creating the RSSD user login name

Mirror Replication Agent requires client access to the ERSSD or RSSD to obtain information about replication definitions.

The following sections describe procedures for:

- Setting up the ERSSD user login for Mirror Replication Agent
- Setting up the RSSD user login for Mirror Replication Agent

Refer to the appropriate procedure for your Replication Server configuration.

### Setting up the RSSD user login for Mirror Replication Agent

Use the following procedure to set up a user login name for the Mirror Replication Agent instance in an RSSD managed by Adaptive Server.

**Note** You can configure Replication Server to use an external Adaptive Server Enterprise database to host the RSSD information. By default, the Replication Server for Mirror Activator uses an embedded RSSD. If your environment requires that an ASE must be used to host the RSSD, these instructions will apply.

You must have a System Administrator user role in the Adaptive Server that manages the RSSD to perform this procedure.

---

**Note** See "Setting up the ERSSD user login for Mirror Replication Agent" on page 71 for information about setting up a user login name for the Mirror Replication Agent instance in an ERSSD managed by Adaptive Server® Anywhere.

---

❖ **To set up the RSSD user login for Mirror Replication Agent**

1. Log in to the Adaptive Server that manages the RSSD with a System Administrator user role.
2. Add the Mirror Replication Agent login name to the RSSD data server:

   ```
   use master
   sp_addlogin mro_rssd_user, mro_rssd_pwd, rssd_db
   ```

   where:

   - *mro_rssd_user* is the Mirror Replication Agent user login name.
   - *mro_rssd_pwd* is the password for the user login name.
   - *rssd_db* is the database name of the RSSD.
3. Add the Mirror Replication Agent user login name to the RSSD:

   ```
   use rssd_db
   sp_adduser mro_rssd_user
   ```

   where:

   - *rssd_db* is the database name of the RSSD.
   - *mro_rssd_user* is the Mirror Replication Agent user login name.

After you set up the Mirror Replication Agent user login in the RSSD, verify that the new user login name is valid (it can log in to the RSSD data server and access the RSSD).

## Setting up the ERSSD user login for Mirror Replication Agent

Use the following procedure to set up a user login name for the Mirror Replication Agent instance in an ERSSD managed by Adaptive Server Anywhere.

You must have the primary user role in the ERSSD ("sa" permission in the Replication Server) to perform this procedure.

---

**Note** See "Setting up the RSSD user login for Mirror Replication Agent" on page 70 for information about setting up a user login name for the Mirror Replication Agent instance in an RSSD managed by Adaptive Server Enterprise.

---

**❖ To set up the ERSSD user login for Mirror Replication Agent**

1. Log in to the ERSSD as the primary user.
2. Add the Mirror Replication Agent login name to the ERSSD:

   ```
   grant connect to mro_rssd_user
   identified by mro_rssd_pwd
   ```

   where:

   - *mro_rssd_user* is the Mirror Replication Agent user login name.
   - *mro_rssd_pwd* is the password for the user login name.
3. Give the Mirror Replication Agent user permission to read the Replication Server system tables:

   ```
   grant membership in group rs_systabgroup
   to mro_rssd_user
   ```

   where *mro_rssd_user* is the Mirror Replication Agent user login name.

After you set up the Mirror Replication Agent user login in the ERSSD, verify that the new user login name is valid (it can log in to the ERSSD and access the Replication Server system tables).

## Setting up the connection configuration parameters

When Mirror Replication Agent connects to another Mirror Activator for Oracle system component, it uses values stored in its configuration parameters to define the following minimal set of connection properties:

- Server host name
- Port number
- User login name
- User login password

For its connection to the Replication Server, Mirror Replication Agent relies on the values of two additional configuration parameters (rs_source_db and rs_source_ds) to identify the Replication Server primary database connection in the LTL connect source command.

The Mirror Replication Agent instance must be in *Admin* state to set up connection parameters. When the Mirror Replication Agent instance is *Admin* state, the instance has no connections established to other replication system components, but is available to execute administrative commands. For more information, see "Understanding Mirror Replication Agent states" on page 86.

---

**Note** The values of the rs_source_db and rs_source_ds parameters must *exactly match* the database and data server names specified in the create connection command for the Replication Server primary database connection. The values are case sensitive.

---

For more information about the rs_source_db and rs_source_ds parameters, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

You can use the "Installation and Setup Worksheet," in the Mirror Activator for Oracle *Installation Guide*, to record the values of connection configuration parameters for each Mirror Replication Agent instance.

Refer to the "Installation and Setup Worksheet" for the connection configuration parameter values you need to set in the following procedures.

---

**Note** The Mirror Replication Agent instance must be running before you can set its connection configuration parameter values. See "Starting the Mirror Replication Agent" on page 58 for more information.

---

### ❖ To set up connection parameters for the primary database

When the Mirror Replication Agent instance is *Admin* state, the instance has no connections established to other replication system components, but is available to execute administrative commands. The Mirror Replication Agent instance must be in *Admin* state to setup connection parameters.

1 Log in to the Mirror Replication Agent administration port, and verify that the Mirror Replication Agent instance is in *Admin* state.

   a Use the following command to verify that the Mirror Replication Agent instance is in *Admin* state:

```
ra_status
```

   b. If the instance is not in *Admin* state, use the following command to put it in *Admin* state:

      ```
      suspend
      ```

2. Specify the primary data server host name:

   ```
   ra_config pds_hostname, pds_host
   ```

   where *pds_host* is the network name of the primary data server host machine.

3. Specify the primary data server port number:

   ```
   ra_config pds_port_number, NNN
   ```

   where *NNN* is the number of the network port where the primary data server listens for connections.

4. Specify the primary database name:

   ```
   ra_config pds_database_name, pdb
   ```

   where *pdb* is the Oracle SID.

5. Specify the primary data server user login name for the Mirror Replication Agent instance:

   ```
   ra_config pds_username, mro_pds_user
   ```

   where *mro_pds_user* is the user login name that the Mirror Replication Agent uses to log in to the primary data server.

6. Specify the password for the Mirror Replication Agent user login:

   ```
   ra_config pds_password, mro_pds_pwd
   ```

   where *mro_pds_pwd* is the password for the user login name that the Mirror Replication Agent uses to log in to the primary data server.

After you set up connection configuration parameters for the primary database, you can use the Mirror Replication Agent test_connection PDS command to test connectivity between the Mirror Replication Agent and the primary database. See "Testing network connectivity" on page 77 for more information.

❖ **To set up connection parameters for the Replication Server**

1. Log in to the Mirror Replication Agent administration port, and verify that the Mirror Replication Agent instance is in *Admin* state:

   a. Use the following command to verify that the Mirror Replication Agent instance is in *Admin* state:

```
ra_status
```

b If the instance is not in *Admin* state, use the following command to put it in *Admin* state:

```
suspend
```

2 Specify the Replication Server host name:

```
ra_config rs_hostname, rs_host
```

where *rs_host* is the network name of the Replication Server host machine.

3 Specify the Replication Server port number:

```
ra_config rs_port_number, NNN
```

where *NNN* is the number of the network port where Replication Server listens for connections.

4 Specify the Replication Server character set:

```
rs_charset
```

where rs_charset matches the character set used by Replication Server. If rs_charset is not set, Mirror Replication Agent refuses to replicate anything.

5 Specify the Replication Server user login name for the Mirror Replication Agent instance:

```
ra_config rs_username, mro_rs_user
```

where *mro_rs_user* is the user login name that the Mirror Replication Agent uses to log in to the Replication Server.

6 Specify the user login password for the Mirror Replication Agent instance:

```
ra_config rs_password, mro_rs_pwd
```

where *mro_rs_pwd* is the password for the user login name that the Mirror Replication Agent uses to log in to the Replication Server.

7 Specify the primary data server name for the Replication Server primary database connection:

```
ra_config rs_source_ds, pds
```

where *pds* is the primary data server name that the Mirror Replication Agent uses in the LTL connect source command.

8 Specify the primary database name for the Replication Server primary database connection:

```
ra_config rs_source_db, pdb
```

where *pdb* is the primary database name that the Mirror Replication Agent uses in the LTL connect source command.

### ❖ To set up connection parameters for the ERSSD (or RSSD)

1 Log in to the Mirror Replication Agent administration port, and verify that the Mirror Replication Agent instance is in *Admin* state:

   a Use the following command to verify that the Mirror Replication Agent instance is in *Admin* state:

   ```
   ra_status
   ```

   b If the instance is not in *Admin* state, use the following command to put it in *Admin* state:

   ```
   suspend
   ```

2 Specify the ERSSD host name:

   ```
   ra_config rssd_hostname, rssd_host
   ```

   where e*rssd_host* is the network name of the ERSSD host machine.

3 Specify the ERSSD port number:

   ```
   ra_config rssd_port_number, NNN
   ```

   where *NNN* is the number of the network port where the ERSSD server listens for connections.

4 Specify the ERSSD database name:

   ```
   ra_config rssd_database_name, rssd_db
   ```

   where *rssd_db* is the database name of the ERSSD.

5 Specify the ERSSD user login name for the Mirror Replication Agent instance:

   ```
   ra_config rssd_username, mro_rssd_user
   ```

   where *mro_erssd_user* is the user login name that the Mirror Replication Agent uses to log in to the ERSSD.

6 Specify the user login password for the Mirror Replication Agent instance:

   ```
   ra_config rssd_password, mro_rssd_pwd
   ```

   where *mro_rssd_pwd* is the password for the user login name that the Mirror Replication Agent uses to log in to the RSSD.

After you set up connection configuration parameters for the Replication Server and RSSD, you can use the Mirror Replication Agent test_connection RS command to test connectivity between the Mirror Replication Agent and the Replication Server and RSSD. For more information, see "Testing network connectivity" on page 77.

## Testing network connectivity

Mirror Replication Agent provides a simple means of testing network connections. The test_connection command sends a connection request and confirms the network connection to the following servers:

- Primary data server
- Replication Server
- RSSD server (if so configured)

**Note** If the value of the use_rssd configuration parameter is true, the test_connection command tests Mirror Replication Agent connectivity to the RSSD when it tests connectivity to the Replication Server. If the value of the use_rssd configuration parameter is false, the test_connection command does *not* test Mirror Replication Agent connectivity to the RSSD.

The test_connection command returns a failure message if:

- The connection specifications (server name, port number, user login, and so forth) recorded in the Mirror Replication Agent configuration parameters are not correct.
- The Mirror Replication Agent cannot establish a connection to a server because of a network failure.
- The Mirror Replication Agent cannot establish a connection to a server because the server is down.

The test_connection command does *not* validate Mirror Replication Agent user login permissions in the primary database. It verifies only that the user login and password specified in the pds_username and pds_password parameters can log in to the primary data server.

The test_connection command does verify that the Mirror Replication Agent user login (specified in the rs_username and rs_password parameters) has connect source permission in the Replication Server.

See “Setting up Mirror Replication Agent connectivity” on page 68 for information about setting up Mirror Replication Agent connection configuration parameters.

---

**Note** You must be in *Admin* state to test network connectivity.

---

❖ **To verify Mirror Replication Agent connections**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to test Mirror Replication Agent network connections:

   ```
   test_connection
   ```

   This command tests all of the connections from the Mirror Replication Agent instance you logged in to.

   ---

   **Note** You can test a specific connection (either the primary data server or the Replication Server) by specifying the connection you want to test.

   ---

If the test_connection command returns a failure, check the Mirror Replication Agent system log to determine the cause of the failure. You may also need to check the system log of the server associated with the connection to determine the cause of the failure.

See the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual* for more information about the test_connection command.

## Marking objects in the primary database

Tables, stored procedures, and sequences must be marked for replication and have replication enabled for the object (table, stored procedure, or sequence). LOB columns must have replication enabled, and the table that contains the LOB column must be marked for replication and have replication enabled for that table.

There are five types of objects that can be marked for replication in a primary database:

- Tables
- Stored procedures
- Large-object (LOB) columns
- Database Definition Language (DDL)
- Sequences

## Marking tables in the primary database

For transactions against a table to be replicated, the primary table in the primary database must be marked for replication and replication must be enabled for that table. The setting of pdb_convert_datetime affects the generated results. For a detailed description of the pdb_convert_datetime configuration parameter, refer to Chapter 2, Configuration Parameters of the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual.*

❖ **To mark a table in the primary database**

1. Log in to the Mirror Replication Agent administration port.
2. Use the pdb_setreptable command to determine if the table you want to mark is already marked in the primary database:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the table in the primary database that you want to mark for replication.

   - If the pdb_setreptable command returns information that the specified table is marked and replication is enabled, you need not continue this procedure.
   - If the pdb_setreptable command returns information that the specified table is marked but replication is disabled, skip step 3 and continue this procedure from step 4 to enable replication for the table.
   - If the pdb_setreptable command returns information that the specified table is not marked, continue this procedure to mark the table for replication.
3. Use the pdb_setreptable command to mark the table for replication and specify the name to use for replication:

- Use the following command to mark the table for replication using a replication definition with the same table name:

  ```
  pdb_setreptable pdb_table, mark
  ```

  where *pdb_table* is the name of the table in the primary database that you want to mark for replication.

- Use the following command to mark the table for replication using a replication definition with a different table name:

  ```
  pdb_setreptable pdb_table, rep_table, mark
  ```

  where:

  - *pdb_table* is the name of the table in the primary database that you want to mark for replication.
  - *rep_table* is the name of the table in the with all tables named rep_table clause in the replication definition for this table.

- When marking a table for replication, you can specify that the table owner's name is sent along with the table name in the LTL. To do this, use the owner keyword after the mark keyword, as shown in the following example:

  ```
  pdb_setreptable pdb_table, mark, owner
  ```

  where *pdb_table* is the name of the table in the primary database that you want to mark for replication.

If the pdb_dflt_object_repl parameter is set to true (the default), the table marked for replication with the pdb_setreptable command is ready for replication after you invoke the pdb_setreptable command successfully, and you can skip step 4 in this procedure.

If the pdb_dflt_object_repl parameter is set to false, you must enable replication for the table before replication can take place.

4 Use the pdb_setreptable command to enable replication for the marked table:

```
pdb_setreptable pdb_table, enable
```

where *pdb_table* is the name of the marked table in the primary database for which you want to enable replication.

5 Use the pdb_setreptable command with the 'all' keyword to mark or enable all user tables at once:

```
pdb_setreptable all, {mark|enable}
```

where mark or enable are the keywords identifying the action that should be taken against all user tables in the database.

After the table is marked and replication is enabled for the table, you can begin replicating transactions that affect data in that table.

## Marking stored procedures in the primary database

To replicate invocations of a stored procedure in the primary database, the stored procedure must be marked for replication, and replication must be enabled for that stored procedure.

---

**Note** DDL replication must be disabled before marking (or unmarking) a stored procedure.

---

❖ **To mark a stored procedure in the primary database**

1 Log in to the Mirror Replication Agent administration port.

2 Use the pdb_setrepproc command to determine if the stored procedure you want to mark is already marked in the primary database:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.

- If the pdb_setrepproc command returns information that the specified stored procedure is marked and replication is enabled, you need not continue this procedure.
- If the pdb_setrepproc command returns information that the specified stored procedure is marked but replication is disabled, skip step 3 and continue this procedure from step 4 to enable replication for the stored procedure.
- If the pdb_setrepproc command returns information that the specified stored procedure is not marked, continue this procedure to mark the stored procedure for replication.

3 Use the pdb_setrepproc command to mark the stored procedure for replication and specify the name to use for replication:

- Use the following command to mark the stored procedure for replication using a function replication definition with the same procedure name:

  ```
  pdb_setrepproc pdb_proc, mark
  ```

  where *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.

- Use the following command to mark the stored procedure for replication using a function replication definition with a different procedure name:

  ```
  pdb_setrepproc pdb_proc, rep_proc, mark
  ```

  where:

  - *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.
  - *rep_proc* is the name of the stored procedure in the with all procedures named rep_proc clause in the function replication definition for this stored procedure.

If the pdb_dflt_object_repl parameter is set to true (the default), the stored procedure marked for replication with the pdb_setrepproc command is ready for replication after you invoke the pdb_setrepproc command successfully, and you can skip step 4 in this procedure.

If the pdb_dflt_object_repl parameter is set to false, you must enable replication for the stored procedure so replication can take place.

4 Use the pdb_setrepproc command to enable replication for the marked stored procedure:

```
pdb_setrepproc pdb_proc, enable
```

where *pdb_proc* is the name of the marked stored procedure in the primary database for which you want to enable replication.

After the stored procedure is marked and replication is enabled for the stored procedure, you can begin replicating invocations of that stored procedure.

## Enabling replication for LOB columns

For transactions that affect a LOB column to be replicated, the table that contains the LOB column must be marked for replication and have replication enabled.

If the value of the pdb_dflt_column_repl parameter is set to true, all LOB columns in a table have replication enabled automatically when you mark the table (by invoking the pdb_setreptable command). If the value of the pdb_dflt_column_repl parameter is set to false, you must enable replication separately for each LOB column before replication can take place.

**Note** The default value of the pdb_dflt_column_repl parameter is false.

❖ **To enable replication for a LOB column in the primary database**

1. Log in to the Mirror Replication Agent administration port.
2. Use the pdb_setrepcol command to determine if replication is already enabled for the LOB column you want to enable replication for in the primary database:

   ```
   pdb_setrepcol tablename, pdb_col
   ```

   where:

   - *tablename* is the name of the table that contains the LOB column.
   - *pdb_col* is the name of the LOB column in the primary database.

   If the pdb_setrepcol command returns information that replication is enabled for the specified column, you need not continue this procedure.

   If the pdb_setrepcol command returns information that replication is not enabled for the specified column, continue this procedure to enable replication for the column.
3. Use the pdb_setrepcol command to enable replication for the LOB column:

   ```
   pdb_setrepcol tablename, pdb_col, enable
   ```

   where:

   - *tablename* is the name of the table that contains the LOB column.
   - *pdb_col* is the name of the LOB column in the primary database for which you want to enable replication.

After replication is enabled for the LOB column, you can begin replicating transactions that affect data in that column.

## Enabling replication for DDL

For DDL to be replicated, the pdb_setrepddl command must be set to enable. If pdb_setrepddl is set to enable, all DDL in your primary database is replicated. Also, you must set the ddl_username and the ddl_password.

---

**Note** To replicate DDL:

- Mirror Replication Agent requires a unique user name to be supplied that will have authority to execute all DDL commands at the standby database.
- Replication Server must have a database-level replication definition with replicate DDL set in the definition.

For details about configuration property ddl_username and for database-level replication definition, refer to the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

---

❖ **To enable replication for DDL in the primary database**

1. Log in to the Mirror Replication Agent administration port.
2. Use the pdb_setrepddl command without an argument to determine if replication is already enabled for DDL in the primary database:

   ```
   pdb_setrepddl
   ```

   If the pdb_setrepddl command returns information that replication is enabled, you do not need to continue this procedure.

   If the pdb_setrepddl command returns information that replication is not enabled for DDL, continue this procedure to enable replication for DDL.
3. Use the pdb_setrepddl command to enable replication for DDL:

   ```
   pdb_setrepddl enable
   ```

   After replication is enabled for the DDL, you can begin replicating your primary database.

   For details on enabling DDL replication, see Appendix C, "Mirror Replication Agent and Oracle Databases."

CHAPTER 3

# Administering Mirror Replication Agent

This chapter describes administrative tasks and procedures for the Mirror Replication Agent.

For information about installing the Mirror Replication Agent software, see the Mirror Activator for Oracle *Installation Guide*.

For information about setting up the Mirror Activator for Oracle system, including Mirror Replication Agent, see Chapter 2, "Setup and Configuration."

**Note** Although example procedures in this chapter show isql as the Open Client application used to log in to the Mirror Replication Agent administration port you can use any Open Client (or Open Client-compatible) application to do so.

## Determining current Mirror Replication Agent status

The Mirror Replication Agent status consists of the current state and activity of the Mirror Replication Agent instance.

❖ **To determine the status of a Mirror Replication Agent instance**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the following command to get current status for the Mirror Replication Agent instance:

```
ra_status
```

This command returns the current state of the Mirror Replication Agent instance and any current activity, as shown in the following example:

```
State  Action
------ ----------------------------
ADMIN  Waiting for operator command
(1 row affected)
```

## Understanding Mirror Replication Agent states

When a Mirror Replication Agent instance is running, it can be in one of two discrete states:

- *Admin* – the instance has no connections established to other Mirror Activator for Oracle system components, but it is available to execute administrative commands, such as initializing the instance, changing configuration parameters, and maintaining the RASD. No replication processing occurs when the Mirror Replication Agent instance is in *Admin* state.
- *Replicating* – the instance is performing its normal replication processing—scanning the transaction log on the mirror log devices, processing log records and change-set data, and sending LTL commands to the Replication Server. In *Replicating* state, some administrative commands are not allowed.

**Note** When the state appears as *Replicating (Waiting at end of log)*, the instance is ready to perform (or continue) its normal replication processing, but the Log Reader component is idle and waiting for additional transactions to be logged.

The default start-up state is *Admin*. The Mirror Replication Agent instance goes to *Admin* state automatically when no start-up state is specified.

The state of a Mirror Replication Agent instance can be changed by either:

- An external event that occurs while the Mirror Replication Agent is processing replicated transactions (for example, a network error on the Replication Server connection), or
- Operator intervention (for example, invoking a command that changes the Mirror Replication Agent state).

From the moment a state-changing event occurs until the Mirror Replication Agent instance is actually in the new state, the instance is said to be "in transition." During state transition, some administrative commands are ignored.

#### Admin state

A Mirror Replication Agent instance goes to *Admin* state when:

- The instance is started in its default state.
- The instance is started with the mro utility -admin option.
- The Mirror Replication Agent quiesce or suspend command is invoked.
- An unrecoverable error occurs when the instance is in *Replicating* state.

In *Admin* state, the Mirror Replication Agent instance is running, but it has no connection established to the Replication Server (or RSSD, if so configured) or the primary database.

You can perform most administrative tasks while the Mirror Replication Agent instance is in *Admin* state, including changing the value of any Mirror Replication Agent configuration parameter.

**Note** In *Admin* state, the instance can open a connection to the primary database, if necessary, to process a command that requests results from the primary database.

A Mirror Replication Agent instance may go to *Admin* state from *Replicating* state when a network failure or communication error causes its connection to the Replication Server to be dropped.

When Mirror Replication Agent drops a connection, before it goes to *Admin* state, it attempts to re-establish the connection using the values recorded in its configuration parameters for that connection. If it cannot reconnect to the Replication Server, the Mirror Replication Agent instance goes to *Admin* state.

#### Replicating state

A Mirror Replication Agent instance goes to *Replicating* state when:

- The instance is started with the mro utility -replicate option.
- The Mirror Replication Agent resume command is invoked.

In *Replicating* state, the Mirror Replication Agent instance maintains a connection to the Replication Server, and its Log Reader component scans the mirror log devices for transactions to replicate.

Normally, when the Mirror Replication Agent instance is in *Replicating* state, it maintains a connection to the primary database to perform log truncation and archiving functions. If it drops the primary database connection (for example, because the primary database goes offline), Mirror Replication Agent logs the event and continues its normal replication processing, scanning the mirror log devices for transactions to replicate.

If the Mirror Replication Agent instance has finished processing all of the records in the transaction log, its state may appear as *Replicating (Waiting at end of log)*. In that case:

- The Log Reader component's log-scanning process "sleeps" according to the values of the scan_sleep_increment and scan_sleep_max configuration parameters.
- After the Log Transfer Interface (LTI) component finishes processing all of the change sets it received from the Log Reader and sending all of the LTL to the Replication Server, no replication throughput occurs until new replicated transactions appear in the log and the Log Reader scans them.
- The Mirror Replication Agent instance remains in *Replicating* state, unless some other event causes it to go to *Admin* state.

## Changing the Mirror Replication Agent state

The state of a Mirror Replication Agent instance indicates its current operational condition, and determines which administrative tasks you can perform.

Generally, there are only two reasons to change the state of a Mirror Replication Agent instance:

- To perform certain administrative or maintenance procedures (change the state from *Replicating* to *Admin*)
- To restore normal replication processing (change the state from *Admin* to *Replicating*), either after an administrative or maintenance procedure, or after recovery from an error

**Changing from *Replicating* state to *Admin* state**

To change the state of the Mirror Replication Agent instance from *Replicating* to *Admin*, you can use either the quiesce or suspend command. For more information, see "Stopping replication in the Mirror Replication Agent" on page 92.

For more detailed information about the quiesce and suspend commands, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

**Changing from *Admin* state to *Replicating* state**

To change the state of the Mirror Replication Agent instance from *Admin* to *Replicating*, you can use the resume command. For more information, see "Starting replication in the Mirror Replication Agent" on page 91.

For more detailed information about the resume command, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

### Getting Mirror Replication Agent statistics

The Mirror Replication Agent records information about the performance of its internal components whenever it is in *Replicating* state. You can use this information to tune Mirror Replication Agent performance or troubleshoot problems.

To get information about Mirror Replication Agent performance, use the ra_statistics command. You can also use ra_statistics to reset the statistics counters.

---

**Note** Each time the Mirror Replication Agent instance goes to *Replicating* state, statistics counters are reset automatically.

---

For more information about the ra_statistics command and Mirror Replication Agent performance statistics, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

## Shutting down the Mirror Replication Agent instance

Each Mirror Replication Agent instance can be started and shut down independently of all other components in a Mirror Activator for Oracle system, and independently of other Mirror Replication Agent instances.

For information about how to start a instance, see "Starting the Mirror Replication Agent" on page 58.

Shutting down the Mirror Replication Agent instance terminates its process on the host machine.

---

**Note** You can stop all replication processing in the Mirror Replication Agent without shutting down the instance. For more information, see "Stopping replication in the Mirror Replication Agent" on page 92.

---

To shut down a Mirror Replication Agent instance, you must log in to the administration port and invoke the shutdown command. The shutdown command gives you two options:

- Normal shutdown – first quiesces the Mirror Replication Agent instance, and then shuts down the instance, terminating its process.
- Immediate shutdown – shuts down the Mirror Replication Agent instance and terminates its process immediately, without first quiescing. To use this method, use the immediate keyword when you invoke the shutdown command.

---

**Note** If the Mirror Replication Agent instance is in state transition, it ignores the shutdown command with no option (normal shutdown). It does *not* ignore shutdown immediate when it is in any state, including transition from one state to another.

---

When a Mirror Replication Agent instance is shut down normally, it does the following:

- Stops reading the mirror log devices
- Drops its connection to the primary database (if it is connected)
- Finishes processing any transactions it already has in its internal queues
- Drops its connection to the Replication Server after successfully sending LTL for any transactions in its internal queues
- Terminates its process

### ❖ To shut down a Mirror Replication Agent instance

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Invoke the shutdown command as follows:

- Use the following command to shut down the Mirror Replication Agent instance normally:

  ```
  shutdown
  ```

  This command first quiesces the Mirror Replication Agent instance before shutting it down. If the Mirror Replication Agent instance is in *Replicating* state, and the internal queues are full when you invoke shutdown, the processing may take a while to complete, and there may be a delay before the process terminates.

  For more information about quiesce processing, see "Quiescing the Mirror Replication Agent" on page 93.

- Use the following command to force an immediate shutdown, regardless of the state of the Mirror Replication Agent instance:

  ```
  shutdown immediate
  ```

  This command shuts down and terminates the Mirror Replication Agent instance immediately, without first quiescing.

For more detailed information about the shutdown command, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

## Starting replication in the Mirror Replication Agent

When you start replication in the Mirror Replication Agent:

- The Mirror Replication Agent instance opens network connections to the primary database and Replication Server (and the RSSD, if so configured).
- The internal Log Reader and Log Transfer Interface components begin normal replication processing—scanning the transaction log for operations to replicate, and sending replicated transactions to the Replication Server.
- The Mirror Replication Agent instance goes from *Admin* state to *Replicating* state.

When the Mirror Replication Agent instance is in *Replicating* state, it maintains a connection to the Replication Server, and its Log Reader component scans the mirror log devices for transactions to replicate.

The Mirror Replication Agent instance must be running before you can start replication. For more information, see “Starting the Mirror Replication Agent” on page 58.

---

**Note** Before you attempt to replicate transactions from a mirror log device, use the checklist in “Setting up the Mirror Activator for Oracle system” on page 15 to verify that your Mirror Activator for Oracle system is set up properly.

---

❖ **To start replication in the Mirror Replication Agent**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the following command to start replication:

```
resume
```

After you invoke the resume command, the Mirror Replication Agent instance should go from *Admin* state to *Replicating* state.

3 Use the following command to verify that the Mirror Replication Agent instance is in *Replicating* state:

```
ra_status
```

If the Mirror Replication Agent instance does not go to *Replicating* state after you invoke the resume command, see Chapter 4, “Troubleshooting Mirror Replication Agent,” for more information.

For more detailed information about the resume command and how Mirror Replication Agent starts replication processing, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

## Stopping replication in the Mirror Replication Agent

When you stop replication in the Mirror Replication Agent:

- The internal Log Reader and Log Transfer Interface components stop their normal replication processing.
- Any open connections to the primary database are released, and the connection to the Replication Server is dropped.

- The Mirror Replication Agent instance goes from *Replicating* state to *Admin* state.

When the Mirror Replication Agent instance is in *Admin* state, it is running and available to execute administrative commands, but it does not maintain connections to the primary database and Replication Server (and RSSD, if so configured), and it does not process replicated transactions.

Some administrative tasks require the Mirror Replication Agent instance to be in *Admin* state. In a normal operating Mirror Activator for Oracle system, you must stop replication in the Mirror Replication Agent to perform those tasks.

There are two ways to stop replication in the Mirror Replication Agent:

- Quiesce the Mirror Replication Agent instance to stop replication gracefully. For more information, see “Quiescing the Mirror Replication Agent” on page 93.
- Suspend the Mirror Replication Agent instance to stop replication immediately. For more information, see “Suspending the Mirror Replication Agent instance” on page 94.

## Quiescing the Mirror Replication Agent

Quiescing the Mirror Replication Agent instance stops its replication processing gracefully, ensuring all transactions from the mirrored log have been read and sent to the Replication Server:

- The Log Reader component stops reading operations from the transaction log as soon as the end of the last log file has been reached. It continues to send change-set data to the Log Transfer Interface component until it finishes processing the last operation it scanned from the log. If activity from the primary database is still occurring, the “end of the log” may not be reached immediately, extending the amount of time the Mirror Replication Agent quiesce command executes.
- The Log Transfer Interface component stops sending LTL commands to the Replication Server as soon as it finishes processing the last change set it received from the Log Reader.
- When the Log Transfer Interface component is finished processing its input queue and sending the resulting LTL, the Mirror Replication Agent instance releases all of its connections to the primary database (if any are open), and drops its connection to the Replication Server (and RSSD, if connected).

- The Mirror Replication Agent instance goes from *Replicating* state to *Admin* state.

❖ **To quiesce a Mirror Replication Agent instance**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to quiesce the Mirror Replication Agent:

   ```
   quiesce
   ```

   After you invoke the quiesce command, the Mirror Replication Agent instance should go from *Replicating* state to *Admin* state.
3. Use the following command to verify that the Mirror Replication Agent instance is in *Admin* state:

   ```
   ra_status
   ```

**Note** If the internal queues are full and the primary database is still recording new activity to the log files when you invoke the quiesce command, the quiesce processing may take a while to complete, and there may be a delay before the Mirror Replication Agent instance completes the transition to *admin* state.

For more detailed information about the quiesce command and Mirror Replication Agent quiesce processing, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

## Suspending the Mirror Replication Agent instance

Suspending the Mirror Replication Agent instance stops its replication processing immediately:

- The Log Reader component stops scanning the transaction log immediately, and the Log Transfer Interface component stops sending LTL commands to the Replication Server immediately.
- All data in the Mirror Replication Agent internal queues (input and output queues of the Log Reader and Log Transfer Interface components) is flushed without further processing.
- The Mirror Replication Agent instance releases all of its connections to the primary database (if any are open), and drops its connection to the Replication Server (and RSSD, if connected).

- The Mirror Replication Agent instance goes from *Replicating* state to *Admin* state.

❖ **To suspend a Mirror Replication Agent instance**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to suspend the Mirror Replication Agent:

   ```
   suspend
   ```

   After you invoke suspend, the Mirror Replication Agent instance should go from *Replicating* state to *Admin* state.
3. Use the following command to verify that the Mirror Replication Agent instance is in *Admin* state:

   ```
   ra_status
   ```

For more detailed information about the suspend command, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

## Managing the Replication Agent System Database

Mirror Replication Agent uses an embedded database, for its Replication Agent System Database (RASD).

There are five tasks you can perform to maintain the RASD:

- "Updating the RASD" on page 97
- "Updating the log device repository" on page 99
- "Backing up the RASD" on page 102
- "Restoring the RASD" on page 102
- "Truncating the RASD" on page 104

### RASD overview

Each Mirror Replication Agent instance depends on the information in its RASD to recognize database structure or schema objects in the transaction log, and to keep track of mirror log devices.

The Mirror Replication Agent RASD is an important fault-tolerance feature of the Mirror Activator for Oracle system. It allows the Mirror Replication Agent to recover from failures in its own environment, so that all logged transactions on the mirror log devices can be replicated when the Mirror Replication Agent starts up, even when the primary database is offline.

When you create a Mirror Replication Agent instance, the RASD is created automatically, but it contains no information until you *initialize* the Mirror Replication Agent instance.

**Note** Before you can start replication with the Mirror Activator for Oracle system, you must initialize the Mirror Replication Agent instance after the instance is created and its connection configuration parameters are set. You must execute the ra_init command to load the RASD. For more information, see Chapter 2, "Setup and Configuration."

When you initialize a Mirror Replication Agent instance, it does the following:

- Queries the primary database to get information about the database structure or schema
- Queries the primary database to get information about the transaction log devices
- Stores information about the database schema and transaction log devices in its RASD

Initializing the Mirror Replication Agent is one of the tasks required to set up the Mirror Activator for Oracle system, and it has several prerequisites. For more information about these tasks and how to initialize the Mirror Replication Agent, see "Setting up the Mirror Activator for Oracle system" on page 15.

After the RASD is initially populated, its contents will be synchronized with the primary database automatically during normal replication (without intervention).

If replication does not occur, the contents of the RASD will become stale (out of sync with the primary database), and you should rebuild them before use.

DDL commands

Most of the common data definition language (DDL) commands and system procedures executed in the primary database are recorded in the transaction log, and they are replicated to the standby database. When it processes those DDL transactions for replication, the Mirror Replication Agent updates the RASD automatically.

If a DDL command or system procedure produces a change in the primary database schema and the Mirror Replication Agent cannot recognize that command or procedure and update its RASD automatically, a replication failure will occur if a subsequent transaction changes data in an object that is not recorded in the RASD. In that event, you must re-initialize and quiesce the primary database, and force the Mirror Replication Agent to update its RASD. For more information, see “Updating the RASD” on page 97.

Each time it processes a DDL transaction that affects an existing database object, the Mirror Replication Agent creates a new *version* of the object metadata in its RASD. The version of each object is identified by the LTM Locator value of the DDL transaction that changed it.

Previous versions of objects must be kept in the RASD long enough to allow system recovery. For example, replaying a transaction that involved an object before it was changed by DDL could produce an error (or data inconsistency) with the current version of the object.

**Note** The Mirror Replication Agent does not support replaying transactions from a restored transaction log.

Object versions and LTM Locator values

The Mirror Replication Agent determines which version of each object to use by comparing the object’s current version string with the current LTM Locator value. If the current LTM Locator value is greater than or equal to the value of the object version, the current object metadata is used. If the current LTM Locator value is less than the value of the object version, a previous version of the object metadata must be used.

Without periodic truncation, the size of the RASD can grow indefinitely, as more and more versions of object metadata are added. For more information, see “Truncating the RASD” on page 104.

## Updating the RASD

**Note** The RASD is usually updated automatically during normal replication activity. The following procedure to force an update of the RASD should only be used with the recommendation of Sybase Technical Support when the RASD is suspected of being corrupt.

After the data in the RASD is created by initializing the Mirror Replication Agent instance, the only way to update the RASD is to *re-initialize* the instance using the ra_init command with the force keyword.

---

**Note** Before you re-initialize the Mirror Replication Agent, the primary database must be quiesced.

---

### ❖ To update the RASD

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to determine the state of the Mirror Replication Agent instance:

   ```
   ra_status
   ```

3. If the Mirror Replication Agent is in *Admin* state, skip this step and continue with step 4.

   If the Mirror Replication Agent is in *Replicating* state:

   a. Use the following command to suspend replication by the Mirror Replication Agent instance:

      ```
      suspend
      ```

   b. Use the following command to verify that the Mirror Replication Agent is in *Admin* state:

      ```
      ra_status
      ```

4. Before you re-initialize the RASD, you must quiesce the primary database, or otherwise prevent any DDL operations that can change the database objects or schema.

   Log in to the primary data server, using a user login that has appropriate permissions or authority, and then quiesce the primary database (or execute the commands that will prevent any DDL operations that change the database objects or schema).

5. Use the following command to re-initialize the Mirror Replication Agent and force it to update its RASD:

```
ra_init force
```

**Note** The ra_init force command does *not* overwrite any existing path information to the redo log in the RASD, if all of the following log information in the RASD matches that returned by the primary data server:

- redo group number
- redo name
- redo path

For each redo log identified in the RASD, if any item in the preceding list does *not* match the information returned by the primary data server, ra_init force overwrites the RASD record for that redo log with the information returned by the primary data server.

6. After the Mirror Replication Agent is initialized, you must release the quiesce on the primary database to restore normal client access to the database.

   Log in to the primary data server, using a user login that has appropriate permissions or authority, and then release the quiesce on the primary database (or execute the commands necessary to restore normal client access to the database).

7. Use the following command to resume replication in the Mirror Replication Agent:

   ```
   resume
   ```

8. Use the following command to verify that the Mirror Replication Agent is in *Replicating* state:

   ```
   ra_status
   ```

If the Mirror Replication Agent does not return to *Replicating* state, see Chapter 4, “Troubleshooting Mirror Replication Agent,” for more information.

## Updating the log device repository

Mirror Replication Agent stores information about primary log devices in its RASD when you initialize the Mirror Replication Agent instance. Log device information in the RASD is referred to as the *log device repository*.

Unlike other information in the RASD, you can update the log device repository at any time using the ra_updatedevices command.

---

**Note** If any log device is added, dropped, extended, or moved at the primary database, the Mirror Replication Agent log device repository must be updated. Sybase recommends that you coordinate all log device changes at the primary database with updating the Mirror Replication Agent log device repository.

---

When you update the log device repository, Mirror Replication Agent does the following:

- Queries the primary database for information about all of its log devices.
- Compares the information returned by the primary database with the information recorded in the log device repository.
- Updates the log device repository with the new information returned by the primary database, if:
    - It finds information for existing log devices in the log device repository that does not match the information returned by the primary database, or
    - It finds information about new log devices in the information returned by the primary database.

If the path for a log device at the primary site is different from the path for the corresponding mirror log device at the standby site, you must use ra_devicepath to specify the path to the mirror log device recorded in the RASD.

---

**Note** The primary database need not be quiesced when you update the Mirror Replication Agent log device repository.

---

❖ **To update the log device repository**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to determine the state of the Mirror Replication Agent instance:

   ```
   ra_status
   ```

3. If the Mirror Replication Agent is in *Admin* state, skip this step and continue with step 4.

   If the Mirror Replication Agent is in *Replicating* state:

a. Use the following command to suspend replication by the Mirror Replication Agent instance:

   ```
   suspend
   ```

b. Use the following command to verify that the Mirror Replication Agent is in *Admin* state:

   ```
   ra_status
   ```

4. If you coordinate log device changes at the primary database with updating the Mirror Replication Agent log device repository, make the log device changes at the primary database after the Mirror Replication Agent is in *Admin* state.

5. After you verify that the Mirror Replication Agent is in *Admin* state, use the following command to update the log device repository in the RASD:

   ```
   ra_updatedevices
   ```

6. If you need to specify the path for a mirror log device, use ra_devicepath.

   Use the following command to specify a mirror log device path:

   ```
   ra_devicepath device, dev_path
   ```

   where:

   - *device* is the device ID (Oracle Group ID).
   - *dev_path* is the path that the Mirror Replication Agent must use to connect to the mirror primary database log device at the standby site.

   **Note** You must invoke ra_devicepath once for each mirror log device whose path you need to specify.

7. Use the following command to start replication in the Mirror Replication Agent instance:

   ```
   resume
   ```

You can update the log device repository as often as necessary to accommodate log device changes at the primary database.

## Backing up the RASD

Like any database, you should periodically back up the RASD to prevent data loss in the event of a device failure.

**Note** Sybase recommends that you always back up the RASD before you truncate the RASD.

The Mirror Replication Agent places RASD backup files in the directory identified by the rasd_backup_dir configuration parameter. For more information about the rasd_backup_dir parameter, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

You do not need to interrupt replication to back up the RASD. You can back up the RASD at any time, when the Mirror Replication Agent instance is in any state.

❖ **To back up the RASD**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to back up the RASD:

```
rasd_backup
```

After the backup completes successfully, the Mirror Replication Agent returns a message to confirm that the RASD backup was successful.

If the Mirror Replication Agent could not find the directory identified in the rasd_backup_dir parameter, or if it could not write the RASD backup files in that directory (for example, because of a permission problem), it returns an error. You must correct the cause of the error before you can successfully back up the RASD.

## Restoring the RASD

If the RASD becomes corrupt (for example, because of a device failure), you can restore the database from the most recent backup files.

The Mirror Replication Agent retrieves the RASD backup files from the directory identified by the rasd_backup_dir configuration parameter. For more information about the rasd_backup_dir parameter, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

---

**Note** To restore the RASD, the Mirror Replication Agent instance must be in *Admin* state.

---

## ❖ To restore the RASD

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to determine the state of the Mirror Replication Agent instance:

   ```
   ra_status
   ```

3. If the Mirror Replication Agent is in *Admin* state, skip this step and continue with step 4.

   If the Mirror Replication Agent is in *Replicating* state:

   a. Use the following command to suspend replication by the Mirror Replication Agent instance:

      ```
      suspend
      ```

   b. Use the following command to verify that the Mirror Replication Agent is in *Admin* state:

      ```
      ra_status
      ```

4. After you verify that the Mirror Replication Agent is in *Admin* state, use the following command to restore the RASD:

   ```
   rasd_restore
   ```

   After the restore operation completes successfully, the Mirror Replication Agent returns a message to confirm that the RASD restore was successful.

   If the Mirror Replication Agent cannot find the directory identified in the rasd_backup_dir parameter, or if it cannot read the RASD backup files in that directory (for example, because of a permission problem), it returns an error. You must correct the cause of the error to restore the RASD.

5. After the RASD is successfully restored from the most recent backup, use the following command to resume replication in the Mirror Replication Agent instance:

```
resume
```

If the Mirror Replication Agent does not return to *Replicating* state, see Chapter 4, “Troubleshooting Mirror Replication Agent,” for more information.

## Truncating the RASD

To keep the RASD from growing indefinitely, you can periodically truncate older versions of its primary database object metadata.

---

**Note** Back up the RASD using rasd_backup *before* you truncate it. For more information, see “Backing up the RASD” on page 102.

---

The RASD stores definitions for two types of database objects:

- Articles – tables and stored procedures that are marked for replication.
- Users – database users who apply transactions in the primary database.

Use the ra_truncatearticles and ra_truncateusers commands to manage the size of the RASD.

You do not need to interrupt replication to truncate the RASD. You can truncate the RASD at any time, when the Mirror Replication Agent instance is in any state.

❖ **To truncate older versions of articles in the RASD**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the following command to truncate articles in the RASD:

   ```
   ra_truncatearticles NNN
   ```

   where *NNN* is an LTM Locator value that identifies the oldest non-current version of any article to be kept.

All non-current versions of all articles that are *less than* the LTM Locator value you specify are truncated from the RASD. If the current (most recent) version of an article is older than the version identified by the LTM Locator value, it is *not* truncated.

❖ **To truncate older versions of users in the RASD**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the following command to truncate users in the RASD:

```
ra_truncateusers NNN
```

where *NNN* is an LTM Locator value that identifies the oldest non-current version of any user to be kept.

All non-current versions of all users that are *less than* the LTM Locator value you specify are truncated from the RASD. If the current (most recent) version of a user is older than the version identified by the LTM Locator value, it is *not* truncated.

## Identifying replicated transactions and procedures

In a Sybase transaction replication system, the Mirror Replication Agent and Replication Server components both provide features that allow you to identify (or select) the transactions that you want to replicate. You do not need to replicate all transactions, or all data-changing operations, in the primary database.

The ability to select transactions for replication is particularly useful when you need to implement a replication system to support an application that uses some of the tables in a database, but not all of them.

By marking tables, you identify the specific tables in the primary database for which transactions are replicated. Transactions that affect the data in marked tables are referred to as *replicated transactions*.

**Note** If a transaction affects data in both marked and unmarked tables, only the operations that affect data in marked tables are replicated.

By marking stored procedures, you identify (or select) the specific procedures in the primary database that will be replicated as *applied functions*. When a marked procedure is invoked in the primary database, its invocation is replicated, along with its input parameter values, to the replicate database.

The ability to select procedures for replication is particularly useful when you need to implement a replication system to support an application that uses stored procedures, or when replicating a single procedure invocation is more efficient than replicating numerous, individual data-changing operations that are produced by a single procedure invocation.

Mirror Replication Agent provides the following features to allow you to select replicated transactions and procedures:

- Marking and unmarking tables
- Enabling and disabling replication for marked tables
- Enabling and disabling replication for LOB columns
- Marking and unmarking stored procedures
- Enabling and disabling replication for stored procedures
- Enabling and disabling replication for DDL
- Marking and unmarking sequences

**Note** By default, Mirror Replication Agent marks and enables DDL and all user tables and their LOB columns for replication when the Mirror Replication Agent is initialized using the ra_init command.

If this is not the default behavior desired, you can turn off automatic marking of tables using the pdb_automark_tables configuration parameter.

## Preparing to mark tables or stored procedures

Before you can mark tables or stored procedures for replication, you must create the Mirror Replication Agent transaction log objects.

See the following for more information:

- "Initializing the Mirror Replication Agent transaction log" on page 29
- "Managing the Replication Agent System Database" on page 95

## Marking and unmarking tables

To replicate transactions that affect the data in a table in the primary database, that table must be marked for replication, and replication must be enabled for the marked table.

Marking a table can be separate from enabling replication for that table. If the value of the pdb_dflt_object_repl parameter is true, replication is enabled automatically at the time a table is marked. For more information, see "Enabling and disabling replication for marked tables" on page 113.

---

**Note** By default, Mirror Replication Agent will mark and enable user tables and their LOB columns for replication when the Mirror Replication Agent is initialized using the ra_init command.

If this is not the default behavior desired, you can turn off automatic marking of tables using the pdb_automark_tables configuration parameter.

---

Table marking with Mirror Replication Agent

When a table is marked for replication, the Mirror Replication Agent does the following:

- Connects to the RASD
- Records the mark status for the table in the RASD Article for that table.

When a table is marked and enabled, any subsequent operations that affect the data in that table are replicated.

Table unmarking with Mirror Replication Agent

When you unmark a table marked for replication, the Mirror Replication Agent does the following:

- Connects to the RASD
- Records the unmark status for the table in the RASD Article for that table

When a table is unmarked, any subsequent operations that affect the data in that table are ignored (not replicated).

### Marking a table for replication

To mark a table for replication use the pdb_setreptable command. It returns replication marking status; marks all user tables or a specified table for replication; and enables replication for all marked tables or a specified table. The following is an example of the pdb_setreptable command that returns replication marking information for all marked tables in the primary database:

```
pdb_setreptable mark
```

Use the following procedure to mark tables for replication with Mirror Replication Agent.

### ❖ To mark a table in the primary database for replication

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the pdb_setreptable command to determine if the table is already marked:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the table that you want to mark for replication.

   If the pdb_setreptable command returns information that the specified table is marked for replication, you do not need to continue this procedure.

   If the pdb_setreptable command returns information that the specified table is not marked, continue this procedure to mark the table for replication.
3. Use the pdb_setreptable command to mark the table for replication.

   The pdb_setreptable command allows you to mark the primary table to be replicated and specify a different table name to use in the replicate database (as specified in a replication definition).

   - Use the following command to mark the table for replication when the replicate table (in the replicate database) has the same table name:

     ```
     pdb_setreptable pdb_table, mark
     ```

     where *pdb_table* is the name of the table that you want to mark for replication.
   - Use the following command to mark the table for replication when the replicate table (in the replicate database) has a different table name:

     ```
     pdb_setreptable pdb_table, rep_table, mark
     ```

     where:

     - *pdb_table* is the name of the table that you want to mark for replication.

  - *rep_table* is the name of the replicate table in the replicate database.

  ---

  **Note** When the primary table and replicate table have different names, you must use the with all tables named rep_table clause when you create the replication definition in the primary Replication Server.

  ---

- When marking a table for replication, you have the option to specify that the table owner's name is sent with the table name in the LTL:
  - If the owner mode is set, then the owner name is used when matching the repdef in Mirror Replication Agent.
  - If the owner mode is not set, then the owner name is not used by Mirror Replication Agent for repdef name matching

  To do this, use the owner keyword after the mark keyword, as shown in the following example:

  ```
  pdb_setreptable pdb_table, mark, owner
  ```

  where *pdb_table* is the name of the table that you want to mark for replication.

  ---

  **Note** The table owner's name returned from the primary database must be the same as the owner name specified in the replication definition for the table.

  ---

  If the value of the pdb_dflt_object_repl parameter is true, the table you marked for replication is ready for replication immediately after the pdb_setreptable command returns successfully.

  The default value of the pdb_dflt_object_repl parameter is true.

If the value of the pdb_dflt_object_repl parameter is true, you can skip step 4 in this procedure.

If the value of the pdb_dflt_object_repl parameter is false, you must enable replication for the table, as described in step 4.

4 Use the pdb_setreptable command to enable replication for a marked table:

```
pdb_setreptable pdb_table, enable
```

where *pdb_table* is the name of the marked table.

After replication is enabled for the table, the Mirror Replication Agent can begin replicating transactions that affect data in that table.

## Unmarking a table

To unmark a table for replication use the pdb_setreptable command. It returns replication marking status; unmarks all user tables or a specified table for replication; and disables replication for all marked tables or a specified table. The following is an example of the pdb_setreptable command that forces unmarking for all marked tables in the primary database:

```
pdb_setreptable all, unmark, force
```

Use the following procedure to unmark tables for replication with Mirror Replication Agent.

### ❖ To unmark a table in the primary database

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setreptable command to confirm that the table is marked in the primary database:

```
pdb_setreptable pdb_table
```

where *pdb_table* is the name of the table in the primary database that you want to unmark.

If the pdb_setreptable command returns information that the specified table is marked, continue this procedure to unmark the table.

If the pdb_setreptable command does not return information that the specified table is marked, you need not continue this procedure.

3 Use the pdb_setreptable command to disable replication from the table:

```
pdb_setreptable pdb_table, disable
```

where *pdb_table* is the name of the table in the primary database that you want to disable.

4 Use the pdb_setreptable command to remove the replication marking from the table:

```
pdb_setreptable pdb_table, unmark
```

where *pdb_table* is the name of the table in the primary database that you want to unmark.

If you need to force the unmark, you can use the following command:

```
pdb_setreptable pdb_table, unmark, force
```

5 Use the pdb_setreptable command to confirm that the table is no longer marked for replication:

```
pdb_setreptable pdb_table
```

where *pdb_table* is the name of the table in the primary database that you unmarked.

**Note** You can unmark all marked objects in the primary database by invoking the pdb_setreptable command with the all keyword.

## Enabling and disabling replication for DDL

**Note** For more information on the DDL commands that are not replicated, see Appendix C, "Mirror Replication Agent and Oracle Databases."

If you need to temporarily suspend replication of DDL, you can use the pdb_setrepddl command to disable replication of DDL. When you are ready to resume replication of DDL, you can use the pdb_setrepddl command to enable replication.

**Note** DDL replication is enabled, by default.

When you set the value of pdb_setrepddl to enable, all DDL in your primary database is replicated.

**Note** To replicate DDL, Replication Server must have a database-level replication definition with replicate DDL set in the definition. For details on creating a database-level replication definition, see the Mirror Replication Agent *Reference Manual*.

### Enabling replication for DDL

❖ **To enable replication for DDL in the primary database**

1 Log in to the Mirror Replication Agent administration port.

2. Use the pdb_setrepddl command without an argument to determine if replication is already enabled for DDL in the primary database:

   ```
   pdb_setrepddl
   ```

   If the pdb_setrepddl command returns information that replication is enabled, you do not need to continue this procedure.

   If the pdb_setrepddl command returns information that replication is not enabled for DDL, continue this procedure to enable replication for DDL.

3. Use the pdb_setrepddl command to enable replication for DDL:

   ```
   pdb_setrepddl enable
   ```

   After replication is enabled for the DDL, you can resume replicating your primary database.

   For details on enabling DDL replication, see Appendix C, “Mirror Replication Agent and Oracle Databases.”

## Disabling replication for DDL

❖ **To disable replication for DDL in the primary database**

1. Log in to the Mirror Replication Agent administration port.
2. Use the pdb_setrepddl command without an argument to determine if replication is already disabled for DDL in the primary database:

   ```
   pdb_setrepddl
   ```

   If the pdb_setrepddl command returns information that replication is disabled, you do not need to continue this procedure.

   If the pdb_setrepddl command returns information that replication is enabled for DDL, continue this procedure to disable replication for DDL.

3. Use the pdb_setrepddl command to disable replication for DDL:

   ```
   pdb_setrepddl disable
   ```

   After replication is disabled for the DDL, you can resume replicating your primary database.

   For details on enabling DDL replication see Appendix C, “Mirror Replication Agent and Oracle Databases.”

## Enabling and disabling replication for marked tables

If you need to temporarily stop replication for a marked table (for example, when maintenance operations are performed in the primary database), you can disable replication for a marked table without affecting replication for other tables in the primary database. Then, when you are ready to resume replication from that table, you can enable replication for that table without affecting other tables in the database.

To replicate transactions that affect the data in a table, that table must be marked for replication, and replication must be enabled for the marked table. For more information, see "Marking and unmarking tables" on page 107.

Mirror Replication Agent has *articles* in the RASD. An article is an object that has a one-to-one relationship to the tables and has a marked indicator.

When replication is disabled for a marked object, the marking infrastructure remains in place, but no transactions for that object are sent to Replication Server.

### Enabling replication for marked tables

Use the following procedure to enable replication for marked tables.

❖ **To enable replication for a marked table**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setreptable command to verify that replication is disabled for the table:

```
pdb_setreptable pdb_table
```

where *pdb_table* is the name of the marked table you want to enable replication for.

If the pdb_setreptable command returns information that the table is marked and has replication disabled, continue this procedure to enable replication for the table.

**Note** A table must be marked for replication before replication can be enabled or disabled for the table.

3 Use the pdb_setreptable command to enable replication for the table:

```
pdb_setreptable pdb_table, enable
```

where *pdb_table* is the name of the marked table in the primary database for which you want to enable replication.

After replication is enabled for the table, any transaction that affects the data in that table will be captured for replication.

4 You can use the pdb_setreptable command again to verify that replication is now enabled for the table:

```
pdb_setreptable pdb_table
```

where *pdb_table* is the name of the marked table for which you want to verify that replication is enabled.

## Disabling replication for marked tables

Use the following procedure to disable replication for marked tables.

❖ **To disable replication for a marked table**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setreptable command to verify that replication is enabled for the table:

```
pdb_setreptable pdb_table
```

where *pdb_table* is the name of the marked table for which you want to disable replication.

If the pdb_setreptable command returns information that the table is marked and has replication enabled, continue this procedure to disable replication for the table.

**Note** A table must be marked for replication before replication can be enabled or disabled for the table.

3 Use the pdb_setreptable command to disable replication for the table:

```
pdb_setreptable pdb_table, disable
```

where *pdb_table* is the name of the marked table in the primary database for which you want to disable replication.

After replication is disabled for the table, transactions that affect the data in that table will not be captured for replication until replication is enabled again.

4 You can use the pdb_setreptable command again to verify that replication is now disabled for the table:

```
pdb_setreptable pdb_table
```

where *pdb_table* is the name of the marked table for which you want to verify that replication is disabled.

## Enabling and disabling replication for LOB columns

In this document, all columns that contain large object (LOB) datatypes are referred to as LOB columns, regardless of the actual datatype name used by the primary database vendor. To replicate transactions that affect a LOB column, replication must be enabled for that column.

You must enable replication for each LOB column you want to replicate, in addition to marking and enabling replication for the table that contains the LOB column.

If the value of the pdb_dflt_column_repl parameter is true, replication is enabled automatically for all LOB columns in a table at the time the table is marked. If the value of the pdb_dflt_column_repl parameter is false, replication is not enabled automatically for any LOB columns in a table at the time the table is marked. For more information on marking a table for replication see “Marking and unmarking tables” on page 107.

**Note** The default value for pdb_dflt_column_repl is true, meaning that all LOB columns will be marked for replication when the table is marked, by default.

When a table is marked for replication and replication is enabled for that table but not for a LOB column in that table, any part of a transaction that affects the LOB column is not replicated. The portion of a transaction that affects all other non-LOB columns is replicated if the table is marked for replication and replication is enabled for the table.

Oracle logs all LOB data (except for *BFILE* datatypes) in the Oracle redo log. This allows the Mirror Replication Agent to apply each individual LOB change. However, BFILE data is not logged in the Oracle redo logs. Replication of BFILE columns is currently disabled.

For more information on LOB handling, see Appendix C, “Mirror Replication Agent and Oracle Databases.”

## Enabling replication for LOB columns

❖ **To enable replication for a LOB column in a marked table**

1. Log in to the Mirror Replication Agent instance with the administrator login.

2. Use the pdb_setrepcol command to verify that replication is disabled for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col
   ```

   where:

   - *pdb_table* is the name of the marked table that contains the LOB column.
   - *pdb_col* is the name of the LOB column.

   If the pdb_setrepcol command returns information that the LOB column has replication disabled, continue this procedure to enable replication for the column.

   **Note** The table that contains the LOB column must be marked for replication before replication can be enabled or disabled for a LOB column.

3. Use the pdb_setrepcol command to enable replication for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col, enable
   ```

   where:

   - *pdb_table* is the name of the marked table that contains the LOB column.
   - *pdb_col* is the name of the LOB column for which you want to enable replication.

   After replication is enabled for the LOB column (and if replication is enabled for the table that contains the column), any transaction that affects the data in that column will be replicated.

4. You can use the pdb_setrepcol command again to verify that replication is now enabled for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col
   ```

   where:

- *pdb_table* is the name of the marked table that contains the LOB column.
- *pdb_col* is the name of the LOB column for which you want to verify that replication is enabled.

## Disabling replication for LOB columns

❖ **To disable replication for a LOB column in a marked table**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepcol command to verify that replication is enabled for the LOB column:

```
pdb_setrepcol pdb_table, pdb_col
```

where:

- *pdb_table* is the name of the marked table that contains the LOB column.
- *pdb_col* is the name of the LOB column you want to disable replication for.

If the pdb_setrepcol command returns information that the LOB column has replication enabled, continue this procedure to disable replication for the column.

**Note** The table that contains the LOB column must be marked for replication before replication can be enabled or disabled for a LOB column.

3 Use the pdb_setrepcol command to disable replication for the LOB column:

```
pdb_setrepcol pdb_table, pdb_col, disable
```

where:

- *pdb_table* is the name of the marked table that contains the LOB column.
- *pdb_col* is the name of the LOB column for which you want to disable replication.

After replication is disabled for the LOB column, transactions that affect the data in that column will not be replicated unless replication is enabled for that column again.

4 You can use the pdb_setrepcol command again to verify that replication is now disabled for the LOB column:

```
pdb_setrepcol pdb_table, pdb_col
```

where:

- *pdb_table* is the name of the marked table that contains the LOB column.
- *pdb_col* is the name of the LOB column for which you want to verify that replication is disabled.

## Marking and unmarking stored procedures

Mirror Replication Agent supports Replication Server function replication by replicating the invocation of stored procedures in the primary database.

**Note** In this document, the terms *function* and *stored procedure* are synonyms.

Mirror Replication Agent can replicate both *applied functions* and *request functions*:

- Applied functions are stored procedures that are executed in the primary database and generate transactions that affect data in the primary database.
- Request functions are stored procedures that are invoked in one database (for example, a replicate database), then executed in another database (for example, a primary database).

Mirror Replication Agent does not distinguish between these two function types, except to supply a specific user and password for use with request functions. If you are using request functions, the configuration parameters function_username and function_password must be supplied. For more information about applied and request functions, see the Managing Replicated Functions chapter of the Replication Server *Administration Guide*. For more information about the function_username and function_password configuration parameters, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

In order to replicate a stored procedure invoked in a primary database, the stored procedure must be marked for replication, and replication must be enabled for that stored procedure. (This is analogous to marking and enabling replication for tables.)

---

**Note** Marking a stored procedure for replication is separate from enabling replication for the stored procedure. If the value of the pdb_dflt_object_repl parameter is true, replication is enabled automatically at the time a stored procedure is marked. For more information on enabling and disabling replication for a marked stored procedure, see "Enabling and disabling replication for stored procedures" on page 123.

---

If a marked stored procedure performs operations that affect a marked table, the operations that affect the marked table are not captured for replication; only the invocation of the marked stored procedure is replicated.

When you mark a stored procedure for replication, Mirror Replication Agent creates a shadow-row procedure for that stored procedure.

Mirror Replication Agent also modifies the marked stored procedure as follows:

- Inserts a new first step to execute the associated shadow-row procedure
- Inserts a new last step to again execute the shadow-row procedure with different parameters.

If you need to temporarily suspend replication of a marked stored procedure (for example, when database maintenance operations are performed in the primary database), you can disable replication for the stored procedure. For more information, see "Enabling and disabling replication for stored procedures" on page 123.

When you unmark an object that has been marked for replication, the transaction log objects that were created to facilitate the replication for that object are removed from the primary database.

For more information on the Replication Server function replication feature, see the Replication Server *Administration Guide*.

## Marking a stored procedure for replication

When you mark a stored procedure for replication, Mirror Replication Agent creates the transaction log objects that capture the stored procedure invocation in the transaction log.

---

**Note** DDL replication must be disabled during the marking of stored procedures. Because marking of a stored procedure modifies that stored procedure, you must first disable DDL replication to prevent the marking modifications from replicating to the standby site. See “Disabling replication for DDL” on page 112.

---

❖ **To mark a stored procedure for replication**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepproc command to determine if the stored procedure is already marked in the primary database:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.

If the pdb_setrepproc command returns information that the specified stored procedure is marked, you do not need to continue this procedure.

If the pdb_setrepproc command returns information that the specified stored procedure is not marked, continue this procedure to mark the stored procedure for replication.

3 Use the pdb_setrepproc command to mark the stored procedure for replication.

The pdb_setrepproc command allows you to mark the primary stored procedure to be replicated and specify a different stored procedure name to use in the replicate database (as specified in a function replication definition).

- Use the following command to mark the stored procedure for replication using a function replication definition with the same stored procedure name:

  ```
  pdb_setrepproc pdb_proc, mark
  ```

  where *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.

- Use the following command to mark the stored procedure for replication using a function replication definition with a different stored procedure name:

  ```
  pdb_setrepproc pdb_proc, rep_proc, mark
  ```

  where:

  - *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.
  - *rep_proc* is the name of the stored procedure in the function replication definition for this stored procedure.

  If the value of the pdb_dflt_object_repl parameter is true, the stored procedure marked for replication with the pdb_setrepproc command is ready for replication after you invoke the pdb_setrepproc command successfully.

If the value of the pdb_dflt_object_repl parameter is true (the default value), you can skip step 4 in this procedure.

If the value of the pdb_dflt_object_repl parameter is false, you must enable replication for the stored procedure before replication can take place.

4 Use the pdb_setrepproc command to enable replication for the marked stored procedure:

```
pdb_setrepproc pdb_proc, enable
```

where *pdb_proc* is the name of the marked stored procedure for which you want to enable replication.

After replication is enabled for the stored procedure, you can begin replicating invocations of that stored procedure in the primary database.

**Note** If you disabled DDL replication during stored procedure marking, remember to re-enable DDL replication. Since marking of a stored procedure modifies that stored procedure, you must first disable DDL replication to prevent the marking modifications from replicating to the standby site. See “Enabling replication for DDL” on page 111.

## Unmarking a stored procedure

When you unmark a stored procedure, Mirror Replication Agent removes the transaction log objects that were created when the stored procedure was marked.

---

**Note** DDL replication must be disabled during the unmarking of stored procedures. See "Disabling replication for DDL" on page 112.

---

### ❖ To unmark a stored procedure

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the pdb_setrepproc command to confirm that the stored procedure is marked in the primary database:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the stored procedure that you want to unmark.

   If the pdb_setrepproc command returns information that the specified stored procedure is marked, continue this procedure to unmark the stored procedure.

   If the pdb_setrepproc command does not return information that the specified stored procedure is marked, you do not need to continue this procedure.
3. Use the pdb_setrepproc command to disable replication of the stored procedure:

   ```
   pdb_setrepproc pdb_proc, disable
   ```

   where *pdb_proc* is the name of the stored procedure that you want to unmark.
4. Use the pdb_setrepproc command to remove the replication marking from the stored procedure:

   ```
   pdb_setrepproc pdb_proc, unmark
   ```

   where *pdb_proc* is the name of the stored procedure that you want to unmark.

   If you need to force the unmark, you can use the following command:

   ```
   pdb_setrepproc pdb_proc, unmark, force
   ```

5 Use the pdb_setrepproc command to confirm that the stored procedure is no longer marked for replication:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the stored procedure in the primary database that you unmarked.

You can unmark all marked stored procedures in the primary database by invoking the pdb_setrepproc command with the all keyword.

**Note** If you disabled DDL replication during stored procedure unmarking, remember to re-enable DDL replication. See "Enabling replication for DDL" on page 111.

## Enabling and disabling replication for stored procedures

If you need to temporarily suspend replication of a stored procedure, you can use the pdb_setrepproc command to disable replication for the marked stored procedure. When you are ready to resume replication of the marked stored procedure, you can use the pdb_setrepproc command to enable replication.

**Note** No procedures are marked by default for replication.

To replicate invocations of a stored procedure in the primary database, the stored procedure must be marked for replication and replication must be enabled for that stored procedure.

Marking a stored procedure for replication is separate from enabling replication for the stored procedure. For more information on marking a stored procedure for replication, see "Marking and unmarking stored procedures" on page 118.

### Enabling replication for stored procedures

❖ **To enable replication for a marked stored procedure**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepproc command to verify that replication is disabled for the stored procedure:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the marked stored procedure you want to enable replication for.

If the pdb_setrepproc command returns information that the stored procedure is marked and has replication disabled, continue this procedure to enable replication for the stored procedure.

**Note** A stored procedure must be marked for replication before replication can be enabled or disabled for the stored procedure.

3. Use the pdb_setrepproc command to enable replication for the stored procedure:

   ```
   pdb_setrepproc pdb_proc, enable
   ```

   where *pdb_proc* is the name of the marked stored procedure for which you want to enable replication.

   After replication is enabled for the stored procedure, any invocation of that stored procedure will be replicated.

4. You can use the pdb_setrepproc command again to verify that replication is now enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the marked stored procedure for which you want to verify that replication is enabled.

## Disabling replication for stored procedures

❖ **To disable replication for a marked stored procedure**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the pdb_setrepproc command to verify that replication is enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the marked stored procedure you want to disable replication for.

If the pdb_setrepproc command returns information that the stored procedure is marked and has replication enabled, continue this procedure to disable replication for the stored procedure.

---

**Note** A stored procedure must be marked for replication before replication can be enabled or disabled for that stored procedure.

---

3 Use the pdb_setrepproc command to disable replication for the stored procedure:

```
pdb_setrepproc pdb_proc, disable
```

where *pdb_proc* is the name of the marked stored procedure for which you want to disable replication.

After replication is disabled for the stored procedure, any invocation of that stored procedure will not be captured for replication until replication is enabled again.

4 You can use the pdb_setrepproc command again to verify that replication is now disabled for the stored procedure:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the marked stored procedure for which you want to verify that replication is disabled.

## Marking and unmarking sequences

Mirror Replication Agent supports replication of sequences in the primary database. In order to replicate a sequence invoked in a primary database, the sequence must be marked for replication and replication must beanball for that sequence. (This is analogous to marking and enabling replication for tables.)

---

**Note** Marking a sequence for replication is separate from enabling replication for the sequence. If the value of the pdb_dflt_object_repl parameter is true, replication is enabled automatically at the time a sequence is marked. For more information on enabling and disabling replication for a marked sequence, see “Enabling and disabling replication for “Enabling and disabling replication for sequences” on page 129.

---

Oracle does not log information every time a sequence is incremented. Sequence replication occurs when the Mirror Replication Agent captures the system table updates that occur when the sequence's cache is refreshed. Therefore, the sequence value replicated when a sequence is marked for replication is the 'next' sequence value to be used when the current cache expires. The result is not every individual increment of a sequence is replicated, but the standby site will always have a value greater than the primary site's currently available cached values. For more information on Sequence replication, see Appendix C, Mirror Replication Agent and Oracle Databases.

If you need to temporarily suspend replication of a marked sequence you can disable replication for the sequence. For more information see "Enabling and disabling replication for “Marking and unmarking sequences” on page 125.

## Marking a sequence for replication

❖ **To mark a sequence for replication**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepseq command to determine if the sequence is already marked in the primary database:

```
pdb_setrepseq pdb_seq
```

where pdb_seq is the name of the sequence in the primary database that you want to mark for replication.

If the pdb_setrepseq command returns information that the specified sequence is marked, you need not continue this procedure.

If the pdb_setrepseq command returns information that the specified sequence is not marked, continue this procedure to mark the sequence for replication.

❖ **To mark a stored procedure for replication**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepproc command to determine if the stored procedure is already marked in the primary database:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.

If the pdb_setrepproc command returns information that the specified stored procedure is marked, you need not continue this procedure.

If the pdb_setrepproc command returns information that the specified stored procedure is not marked, continue this procedure to mark the stored procedure for replication.

3 Use the pdb_setrepseq command to mark the sequence for replication.

The pdb_setrepseq command allows you to mark the primary stored procedure to be replicated and specify a different sequence name to use in the replicate database.

- Use the following command to mark the sequence for replication when the sequence name you wish to increment at the standby site has a different name:

  ```
  pdb_setrepseq pdb_seq, mark
  ```

  where *pdb_seq* is the name of the sequence in the primary database that you want to mark for replication.

  ---

  **Note** Replicating a sequence with a different name is provided to be consistenct with other marking commands, but is not a typical configuration.

  ---

- Use the following command to mark the sequence for replication using a different sequence name:

  ```
  pdb_setrepseq pdb_seq, rep_seq, mark
  ```

  where:

  - *pdb_seq* is the name of the sequence in the primary database that you want to mark for replication.
  - *rep_seq* is the name of the sequence in the standby database that you wish to increment.

  ---

  **Note** Replicating sequence values to a sequence with a different name at the standby site assumes the standby site's sequence has the same attributes and starting value as the primary site's sequence.

  ---

If the value of the pdb_dflt_object_repl parameter is true, the sequence marked for replication with the pdb_setrepseq command is ready for replication after you invoke the pdb_setrepseq command successfully.

If the value of the pdb_dflt_object_repl parameter is true (the default value), you can skip step 4 in this procedure.

If the value of the pdb_dflt_object_repl parameter is false, you must enable replication for the sequence before replication can take place.

4 Use the pdb_setrepseq command to enable replication for the marked sequence:

```
pdb_setrepseq pdb_seq, enable
```

where *pdb_seq* is the name of the marked sequence for which you want to enable replication.

After replication is enabled for the sequence, you can begin replicating invocations of that sequence in the primary database.

## Unmarking a sequence

❖ **To unmark a sequence**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepseq command to confirm that the sequence is marked in the primary database:

```
pdb_setrepseq pdb_seq
```

where pdb_seq is the name of the sequence that you want to unmark.

If the pdb_setrepseq command returns information that the specified sequence is marked, continue this procedure to unmark the sequence.

If the pdb_setrepseq command does not return information that the specified sequence is marked, you need not continue this procedure.

3 Use the pdb_setrepseq command to disable replication of the sequence:

```
pdb_setrepseq pdb_seq, disable
```

where *pdb_proc* is the name of the sequence that you want to unmark.

4 Use the pdb_setrepseq command to remove the replication marking from the sequence:

```
pdb_setrepseq pdb_seq, unmark
```

where *pdb_seq* is the name of the sequence that you want to unmark.

If you need to force the unmark, you can use the following command:

```
pdb_setrepseq pdb_seq, unmark, force
```

5 Use the pdb_setrepseq command to confirm that the sequence is no longer marked for replication:

```
pdb_setrepseq pdb_seq
```

where *pdb_seq* is the name of the sequence in the primary database that you unmarked.

## Enabling and disabling replication for sequences

If you need to temporarily suspend replication of a sequence, you can use the pdb_setrepseq command to disable replication for the marked sequence. When you are ready to resume replication of the marked sequence, you can use the pdb_setrepseq command to enable replication.

**Note** No sequences are marked by default for replication.

To replicate updates of a sequence in the primary database, the sequence must be marked for replication and replication must be enabled for that sequence.

Marking a sequence for replication is separate from enabling replication for the sequence. For more information on marking a sequence for replication, see “Marking a sequence for replication” on page 126.

### Enabling replication for sequences

❖ **To enable replication for a marked sequence**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the pdb_setrepseq command to verify that replication is disabled for the sequence:

```
pdb_setrepseq pdb_seq
```

where *pdb_seq* is the name of the marked sequence you want to enable replication for.

If the pdb_setrepseq command returns information that the sequence is marked and has replication disabled, continue this procedure to enable replication for the sequence.

---

**Note** A sequence must be marked for replication before replication can be enabled or disabled for the sequence.

---

3. Use the pdb_setrepseq command to enable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, enable
   ```

   where *pdb_seq* is the name of the marked sequence for which you want to enable replication.

   After replication is enabled for the sequence, any invocation of that sequence will be replicated.

4. You can use the pdb_setrepseq command again to verify that replication is now enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the marked sequence for which you want to verify that replication is enabled.

## Disabling replication for marked sequence

❖ **To disable replication for a marked sequence**

1. Log in to the Mirror Replication Agent instance with the administrator login.

2. Use the pdb_setrepseq command to verify that replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the marked sequence you want to disable replication for.

   If the pdb_setrepseq command returns information that the sequence is marked and has replication enabled, continue this procedure to disable replication for the sequence.

---

**Note** A sequence must be marked for replication before replication can be enabled or disabled for that sequence.

---

3 Use the pdb_setrepseq command to disable replication for the sequence:

```
pdb_setrepseq pdb_seq, disable
```

where *pdb_seq* is the name of the marked sequence for which you want to disable replication.

After replication is disabled for the sequence, any invocation of that sequence will not be captured for replication until replication is enabled again.

4 You can use the pdb_setrepseq command again to verify that replication is now disabled for the sequence:

```
pdb_setrepseq pdb_seq
```

where *pdb_seq* is the name of the marked sequence for which you want to verify that replication is disabled.

## Configuring and tuning the Mirror Replication Agent

The performance of Mirror Replication Agent can be tuned or optimized by adjusting some of the Mirror Replication Agent configuration parameters.

You can set or change a Mirror Replication Agent configuration parameter with the ra_config command.

Because the Mirror Replication Agent overwrites its entire configuration file whenever ra_config or ra_set_login is invoked, Sybase recommends that you do *not* edit the configuration file. Furthermore, each Mirror Replication Agent instance reads its configuration file only at start-up. You must use the ra_config command if you want a new configuration parameter value to take effect before the instance is shut down and restarted.

**Note** Some configuration parameter changes are recorded in the configuration file when you invoke ra_config, but they do not take effect until the Mirror Replication Agent instance is shut down and restarted.

All Mirror Replication Agent configuration parameters can be changed when the Mirror Replication Agent instance is in *Admin* state. Some configuration parameters *cannot* be changed when the instance is in *Replicating* state.

Table 3-1 lists all Mirror Replication Agent configuration parameters.

*Table 3-1: Mirror Replication Agent configuration parameters*

| Parameter name | Description | Default |
|---|---|---|
| admin_port | Port number on which Mirror Replication Agent listens for incoming user connections. | 10000 |
| column_compression | Use minimal column information. | true |
| compress_ltl_syntax | Use abbreviated LTL syntax. | true |
| connect_to_rs | Enable/disable connection from LTI to Replication Server. | true |
| ddl_password | Password for user ID passed in LTL with replicated DDL invocations. | <not_configured> |
| ddl_username | User ID passed in LTL with replicated DDL invocations. | <not_configured> |
| dump_batch_timeout | Time to send incomplete buffer to Replication Server. | 5 |
| filter_maint_userid | Log Reader filters operations with maintenance user ID. | true |
| function_password | Password for user ID passed in LTL with replicated stored procedure invocations. | ““ (empty string) |
| function_username | User ID passed in LTL with replicated stored procedure invocations. | sa |
| log_backup_files | Determines the number of log backup files kept in the log directory. | 3 |
| log_directory | Directory where Mirror Replication Agent system log file is located. | Current directory on the Mirror Replication Agent host machine from which the Mirror Replication Agent instance was started. |
| log_trace_verbose | Switch on/off verbose mode in trace log file. | false |
| log_wrap | Number of 1KB blocks written to log file before wrapping. | 1000 |
| lti_batch_mode | Specifies whether to batch the LTL commands sent to Replication Server. | true |

| Parameter name | Description | Default |
|---|---|---|
| lti_max_buffer_size | Maximum number of change sets stored in the LTI input buffer. | 5000 |
| lti_update_trunc_point | Number of LTL commands sent before LTI requests new LTM Locator. | 1000 |
| ltl_batch_size | Size of the LTL batch buffer. | 40000 |
| ltl_character_case | Case of database object names sent to Replication Server. | asis |
| ltl_origin_time_required | Specifies whether to send origin_time command tag in LTL. | false |
| ltm_admin_pw | Password for Mirror Replication Agent administrative port. | ““ (empty string) |
| ltm_admin_user | User ID for Mirror Replication Agent administrative port. | sa |
| max_ops_per_scan | Maximum number of operations Log Reader will read in a single log scan. | 1000 |
| pdb_auto_create_repdefs | Automatic creation of Replication Definitions when tables or procs are marked. | true |
| pdb_automark_tables | Automatic marking of user tables when the Replication Agent is initialized. | true |
| pdb_auto_run_scripts | Automatic execution of SQL scripts used to create/remove transaction log objects and mark/unmark primary database objects. | true |
| pdb_convert_datetime | Converts native date/time formats to Sybase datetime format. | false |
| pdb_dflt_column_repl | Enables replication for LOB columns by default when table is marked. | true |
| pdb_dflt_object_repl | Enables replication by default when object is marked. | true |
| pdb_xlog_device | Name of the primary database device. | NULL |
| pdb_xlog_prefix | Character string prefix used to identify transaction log objects. | ra_ |
| pdb_xlog_prefix_chars | Non-alphabetic characters allowed in pdb_xlog_prefix. | _#$ |

| Parameter name | Description | Default |
|---|---|---|
| pds_connection_type | Type of connection to primary data server. | ORAJDBC |
| pds_database_name | Name of database replicated from primary data server. | <not_configured> |
| pds_datasource_name | Data source name of database replicated from primary data server. | <not_configured> |
| pds_host_name | Name of primary data server host machine. | <not_configured> |
| pds_password | Password for user ID Mirror Replication Agent uses to access primary data server. | ““ (empty string) |
| pds_port_number | Port number for primary data server. | 1111 |
| pds_retry_count | Number of times to retry connection to primary data server. | 5 |
| pds_retry_timeout | Number of seconds to wait between connection retry attempts. | 10 |
| pds_server_name | Server name of primary data server. | <not_configured> |
| pds_username | User ID that Mirror Replication Agent uses to access primary data server. | <not_configured> |
| ra_retry_count | Number of times LTM attempts to get back to *Replicating* state after a failure. | 2 |
| ra_retry_timeout | Number of seconds to wait between LTM attempts to get back to *Replicating* state. | 10 |
| rs_charset | Character set used to communicate with Replication Server. | ““ (empty string) |
| rs_host_name | Name of primary Replication Server host machine. | <not_configured> |
| rs_packet_size | Network I/O packet size sent to Replication Server. | 2048 |
| rs_password | Password for user ID Mirror Replication Agent uses to access Replication Server. | ““ (empty string) |

| Parameter name | Description | Default |
|---|---|---|
| rs_port_number | Port number for primary Replication Server. | 1111 |
| rs_retry_count | Number of times to retry connection to primary Replication Server. | 5 |
| rs_retry_timeout | Number of seconds to wait between connection retry attempts. | 10 |
| rs_source_db | Name of primary database identified to Replication Server. | <not_configured> |
| rs_source_ds | Name of primary data server identified to Replication Server. | <not_configured> |
| rs_username | User ID that Mirror Replication Agent uses to access primary Replication Server. | <not_configured> |
| rssd_charset | Character set used to communicate with RSSD. | ““ (empty string) |
| rssd_database_name | Name of RSSD database. | <not_configured> |
| rssd_host_name | Name of RSSD host machine. | <not_configured> |
| rssd_password | Password for user ID Mirror Replication Agent uses to access RSSD. | ““ (empty string) |
| rssd_port_number | Port number for RSSD. | 1111 |
| rssd_username | User ID that Mirror Replication Agent uses to access RSSD. | <not_configured> |
| scan_sleep_increment | Number of seconds to increase Log Reader wait before next scan after finding no operations to replicate. | 5 |
| scan_sleep_max | Maximum number of seconds for Log Reader to wait before next scan after finding no operations to replicate. | 60 |
| skip_ltl_errors | LTI ignores error messages returned by Replication Server. | false |
| structured_tokens | LTI uses structured tokens when generating LTL output. | true |
| truncation_interval | Number of minutes to wait between automatic log truncations. | 0 |

| Parameter name | Description | Default |
|---|---|---|
| truncation_type | Method of log truncation allowed. | locator_update<br>interval |
| use_rssd | Switches on/off access to RSSD for replication definitions. | true |

For more information about the ra_config command and Mirror Replication Agent configuration parameters, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

## Configuring Mirror Replication Agent

To set or change a Mirror Replication Agent configuration parameter, use the ra_config command.

Because the Mirror Replication Agent overwrites its entire configuration file whenever ra_config or ra_set_login is invoked, Sybase recommends that you do *not* edit the configuration file. Furthermore, Mirror Replication Agent reads the configuration file only at start-up. You must use the ra_config command if you want a new configuration parameter value to take effect before the Mirror Replication Agent is shut down and restarted.

**Note** Some configuration parameter changes are recorded in the configuration file when you invoke ra_config, but do not take effect until the Mirror Replication Agent is shut down and restarted.

## Customizing tuning

Generally, Mirror Replication Agent's default configuration values provide optimal performance. However, there may be certain situations where the configuration should be changed to suit or optimize your particular environment.

### Adjusting the size and volume of the Mirror Replication Agent system logs

By default, the system logs produced by the Mirror Replication Agent are a pre-set size. They roll over occasionally to prevent continual disk consumption.

You can adjust the size of a log and adjust the number of backup files. By increasing these sizes, you can save log data for a longer period of time. By decreasing them, you can increase the unused space in your environment.

❖ **To adjust the size and volume of log files**

1. Log in to the running Mirror Replication Agent instance using the administrator login.
2. Use the ra_status command to verify that the Mirror Replication Agent instance is in *Admin* state:

   ```
   ra_status
   ```

3. Increase the following values if you want to increase the size and number of backup files. Decrease the following values if you want to make more space available in your environment. Use the ra_config command to set the values of the following Mirror Replication Agent configuration parameters for the primary database:

   ra_config log_backup_files, *n*

   ra_config log_wrap, *m*

**Preventing continual spinning at the end of a log scan**

Mirror Replication Agent uses the configuration parameters scan_sleep_increment and scan_sleep_max to “pause” scanning when the end of the log is reached. This prevents Mirror Replication Agent from continually “spinning” on the end of the log. The downside is that Mirror Replication Agent may pause up to 60 seconds (by default) before a new transaction appears, because it was sleeping. When you need the maximum possible latency for a transaction to be less than the 60-second default, the scan parameters can be reduced. This results in additional CPU usage when the end of the log is reached.

Conversely, if CPU maximization is a greater concern than latency, you can increase these parameters to allow Mirror Replication Agent to use less CPU on an inactive log, at the cost of having the latency of the ‘next’ transaction increased.

**Note** These parameters have effect *only* when the end of the log has been reached and there is no additional activity to be replicated. By default, Mirror Replication Agent immediately re-scans (without pause) when the end of the log has not been reached.

CHAPTER 4

# Troubleshooting Mirror Replication Agent

This chapter describes basic troubleshooting procedures for Mirror Replication Agent and the Mirror Activator for Oracle system.

| Topic | Page |
|---|---|
| Diagnosing command errors and replication errors | 139 |
| Troubleshooting specific command errors | 140 |
| Examining the Mirror Replication Agent when replication failure occurs | 140 |
| Checking the Replication Server | 149 |

## Diagnosing command errors and replication errors

Problems that prevent replication from starting do not always result in an error message. For example, a Mirror Activator for Oracle component may not recognize a configuration problem that prevents replication from starting.

In a functioning Mirror Activator for Oracle system—one that has previously replicated transactions successfully—most system problems result in an error message from one or more of the system's components. However, some problems that interrupt replication might not be interpreted as errors by the system's components. In that case, replication fails, but no error message is returned.

Use the diagnostic and troubleshooting tips in the following sections to identify and correct the cause of a Mirror Activator for Oracle system problem:

- Troubleshooting specific command errors
- Examining the Mirror Replication Agent when replication failure occurs
- Checking the Replication Server

## Troubleshooting specific command errors

This section describes troubleshooting for specific errors you may encounter in a Mirror Replication Agent. The following error message can be returned from a command or appear in the log file.

### Connection refused

Error message

```
Could not connect to <jdbc:sybase:Tds:localHost:5001/emb>:
JZ006: Caught IOException: java.net.ConnectException:
Connection refused: connectJZ006:
```

Explanation

The Mirror Replication Agent attempted to connect to a Sybase server on a host called *localHost* and port 5001. The error indicates no server was found.

Action

This is a usually a configuration error. Either the server that Mirror Replication Agent is attempting to connect to is not running, or the host and port information configured in Mirror Replication Agent is incorrect.

- Verify that your server is running.
- Verify that your Mirror Replication Agent is configured with the correct host and port information. See "Setting up the connection configuration parameters" on page 72 for more information. Test your connection after you have verified them. For more information, see "Testing network connectivity" on page 77.

## Examining the Mirror Replication Agent when replication failure occurs

When no transactions appear to be replicated to the replicate database, and you receive specific error messages, see "Troubleshooting specific command errors" on page 140.
When no errors are returned by any Mirror Activator for Oracle system components:

- Verify that primary database objects are marked for replication
- Check the Mirror Replication Agent status
- Examine the Mirror Replication Agent logs

- Use the ra_statistics command to troubleshoot

## Verify that primary database objects are marked for replication

In a Sybase transaction replication system, the Mirror Replication Agent and Replication Server components both provide features that allow you to select the objects that you want to replicate. You need not replicate all objects, or all data-changing operations, in the primary database.

If a primary database object, such as a table or stored procedure, is not replicating, verify the object you intended to replicate is marked.

❖ **To verify that a primary database object is marked for replication**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the appropriate command to determine if the object is already marked:

- For a table:

  ```
  pdb_setreptable pdb_table
  ```

  where *pdb_table* is the name of the table that you want to verify is marked for replication.

- For a LOB column:

  ```
  pdb_setrepcol pdb_table, pdb_col
  ```

  where:

  - *pdb_table* is the name of the marked table that contains the LOB column.
  - *pdb_col* is the name of the LOB column.

- For a stored procedure:

  ```
  pdb_setrepproc pdb_proc
  ```

  where *pdb_proc* is the name of the stored procedure in the primary database that you want to verify is marked for replication.

- For DDL:

  pdb_setrepddl *pbd_ddl*

where *pdb_ddl* is the name of the DDL in the primary database that you want to verify is marked for replication.

After you verify that the primary database objects are marked, see the following table:

| If... | Then... |
|---|---|
| The primary database object is not marked | Mark the object:<br>• Table – see "Marking a table for replication" on page 107.<br>• LOB column – see "Enabling replication for LOB columns" on page 116.<br>• Stored procedure – see "Marking and unmarking stored procedures" on page 118.<br>• DDL – see "Enabling and disabling replication for DDL" on page 111.<br>• Sequence - see "Marking and unmarking sequences" on page 125. |
| The primary database object is marked | Check the Mirror Replication Agent status. See "Check the Mirror Replication Agent status" on page 142. |

## Check the Mirror Replication Agent status

The status of the Mirror Replication Agent instance indicates whether it is in *Replicating* state, or in *Admin* state:

No replication takes place when the Mirror Replication Agent instance is in *Admin* state. For more information, see"Understanding Mirror Replication Agent states" on page 86.

### ❖ To check the current Mirror Replication Agent status

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Use the following command to check the current status of the Mirror Replication Agent:

```
ra_status
```

This command returns the current state of the Mirror Replication Agent instance, as shown in the following example:

```
State  Action
------ ----------------------------
ADMIN  Waiting for operator command
(1 row affected)
```

For more information about the ra_status command, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

When the Mirror Replication Agent instance is in one of the following states, take the suggested actions.

| **If...** | **Then...** |
|---|---|
| The Mirror Replication Agent instance is in *Replicating (Waiting at end of log)* state | Examine the statistics output to check the progress of the replication process. See "Use the ra_statistics command to troubleshoot" on page 145 for more information. |
| The Mirror Replication Agent instance is *Replicating* state<br><br>**Note** When the Mirror Replication Agent instance is in the *Replicating* state, all data may *not* have yet been replicated. You can only be sure that the Mirror Replication Agent instance is finished replicating when the state is *Replicating (Waiting at end of log)*. | The instance is operating normally, but it has not reached the end of the transaction log. Wait until Mirror Replication Agent instance is in *Replicating (Waiting at end of log)* state. Then repeat the procedure on page 142. |
| The Mirror Replication Agent instance is in *Admin* state | Start replication and put the Mirror Replication Agent instance in *Replicating* state by executing the Mirror Replication Agent resume command. See "Starting replication in the Mirror Replication Agent" on page 91 for more information.<br><br>If the Mirror Replication Agent instance returns to *Admin* state after you invoke the resume command, there is at least one unresolved problem that prevents the instance from going to *Replicating* state. See "Examine the Mirror Replication Agent logs" on page 144 for more information. |

## Examine the Mirror Replication Agent logs

The Mirror Replication Agent system log files contain warning and error messages, as well as information about the Mirror Replication Agent connections to the primary database and the Replication Server. Look for the most recent command you executed at the bottom of the log file to find the most recent message. The logs are located in the *$SYBASE/MRO-12_6/inst_name/log* directory.

The following is sample output from an instance's log file:

```
W.     2005/04/26 11:33:23.075  OracleLogScanne
com.sybase.ds.oracle.log.OracleLogScanner    scanForward 23      The change
vector list for log record <00001610.000002d2.0170> is empty.

W.     2005/04/26 11:33:43.313  OracleLogScanne
com.sybase.ds.oracle.log.OracleLogScanner    scanForward 23      The change
vector list for log record <00001610.00000483.011c> is empty.

W.     2005/04/26 11:33:47.879  OracleLogScanne
com.sybase.ds.oracle.log.OracleLogScanner    scanForward 23      The change
vector list for log record <00001610.000004f7.012c> is empty.

T.     2005/04/26 11:35:28.867  OracleLogScanne
com.sybase.ds.oracle.log.OracleLogScanner    scanForward 23      Moving to log
<5649>.

E.     2005/04/26 11:35:30.359  ERROR
com.sybase.ds.oracle.log.record.RecordFactoryparseLogRecord 23
com.sybase.ds.oracle.record.UnknownRecordException: Unkown CVxE_4 inner op
type: <63>.

E.     2005/04/26 11:35:30.359  ERROR
com.sybase.ds.oracle.log.record.RecordFactoryparseLogRecord 23
java.lang.RuntimeException:
com.sybase.ds.oracle.record.UnknownRecordException: Unkown CVxE_4 inner op
type: <63>.

E.     2005/04/26 11:35:30.359  ERROR
com.sybase.ds.oracle.log.record.RecordFactoryparseLogRecord 23
com.sybase.ds.oracle.log.record.RecordFactory.createChangeVector(RecordFactor
y.java:430)
```

where:

- The first column displays a single character indicating the type of message:
    - I = information
    - W = warning

    - E = error
    - T = trace
    - S = severe
- The second column is a time stamp indicating when the message was written.
- The third column is a description.
- The fourth column identifies the Java class that produced the error.

**Note** The following two columns appear only when configuration property log_trace_verbose is set to TRUE.

- The fifth column includes the method.
- The sixth column includes the line number.
- The final column is a text description of the message.

**Note** In some cases, the information in a specific column will not be consistent with these descriptions. In these cases, other information is generated that is used by technical support to determine where the message was generated from.

## Use the *ra_statistics* command to troubleshoot

The ra_statistics command returns activity-related statistics that you can use to evaluate Mirror Replication Agent operations and performance. By comparing the statistics returned when you first run the command, to the statistics returned after you have successfully replicated something you know will work, you can analyze the differences in the statistics and troubleshoot where the problem lies. The statistics help you determine if the instance is:

- Scanning the transaction log
- Processing replicated transactions
- Sending LTL to the Replication Server

❖ **To check Mirror Replication Agent operations**

1 Log in to the Mirror Replication Agent instance with the administrator login.

2 Verify that you are in *Replicating* state. If you are not, change the state to *Admin*. For information, see "Check the Mirror Replication Agent status" on page 142.

3 Use the following command to get current Mirror Replication Agent activity statistics:

```
ra_statistics
```

This command returns statistics for all of the Mirror Replication Agent components and the Java VM.

4 Save the statistics returned to use as a baseline for comparison.

5 Perform activity against the object that is not being replicated.

For example, update a table that is not being replicated.

6 Repeat step 2.

**Note** Be sure to allow enough time for the Mirror Replication Agent to process the transaction.

7 Compare the new returned statistics activity with the baseline.

Check for differences and see the table below.

| If... | Then... |
|---|---|
| The value returned for `Total Maintenance User operations filtered` increases | You are executing the transaction as the Replication Server maintenance user for this Replication Server connection. By default, these transactions are not sent to Replication Server. You must either connect to the primary database as a different user, or you can set the configuration value of filter_maint_userid to false.<br>See "Setting up the connection configuration parameters" on page 72. |

For more information about the ra_statistics command, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

## Running out of memory

If you are running out of memory, you will get the following error message:

```
java.lang.OutOfMemoryError
```

When you are running out of memory, the Mirror Replication Agent drops out of *Replicating* state, or the entire Mirror Replication Agent server may stop executing.

To support adjusting the amount of memory available to the JRE, all of the executable scripts (or batch files) in the Mirror Replication Agent *bin* directory refer to an environment variable named RA_JAVA_MAX_MEM.

All Mirror Replication Agent instance RUN scripts also reference the RA_JAVA_MAX_MEM environment variable.

You can adjust the amount of memory available to the JRE by either:

- Setting the value of a system variable named RA_JAVA_MAX_MEM in your environment and using the ra utility to start the Mirror Replication Agent instance, or
- Setting the value of the RA_JAVA_MAX_MEM variable in the Mirror Replication Agent instance RUN script and using the RUN script to start the Mirror Replication Agent instance.

If you start a Mirror Replication Agent instance by invoking the ra utility, you can set the value of the RA_JAVA_MAX_MEM system variable in your environment to specify the amount of memory available to the JRE. Before it sets the RA_JAVA_MAX_MEM variable to a default value, the ra and ra_admin utilities check to see if it is already set.

If you start a Mirror Replication Agent instance by invoking the instance *RUN* script (or batch file), you can edit the instance RUN script to change the default value of RA_JAVA_MAX_MEM and specify the amount of memory available to the JRE.

**Note** When a Mirror Replication Agent instance is started with the instance *RUN* script, the value of the RA_JAVA_MAX_MEM variable specified in the *RUN* script overrides the value set elsewhere. Therefore, you can edit the *RUN* script to adjust the memory available to the JRE for each instance.

### Debugging LTL

LTL (Log Transfer Language) is the syntax used to communicate or distribute replication data to Replication Server. It is the principle output from a Mirror Replication Agent. For more details about LTL syntax, see the Replication Server *Design Guide*.

❖ **To debug LTL**

1. Log in to the running Mirror Replication Agent instance using the administrator login.
2. Use the ra_status command to verify that the Mirror Replication Agent instance is in *Admin* state:

   ```
   ra_status
   ```
3. Use the ra_config command to set the values of the following Mirror Replication Agent configuration parameters for the primary database:

   ```
   ra_config LITTRACELTL, true
   ```
4. Use the resume command to change the Mirror Replication Agent state to *Replicating*:

   ```
   resume
   ```
5. When new replication activity is generated, check the *LTITRACELTL.log* file in the log directory to debug your problem.

Once new activity is generated, check the *LTITRACELTL.log* file in the log directory to debug your problem.

Uncompressing LTL for debugging a problem

By default, the LTL generated by the Mirror Replication Agent is compressed to reduce the amount of data sent to Replication Server, reducing network bandwidth. In cases where you require more verbose output to help debug a problem, change the following configuration parameters to produce more verbose LTL.

❖ **To produce more verbose LTL**

1. Log in to the running Mirror Replication Agent instance using the administrator login.
2. Use the ra_status command to verify that the Mirror Replication Agent instance is in *Admin* state:

   ```
   ra_status
   ```
3. Use the ra_config command to set the values of the following Mirror Replication Agent configuration parameters for the primary database:

   ra_config column_compression = false

   ra_config compress_ltl_syntax = false

   ra_config structured_tokens = false

4 When new replication activity is generated, check the *LTITRACELTL.log* file in the log directory to debug your problem.

## Checking the Replication Server

This section describes how to use Replication Server commands to check for the most common replication problems. For more detailed information about diagnosing and solving Replication Server problems see the Replication Server *Troubleshooting Guide*.

### Replication Server status and operation

Replication Server provides several admin commands that you can use to check on its status and operation.

❖ **To check the status and operation of the Replication Server**

1 Log in to the Replication Server with a user login that has “sa” permission.

2 Use the following command to check the current status of the Replication Server:

```
admin health
```

This command returns the current status of the Replication Server, as shown in the following example:

```
Mode         Quiesce     Status
-------       -------     ------
NORMAL        FALSE      HEALTHY
```

If the Replication Server status is `SUSPECT`, use the admin who_is_down command to check for Replication Server threads that may be down or attempting to connect to other servers.

3 Use the following command to check the current status of the Replication Server database connections, including the connection from the Mirror Replication Agent to the Replication Server:

```
admin show_connections
```

You can also use the admin who, dsi command to get more information about the Mirror Replication Agent connection in the Replication Server.

---

**Note** You can use the admin show_connections or admin who, dsi command output to verify that the primary data server and primary database names are correct for the Mirror Replication Agent connection in the primary Replication Server.

---

For more information about the admin show_connections and admin who commands, see the Mirror Activator for Oracle Mirror Replication Agent *Reference Manual*.

## Mirror Replication Agent login in Replication Server

Executing the Replication Server connect source lti command accomplishes three things:

- Verifies that the Replication Server database connection used by the Mirror Replication Agent exists in the Replication Server
- Verifies that the login name specified in the Mirror Replication Agent rs_username parameter has permission to connect to the Replication Server as a data source
- Returns a version string that shows the highest numbered version of LTL that the Replication Server supports

### ❖ To verify that the *rs_username* login has appropriate permissions

1. Log in to the Replication Server with the Mirror Replication Agent user login name specified in the rs_username configuration parameter.

   Refer to the “Installation and Setup Worksheet” in the Mirror Activator for Oracle *Installation Guide* for this login name.

2. Execute the connect source lti command using the following example syntax:

   ```
   connect source lti pds.pdb version
   ```

   where:

   - *pds* is the value specified for the Mirror Replication Agent rs_source_ds configuration parameter.

- *pdb* is the value specified for the Mirror Replication Agent rs_source_db configuration parameter.
- *version* is the proposed LTL version number.

Refer to the "Installation and Setup Worksheet" in the Mirror Activator for Oracle *Installation Guide* for the values of the rs_source_ds and rs_source_db parameters.

Enter 999 for the value of the LTL version number. Replication Server returns the highest numbered version of LTL that it supports.

3 Disconnect from the Replication Server as rs_username, and then log in to the Mirror Replication Agent instance with the administrator login and invoke the resume command.

For more information about the connect source lti command, see the Replication Server *Design Guide* and *Reference Manual*.

## Replication Server stable queues

Check the Replication Server stable queues to determine which transactions are being processed or ignored, and to determine whether transactions are open (not committed).

### ❖ To display information about SQM and SQT threads

1. Log in to the Replication Server and execute the admin who, sqm command.
2. View the results to determine the number of duplicate messages being detected and ignored, and the number of blocks being written in the Replication Server stable queues.
3. Execute the Replication Server admin who, sqt command.
4. View the results to find open transactions.

For more information about the admin who command, see the Mirror Replication Agent *Reference Manual*.

APPENDIX A

# Materializing Databases

This appendix describes how to materialize the databases in a Mirror Activator for Oracle system.

| Topic | Page |
|---|---|
| Selecting a materialization option | 153 |

The procedures in this appendix assumes that:

- You have already installed the Mirror Replication Agent software and Replication Server software, as described in the Mirror Activator for Oracle *Installation Guide* and the Replication Server installation and configuration guides for your platform.
- You have an existing, operational Oracle database, and you have configured that database as the primary database in a Mirror Activator for Oracle system, as described in Chapter 2, "Setup and Configuration."
- You have an existing, operational Oracle database, and you have configured that database as the standby database in a Mirror Activator for Oracle system, as described in Chapter 2, "Setup and Configuration."

## Selecting a materialization option

The term *materialization* refers to the process of copying the contents of one database (the source) to another (the target), so that both databases contain identical data. This is the prerequisite condition (or starting point) for any system that provides continuous data replication, including the Mirror Activator for Oracle system.

Materialization always replaces existing data (if any) in the target database with the data in the source database. Therefore, any process that copies only the differences between the source data and the target data (for example, an "incremental replication") is not materialization.

Some materialization techniques copy all of the source data to the target, but for only part of the source database (for example, subscription materialization in Replication Server). Such techniques are not well suited for databases in the Mirror Activator for Oracle system.

The Mirror Activator for Oracle system is intended to replicate a complete database, with a one-to-one relationship between each object in one primary database and each object in one standby database. Because the Mirror Activator for Oracle system relies on mirror log devices, which are exact copies of the primary database log devices, a device-level database materialization technique may be well suited for a Mirror Activator for Oracle system.

Snapshot materialization allows you to take advantage of your disk replication system's facilities to simplify the materialization process, and to reduce the time required for a complete database materialization.

Choosing the materialization technique to use will depend on several factors. Using your disk replication solution will typically provide the fastest transfer mechanism, but it may require skills that the typical DBA is not familiar with. Similarly, hardware system administrators may be comfortable with disk replication configuration and file system organization but less familiar with timing and commands required for database execution and reloading. Existing copy mechanisms used by your organization may provide slower raw transfer rates but faster overall execution time, simply due to having existing familiarity with the technique. Therefore, the choice of materialization technique is left to individual discretion.

The Mirror Activator for Oracle system setup procedures in Chapter 2, "Setup and Configuration," and the failover and failback procedures in Appendix B, "Failover and Failback with Mirror Activator for Oracle," are based on using the snapshot materialization technique.

APPENDIX B

# Failover and Failback with Mirror Activator for Oracle

This appendix describes tasks that you must incorporate in failover and failback procedures for the Mirror Activator for Oracle system.

## Limitations and assumptions

This appendix does not consider all of the many possible configurations, options, and scenarios that can affect failover and failback procedures. It also does not consider the impact and interaction of any other system that might share resources with, or have some interdependency with the Mirror Activator for Oracle system.

This appendix describes only failover and failback tasks that are specific to the Mirror Activator for Oracle system. General failover and failback tasks, such as re-routing network connections and switching client access from one database to another, are *not* covered in this document.

The procedures in this appendix assume that:

- All Mirror Activator for Oracle system components are set up and properly configured, as described in Chapter 2, “Setup and Configuration.”Also, when functioning normally, the Mirror Activator for Oracle system is capable of replicating transactions from the primary database to the standby database, with no replication failures.
- The disk replication system is set up and configured to mirror all data devices, and to mirror all log devices to both the local site and the mirror log device site.

- The Mirror Activator for Oracle system is essentially self-contained:
  - It is capable of operating independently of other systems and databases outside of its control.
  - It shares no system resources (including servers, networks, devices, and disk replication system facilities) with other systems that reside at the primary and standby sites.
  - All Mirror Activator for Oracle system components—the primary and standby databases and all of their devices, disk replication system and mirror log devices, Mirror Replication Agent, and Replication Server—are dedicated solely to the Mirror Activator for Oracle system.

Additional assumptions may apply to specific procedures in this appendix.

## Failover procedure

Failover refers to the process of switching normal system operations from a primary database to a standby database, particularly in the event of a failure that interrupts:

- Operations at the primary database, or
- Access to the primary database.

You can also use a failover procedure to mitigate the impact of scheduled downtime on database users and clients (for example, when maintenance operations require the primary database to be offline).

This section describes only the failover tasks that are specific to the Mirror Activator for Oracle system. General system failover procedures are not covered in this document.

Table B-1 provides a checklist of the tasks that you must include in failover procedures for the Mirror Activator for Oracle system.

The checklist in Table B-1 assumes that:

- The primary database has gone offline, triggering the failover procedure to begin.
- To provide recovery from a standby database failure, you will use the disk replication system to capture and mirror all changes on the standby database devices while the primary database is offline.

Sybase recommends that you perform these tasks in the order shown.

**Table B-1: Mirror Activator for Oracle system failover tasks**

| Task | Description |
|---|---|
| 1 | Verify that the Mirror Replication Agent has finished processing the last transaction record in the log, and then stop replication in the Mirror Replication Agent. |
| 2 | Verify that the Replication Server has distributed the last committed transaction in the log. |
| 3 | Check the Replication Server stable queue for open transactions.<br><br>**Note** Before failback, you must purge any open transactions that remain in the Replication Server stable queue. |
| 4 | Fail over the disk replication system to:<br>• Treat the standby database data and log devices as "primary" devices.<br>• Mirror all subsequent changes on the standby database data and log devices to another set of standby (or backup) devices. |
| 5 | Switch access for users and client applications from the primary database to the standby database. |

Use the following procedure for Mirror Activator for Oracle system failover tasks.

❖ **To fail over the Mirror Activator for Oracle system**

1 Verify that the Mirror Replication Agent has finished processing all transactions in the log, and then stop replication in the Mirror Replication Agent.

   a Log in to the Mirror Replication Agent administration port and execute the following command:

   ```
   ra_status
   ```

   When the Mirror Replication Agent has finished processing all of the log records, the ra_status command shows its state is *Replicating (Waiting at end of log)*.

   b After it finishes processing the last log record, execute the following command to stop replication in the Mirror Replication Agent:

   ```
   quiesce
   ```

   After you invoke quiesce, use the ra_status command to verify that the Mirror Replication Agent is in *Admin* state.

For more information, see "Stopping replication in the Mirror Replication Agent" on page 92 and "Determining current Mirror Replication Agent status" on page 85.

2 Check the Replication Server stable queue for open transactions.

Log in to the Replication Server with a user login that has "sa" permission, and execute the following command:

```
admin who, sqt
```

Check the admin who, sqt output for the SQT thread associated with the standby database connection. The following column values indicate that all committed transactions have been distributed successfully, and that open (uncommitted) transactions remain in the stable queue:

- Closed – 0
- Open – (any number other than zero)
- SQM Blocked – 1
- First Trans – STO

**Note** Before failback, you must purge any open transactions that remain in the Replication Server stable queue.

For more information about the admin who command, see the Replication Server *Reference Manual*.

3 Fail over the disk replication system.

Reconfigure the disk replication system to:

- Treat the standby database data and log devices as "primary" (source) devices
- Mirror all subsequent changes on the standby database data and log devices to another set of standby (backup) devices, preferably at an alternate site (neither the primary nor standby site)

Refer to the documentation provided by your disk replication system vendor for more information about the procedures in this step.

4 After you complete all of the previous tasks, you can switch access for users and client applications from the primary database to the standby database.

## Failback procedure

Failback refers to the process of restoring normal user and client access to a primary database, after a failover switched access from the primary database to a standby database.

This section describes only the failback tasks that are specific to the Mirror Activator for Oracle system. General system failback procedures are not covered in this document.

Table B-2 provides a checklist of the tasks that you must include in failback procedures for the Mirror Activator for Oracle system.

The checklist in Table B-2 assumes that:

- You have purged the Replication Server stable queue to remove any open transactions remaining after failover.
- The primary data server is up and functioning correctly, and you are ready to return it to normal operation.

**Note** If you do not purge the Replication Server stable queue to remove any open transaction that remained in the queue after failover:

- Replication will not start, because Replication Server will not send any new transactions to the standby database until it receives operations to complete the open transactions.
- Eventually, the stable queue will run out of space, because Replication Server will not send any transactions after you resume replication upon failback.

Sybase recommends that you perform these tasks in the order shown.

***Table B-2: Mirror Activator for Oracle system failback tasks***

| Task | Description |
|---|---|
| 1 | Quiesce the standby database to suspend update activity. |
| 2 | Fail back the disk replication system to: <br>• Re-materialize the primary database data and log devices from the standby database devices. <br>• Materialize the mirror log devices at the standby site from the materialized primary log devices. <br>• Re-establish synchronous replication from the primary log devices to the mirror log devices at the standby site. |
| 3 | Bring the primary database online. |
| 4 | Initialize the primary database using the pdb_init move_truncpt command to set the truncation point to the end of the log. |
| 5 | Quiesce the primary database. |
| 6 | Initialize the Mirror Replication Agent using the ra_init force command, and set the paths to the mirror log devices. |
| 7 | Release the quiesce on the primary database, *after* the Mirror Replication Agent initialization is complete. |
| 8 | Switch access for users and client applications from the standby database to the primary database. |
| 9 | Release the quiesce on the standby database, *after* client access is switched to the primary database. |
| 10 | Start replication in the Mirror Replication Agent with the resume command. |

Use the following procedure for Mirror Activator for Oracle system failback tasks.

❖ **To fail back the Mirror Activator for Oracle system**

1 Quiesce the standby database to suspend update activity.

This can be accomplished in several ways, including running an oracle command, such as "alter system quiesce restricted", or performing this at a time when no users are on the system (Sunday night at 2:00 AM). The exact method is left up to the Oracle DBA.

2 Fail back the disk replication system to:

- Materialize the primary database data and log devices from the standby database devices.
- Materialize the mirror log devices at the standby site from the materialized primary log devices.

- Re-establish synchronous replication from the primary log devices to the mirror log devices at the standby site.

Refer to the documentation provided by your disk replication system vendor for more information about the procedures in this step.

For more information about re-materializing a primary database, see Appendix A, “Materializing Databases.”

3. Bring the primary database online.
4. Initialize the primary database using the Mirror Replication Agent pdb_init move_truncpt command. This command sets the truncation point to the end of the log.

   For more information about initializing the primary database, see “Setting up the Mirror Activator for Oracle system” on page 15.
5. Quiesce the primary database.

   This can be accomplished in several ways, including running an oracle command, such as "alter system quiesce restricted", or performing this at a time when no users are on the system (Sunday night at 2:00 AM). The exact method is left up to the Oracle DBA.
6. Initialize the Mirror Replication Agent using the ra_init force command to update the system data repository in the RASD, and then use ra_devicepath to set the paths to the mirror log devices (if necessary).

   For more information, see “Updating the RASD” on page 97.
7. Release the quiesce on the primary database, *after* the Mirror Replication Agent initialization is complete.
8. Switch access for users and client applications from the standby database to the primary database.
9. Release the quiesce on the standby database, after client access is switched to the primary database.
10. Start replication in the Mirror Replication Agent.

    Log in to the Mirror Replication Agent administration port and execute the following command:

    ```
    resume
    ```

    After you invoke resume, use the ra_status command to verify that the Mirror Replication Agent is in *Replicating* state.

For more information, see “Starting replication in the Mirror Replication Agent” on page 91.

APPENDIX C

# Mirror Replication Agent and Oracle Databases

This appendix describes the characteristics of the Mirror Replication Agent that are unique to its use in Mirror Activator for Oracle.

## Oracle-specific issues

This section describes general issues and considerations that are specific to using Mirror Replication Agent version 12.6 with the Oracle data server.

The following topics are included in this section:

- Mirror Replication Agent connectivity
- Mirror Replication Agent permissions
- Redo log setup
- Setting ddl_username and ddl_password
- Character case of database object names
- Format of origin queue ID
- Datatype compatibility
- Oracle datatype restrictions
- Oracle large object (LOB) support

- Oracle user-defined types

## Mirror Replication Agent connectivity

Connectivity between the Mirror Replication Agent and the Oracle data server is through the Oracle JDBC thin driver.

The Oracle JDBC driver must be installed on the Mirror Replication Agent host machine, and the directory this driver is installed in must be in the CLASSPATH environment variable.

The TNS Listener Service must be installed and running so the Mirror Replication Agent instance can connect to it. See the Oracle Networking document for more information.

## Mirror Replication Agent permissions

Mirror Replication Agent uses the pds_username to connect to Oracle and must have the following Oracle permissions:

- create session – required to connect to Oracle.
- select_catalog_role – required to select from the DBA_* views.
- alter system – required to perform redo log archive operations.
- execute on DBMS_FLASHBACK – required to execute DBMS_FLASHBACK.get_system_change_number.
- alter any procedure – required to instrument procedures for replication.
- create table – required to create tables in the primary database.
- create procedure – required to create rs_marker and rs_dump proc procedures.
- create public synonym – required to create synonyms for created tables in the primary database.
- create sequence – required to create a sequence.
- drop public synonym – required to drop created synonyms.
- drop sequence – required to drop a sequence.
- select on SYS.OBJ$ – required to process procedure DDL commands.

- select on SYS.LOB$ – required to support LOB replication.
- select on SYS.COLLECTION$ – required to support table replication.
- select on SYS.COL$ – required to support table replication.
- select on SYS.CON$ – required to support table replication.
- select on SYS.COLTYPE$ – required to support table replication.
- select on SYS.CDEF$ – required to support table replication.
- select on SYS.USER$ – required to support all replication.
- select on SYS.SEQ$ – required to support sequence replication.

In addition, the user who starts the Mirror Replication Agent instance must have read access to the Oracle *redo* logs.

## Redo log setup

**Note** Mirror Replication Agent requires that automatic archiving of Oracle redo logs be disabled. Archiving is performed manually by the Mirror Replication Agent as the data in the redo logs is replicated. The following tasks should not be performed until the Mirror Replication Agent is ready to be initialized, so that archiving of redo logs will continue to take place.

Mirror Replication Agent requires the following settings in your Oracle database:

- Redo log archiving must be enabled. To enable redo log archiving, execute the following command:

  ```
  alter database ARCHIVELOG;
  ```

  To verify that log archiving is enabled, execute the following query:

  ```
  select log_mode from v$database;
  ```

  If `ARCHIVELOG` is returned, then log archiving is enabled.

- Automatic archiving must be disabled in the active server and when you re-start the Oracle server. To stop automatic archiving in the active server, enter the following Oracle command:

  ```
  alter system archive log stop
  ```

To disable automatic archiving when you re-start the Oracle server, change the value of the server's LOG_ARCHIVE_START parameter to FALSE.

You can change the LOG_ARCHIVE_START parameter by manually editing the server's startup parameter file, or by using the following Oracle command:

```
alter system set log_archive_start=false
scope=spfile
```

To check the setting of the LOG_ARCHIVE_START parameter, execute the following query:

```
select value from v$system_parameter where name =
'log_archive_start';
```

If FALSE is returned, then the value in the server parameter file has been correctly modified to prevent automatic archiving when you re-start the Oracle server.

For more information about the LOG_ARCHIVE_START parameter, or the ALTER SYSTEM commands, see the Oracle database reference guide.

After Mirror Replication Agent is initialized, automatic archiving must never be enabled, even temporarily. If automatic archiving is re-enabled, or manual archiving is performed, causing a redo log file not yet processed by the Mirror Replication Agent to be overwritten, then the data in the lost redo log file will not be replicated. To recover from this situation requires that the replicate be re materialized

- In Oracle release 9.2 and later, supplemental logging of primary key data and index columns must be enabled. To enable supplemental logging, execute the following Oracle command:

  ```
  alter database add SUPPLEMENTAL LOG DATA (PRIMARY
  KEY, UNIQUE INDEX) COLUMNS;
  ```

  To verify that supplemental logging of primary key information is enabled, execute the following query:

  ```
  select SUPPLEMENTAL_LOG_DATA_PK from v$database;
  ```

  If `YES` is returned, then supplemental logging of primary key information is enabled.

You can enable the forced logging of all database changes to the Oracle redo log file. Sybase recommends setting this option to insure that all data that should be replicated is logged. To enable the force logging command, execute the following statement on the primary database:

```
alter database FORCE LOGGING;
```

To verify the current setting of the force logging command, execute the following statement on the primary database:

```
select force_logging from v$database;
```

---

**Note** Mirror Replication Agent must be executed on a machine that has physical access to the redo logs.

---

## Setting *ddl_username* and *ddl_password*

To replicate DDL in Oracle, in addition to setting the value of pdb_setrepddl to enable, you must set the Mirror Replication Agent ddl_username and ddl_password parameters. The ddl_username parameter is the database user name included in LTL for replicating DDL commands to the standby database. This user must have permission to execute all replicated DDL commands at the standby database. The ddl_password parameter is the database user name's password.

For details on setting these parameters, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

When you replicate DDL in Oracle, you must use Oracle as the standby database. You cannot replicate DDL commands from Oracle to non-Oracle standby databases.

---

**Note** To replicate DDL, Replication Server must have a database-level replication definition with replicate DDL set in the definition. For details, see the Mirror Replication Agent *Reference Manual*.

---

### DDL commands and objects filtered from replication

The following DDL commands are not replicated:

alter database
create database link
drop database link
alter session
create snapshot
create snapshot log

alter snapshot
alter snapshot log
drop snapshot
drop snapshot/log
alter rollback segment
create rollback segment
drop rollback segment
alter system switch log
create control file
create pfile from spfile
create schema authorization
create spfile from pfile
explain
lock table
rename
set constraints
set role
set transaction
analyze
audit
no audit
create tablespace
alter tablespace
drop tablespace

The following objects are not replicated:

- Any objects that are owned by SYS.
- Any object owned by users defined in the list of non-replicated users. You can modify this list using the pdb_ownerfilter command. In addition, Sybase has provided a default list of owners whose objects will not be replicated. However, you cannot remove the SYS owner. You can use the pdb_ownerfilter command to return, add, or remove the list of owners whose objects will not be replicated. For more information, see the Mirror Activator for Oracle *Mirror Replication Agent Reference Manual*.

## Character case of database object names

Database object names must be delivered to the primary Replication Server in the same format as they are specified in replication definitions. For example, if a replication definition specifies a table name in all lowercase, then that table name must appear in all lowercase when it is sent to the primary Replication Server by the Mirror Replication Agent.

---

**Note** Replication will fail if a database object name is delivered to the primary Replication Server in a format different from that specified in the replication definition.

---

Mirror Replication Agent version 12.6 gives you some control over the way it treats the character case of database object names when it sends LTL to the primary Replication Server. You have three options:

- asis – database object names are passed to Replication Server in the same format as they are actually stored in the primary data server.

  ---

  **Note** asis is the default and recommended setting.

  ---

- lower – database object names are passed to Replication Server in *all lowercase*, regardless of the way they are actually stored in the primary data server.
- upper – database object names are passed to Replication Server in *all uppercase*, regardless of the way they are actually stored in the primary data server.

You specify the character case option you want by setting the value of the ltl_character_case configuration parameter. The parameter values are asis (the default), lower, and upper.

In the case of the Oracle data server, database object names are stored in all uppercase by default. However, if you create a case-sensitive name, the case sensitivity is retained in Oracle.

See the following examples using the asis option:

- `create table tabA` is stored as `TABA`
- `create table Tabb` is stored as `TABB`
- `create table 'TaBc'` is stored as `TaBc`

See the following examples using the upper option:

- `create table tabA` is stored as `TABA`
- `create table Tabb` is stored as `TABB`
- `create table 'TaBc'` is stored as `TABC`

## Format of origin queue ID

Each record in the transaction log is identified by an origin queue ID that consists of 64 hexadecimal characters (32 bytes). The format of the origin queue ID is determined by the Mirror Replication Agent instance, and it varies according to the primary database type.

Table C-1 illustrates the format of the origin queue ID for the Mirror Replication Agent.

***Table C-1: Mirror Replication Agent origin queue ID***

| Character | Bytes | Description |
|---|---|---|
| 0-3 | 2 | Database generation ID |
| 4-19 | 8 | System change number |
| 20-27 | 4 | Log sequence number |
| 28-35 | 4 | Block number |
| 36-39 | 2 | Block offset, relative to the start of the block |
| 40-47 | 4 | Oldest active transaction begin log sequence number |
| 48-55 | 4 | Oldest active transaction begin block number |
| 56-59 | 2 | Oldest active transaction begin block offset |
| 60-63 | 2 | Available for specifying uniqueness |

## Datatype compatibility

Mirror Replication Agent processes Oracle transactions and passes data to the primary Replication Server. In turn, the primary Replication Server uses the datatype formats specified in the replication definition to receive the data from Mirror Replication Agent.

Table C-2 describes the conversion of Oracle datatypes to Sybase datatypes.

***Table C-2: Oracle to Sybase datatype mapping***

| Oracle datatype | Oracle length/range | Sybase datatype | Sybase length/range | Notes |
|---|---|---|---|---|
| CHAR | 255 bytes | char | 32K | |

| Oracle datatype | Oracle length/range | Sybase datatype | Sybase length/range | Notes |
|---|---|---|---|---|
| DATE | 8 bytes, fixed-length, default format: DD-MON-YY | rs_oracle_ datetime | 8 bytes | Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD. The default value replicated is: MM/DD/YYYY HH24:MI:SS. If pdb_convert_datetime is true, the value replicated is YYYYMMDD HH:MM:SS.sss. |
| TIMESTAMP(n) | 21-31 bytes, variable-length, default format: DD-MON-YY hh.mm.ss.ffffff AM | rs_oracle_ datetime9 | 8 bytes | Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD. Default format is DD-MON-YY HH.MI.SS.FF AM. |
| TIMESTAMP(n) WITH [LOCAL] TIME ZONE | Variable-length, default format: DD-MON-YY hh.mm.ss.ffffff AM {+\|-}hh:mm | rs_oracle_ timestamptz | | |
| INTERVAL YEAR(n) TO MONTH | Variable-length | rs_oracle_ interval | | |
| INTERVAL DAY(n) TO SECOND(n) | Variable-length | rs_oracle_ interval | | |
| LONG | 2GB, variable-length character data | text | | |
| LONG RAW | 2GB, variable-length binary data | image | | |
| BLOB | 4GB, variable-length binary large object | image | | |
| CLOB | 4GB, variable-length character large object | text | | |

| Oracle datatype | Oracle length/range | Sybase datatype | Sybase length/range | Notes |
|---|---|---|---|---|
| NCHAR | 255 bytes, multibyte characters | unichar or char | 32K | |
| NCLOB | 4GB, variable-length multibyte character large object | text | | |
| NVARCHAR2 | 2000 bytes, variable-length, multibyte character data | univarchar or varchar | 32K | |
| BFILE | 4GB, locator points to large binary file | image | | Not supported. |
| MLSLABEL | 5 bytes, variable-length binary OS label | | | Not supported. |
| NUMBER (p,s) | 21 bytes, variable-length numeric data | rs_oracle_ decimal | float is 4 or 8 bytes. int is 4 bytes. real is 4 bytes. number and decimal are 2 to 17 bytes. | The float datatype can convert to scientific notation if the range is exceeded. Integers (int) are truncated if they exceed the Replication Server range of 2,147,483,647 to -2,147,483,648, or 1x10$^{-130}$ to 9.99x10$^{25}$. The number and decimal datatypes are truncated if they exceed the range of -10$^{38}$ to 10$^{38}$-1. Oracle precision ranges from 1 to 38 digits. Default precision is 18 digits. Oracle scale ranges from -84 to 127. Default scale is 0. |
| RAW | 2000 bytes, variable-length binary data | rs_oracle_ raw | 32K | |
| ROWID | 6 bytes, binary data representing row addresses | rs_oracle_ rowid | 32K | |
| UDD object type | variable length character data | rs_rs_char_ raw | 32K | See “Oracle user-defined types” on page 178. |

| Oracle datatype | Oracle length/range | Sybase datatype | Sybase length/range | Notes |
|---|---|---|---|---|
| VARCHAR2 | 4000 bytes, variable-length character data | varchar | 32K | |

## Oracle datatype restrictions

**Note** See the Mirror Activator for Oracle *Release Bulletin* for the latest information on datatype restrictions.

Replication Server and Mirror Replication Agent impose the following constraints on the Oracle NUMBER datatype:

- In the *integer* representation:
  - The corresponding Sybase int datatype has a smaller absolute maximum value.

    The Oracle NUMBER absolute maximum value is 38 digits of precision, between 9.9 x $10^{125}$ and 1 x $10^{-130}$. The Sybase int value is between $2^{31}$ - 1 and -$2^{31}$ (2,147,483,647 and -2,147,483,648), inclusive.
  - Oracle NUMBER values greater than the Sybase int maximum are rejected by Replication Server.

  **Note** To avoid Replication Server restrictions using the int datatype, use the rs_oracle_decimal datatype instead when creating replication definitions for tables with large Number values.

- In the *floating point* representation:
  - The precision of the floating point representation has the same range limitation as the integer representation.
  - If the floating point value is outside the Sybase range of $2^{31}$ - 1 and -$2^{31}$ (2,147,483,647 and -2,147,483,648), Mirror Replication Agent converts the number into exponential format to make it acceptable to Replication Server. No loss of precision or scale occurs.

Replication Server and Mirror Replication Agent impose the following constraints on the Oracle TIMESTAMP WITH [LOCAL] TIME ZONE datatype:

- When a timestamp with time zone datatype is replicated, the time zone information is used to resolve the timestamp value to the 'local' time zone and then the resolved value is replicated. The time zone information itself is not replicated.
- As an example, if a timestamp with time zone is recorded in Oracle as "01-JAN-05 09:00:00.000000 AM -8:00" and the 'local' timezone is -6:00, the value replicated will be "01-JAN-05 11:00:00.000000." The timestamp value is adjusted for the difference between the recorded timezone of -8:00 and the local time zone of -6:00, and the adjusted value is replicated.

---

**Note** Oracle time zone files are platform-dependent.

---

If you use a version of Replication Server prior to version 12.5, the following size restrictions are imposed on Oracle datatypes:

- Oracle BLOB, CLOB, NCLOB, and BFILE datatypes that contain more than 2GB will be truncated to 2GB.
- Oracle CHAR, RAW, ROWID, and VARCHAR2 datatypes that contain more than 255 bytes will be truncated to 255 bytes.
- Oracle NCHAR and NVARCHAR2 multibyte character datatypes are replicated as char or varchar single-byte datatypes.

---

**Note** With Replication Server version 12.5 or later, these datatype size restrictions are no longer in effect.

---

The following Oracle datatypes are not supported:

- ORDDoc
- BFILE
- UnType
- DBUnType
- HttpUriType
- FTPUriType
- ORDAudio
- ORDImage
- ORDVideo

- Opaque
- XML types
- Oracle UDD Ref Type
- Oracle UDD Varray Type
- Oracle UDD Nested Table Type
- Supplied types

See Also

Mirror Replication Agent *Reference Manual* for more information on replication definitions and the create replication definition command.

## Oracle large object (LOB) support

Oracle LOB data can exist in several formats in Oracle. The LOB datatypes in Oracle are:

- character
  - LONG
  - CLOB
  - NCLOB
- binary
  - LONG RAW
  - BLOB
  - BFILE

  BFILE points to file contents stored outside of the Oracle database.

For those types stored in the database (all types except BFILE), Oracle records the content of the LOB in the redo files. The Mirror Replication Agent reads the LOB data from the redo file and submits the data for replication.

**Note** Replication of BFILE datatypes is not supported.

## Special handling for *off row* LOBS

LOB types that are stored within the Oracle database (BLOB, CLOB and NCLOB) can be defined with certain storage characteristics. One of those characteristics, "disable storage in row," indicates that the data for the LOB should *always* be recorded separate from the rest of the data in the row the LOB belongs to. This *off row* storage requires special handling for replication of updates to these LOB values.

When an off row LOB value is updated, the change recorded in the redo log is for the index that holds the LOB's data. The row the LOB belongs to is not changed. As a result, information is missing from the redo log to identify what row in the table the LOB belongs to.

As an example, when a non-LOB column is updated in a table, all of the column data that identifies the changed values and look-up columns is recorded. The following command records values in the redo log for the values of both "col2" and "col1":

```
updated myTable set col2 = 2 where col1 = 1
```

In contrast, a command that updates only a LOB that has been defined with the disable storage in row clause records only the LOB data's change to its index, and not the table that holds the LOB. So the following command only records the value changed, and does not include the value of "col1":

```
updated myTable set ClobColumn = 'more data' where col1 = 1.
```

Because the value of the columns in the where clause are not logged in that update, there is insufficient information to build the correct where clause to be used to apply the data at the standby site. To resolve this problem, Mirror Replication Agent requires that an update to a LOB column defined with disable storage in row *must* be immediately accompanied by an insert or update to the same row in the table the LOB belongs to.

The Mirror Replication Agent uses the additional column data from the associated operation to correctly build the where clause required to support replication.

For example, the following transaction sequences support replication of updates to LOB column "ClobColumn" when it has been defined with the disable storage in row clause:

```
begin
insert into myTable (col1, col2, ClobColumn, updated)
values (1,1,empty_clob(), sysdate);
update myTable set ClobColumn = 'more data' where col1
= 1;
```

```
commit

begin
update myTable set updated = sysdate() where col1 = 1;
update myTable set ClobColumn = 'more data' where col1
= 1
commit

begin
update myTable set ClobColumn = 'more data' where col1
= 1
update myTable set updated = sysdate() where col1 = 1;
commit
```

The following transaction sequences are *not* supported for LOB columns defined with the disable storage in row and result in a failure to supply the LOB data to the standby site:

- Missing accompanying change to the same row:

  ```
  begin
  update myTable set ClobColumn = 'more data' where
  col1 = 1
  commit
  ```

- Accompanying change for the same row is not immediately adjacent to the LOB change:

  ```
  begin
  update myTable set updated = sysdate where col1 = 1;
  update myTable set col2 = 5 where col1 = 5;
  update myTable set ClobColumn = 'more data' where
  col1 = 1
  commit
  ```

This limitation only applies to LOB columns that have been defined with the disable storage in row clause.

You can identify the LOB columns in your database that have this constraint using the following query against your Oracle database:

```
select owner, table_name, column_name from dba_lobs where in_row = 'NO';
```

## Oracle user-defined types

User-defined datatypes (UDD) use Oracle built-in datatypes and other user-defined datatypes as building blocks that model the structure and behavior of data in applications.

Mirror Replication Agent version 12.6 supports replication of user-defined object types. Object types are abstractions of real-world entities, such as purchase orders, that application programs deal with. An object type is a schema object with three kinds of components:

- A name, which identifies the object type uniquely within that schema.
- Attributes, which are built-in types or other user-defined types. Attributes model the structure of the real-world entity.
- Methods, which are functions or procedures written in PL/SQL and stored in the database, or written in a language such as C or Java and stored externally. Methods implement operations the application can perform on the real-world entity.

### Replicating UDDs

To replicate UDDs in Oracle, you must add a datatype definition to Replication Server so the UDD is replicated exactly as it is executed in the primary database. UDDs from Oracle are sent to Replication Server as data for a single varchar column. By default, Replication Server wraps all varchar data in single quotation marks. In order to prevent Replication Server from adding these quotation marks to UDD data, a special datatype must be created in Replication Server *and* that dataype must be used as the datatype for any UDD column defined in a replication definition.

When you create a datatype definition in Replication Server, you must use an unused datatype ID. This is the DTID column of the rs_datatype table. The new datatype is a Replication Server datatype, so it will be available to all connections defined in the Replication Server that owns the Replication Server System Database (RSSD); you only have to do this once for each Replication Server instance.

❖ **To create a datatype definition in Replication Server**

Creating the datatype requires Replication Server administrator privileges or granted permission.

1. Log into the RSSD.
2. Add a row to the rs_datatype table using the following example as a guide:

```
/*  rs_oracle_udd_raw - char with no delimiters */
insert into rs_datatype values(
0,                     /* prsid */
0x0000000001000008,  /* classid */
'rs_oracle_udd',     /* name */
0x0000000000010210,  /* dtid */
0,                     /* base_coltype */
255,                   /* length */
0,                     /* status */
1,                     /* length_err_act */
'CHAR',                /* mask */
0,                     /* scale */
0,                     /* default_len */
'',                    /* default_val */
0,                     /*-delim_pre_len-*/
'',                    /* delim_pre */
0,                     /*-delim_post_len-*/
'',                    /* delim_post */
0,                     /* min_boundary_len */
'',                    /* min_boundary */
3,                     /* min_boundary_err_act */
0,                     /* max_boundary_len */
'',                    /* max_boundary_err_act */
0                      /* rowtype */
)
go
```

3 You *must* restart Replication Server after adding a new type.

4 In Replication Server, test the new type using the admin translate command:

```
admin translate, 'The quick brown fox jumped over the lazy dog.',
'char(255)', 'rs_oracle_udd'

go
Delimiter Prefix   Translated                  Value Delimiter Postfix

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

NULL                 The quick brown fox jumped over the lazy dog.  NULL
```

The new type has been defined correctly if the sentence was translated correctly.

Example

The following example demonstrates how to create a replication definition, using a new type defined in Replication Server. The following Oracle table and type definitions are used in the example:

- Oracle UDD object type name: NAME_T

- Oracle table name: USE_NAME_T
- Oracle table columns: PKEY INT, PNAME NAME_T

```
create replication definition use_name_t_repdef
with primary at ra_source_db.ra_source_ds
with all tables named 'USE_NAME_T'
(
   PKEY int,
   PNAME rs_oracle_udd
)
primary key (PKEY)
searchable columns (PKEY)
go
```

**Note** The ltl_character_case must be in uppercase for this example.

## Understanding Sequence Replication

The following sections describe sequence logging, replicating, performance and replication alternatives

### Logging of sequence information

Individual sequence changes are not logged in the Oracle database log file. But changes to Oracle Sequence impact (update) the Oracle sys.seq$ table. These changes do not occur with each new sequence value generated. Instead, the sys.seq$ table is updated periodically, based on sequence caching refresh activity or other system changes. The value stored in the sys.seq$ table for a sequence is the *next* value to be assigned *after* the existing cache of values has been exhausted.

As an example, a newly created sequence starts with a value of 1, increments by 1 and has a cache value of 20. These are all default values and can all be customized. The value stored in the sys.seq$ record for this new sequence is 21. This indicates that the "next" value to be used by the sequence, after the existing cache of 20 numbers is used, is 21. The record in sys.seq$ does not change until the sequence value hits 21. At that time, Oracle will cache the next 20 values for the sequence, and the sys.seq$ record will be updated to 41. It is this value (41) recorded in change to the sequences sys.seq$ record, that will be used for replication. The key point is recognizing that not every individual sequence update is recorded in the log, and is therefore not available for replication.

## Replicating sequence changes

When a sequence is marked for replication, changes to that sequence against sys.seq$ are captured and sent to Replication Server in the form of parameters passed to a procedure. The procedure (rs_update_sequence) must be installed at the standby site as part of system setup, as well as a function replication definition for that procedure. At the standby site, an implementation of rs_update_sequence will increment a same-named sequence until its value is equal to that at the primary.

Scripts are provided to create the rs_update_sequence stored procedure and function replication definition. The scripts can be found in the Mirror Replication Agent installation as:

```
$SYBASE/MRO-12_6/scripts
/oracle_create_replicate_sequence_proc.sql
```

```
$SYBASE/MRO-12_6/scripts
/oracle_create_rs_sequence_repdef.sql
```

## Performance considerations

Compared to the performance of incrementing a sequence at the Primary database, particularly where sequence values are cached, the effort to increment the same sequence at the standby site may be less efficient. The stored procedure must dynamically determine the sequence to increment, and must loop internally, incrementing the sequence until the primary value has been reached. The loop is required because there is no way to "assign" a specific value to a sequence.

Since the name of the sequence is passed as a parameter, Oracle cannot pre-compile the procedure for efficiency. With the addition of the looping activity required to properly increment the sequence, the performance of the solution may impact some environments where a large number of highly used sequences is the norm.

## Sequence replication alternatives

If the performance of sequence replication is a concern, there are other alternatives to replication that support primary and standby use of the same sequence. These alternatives are currently suggested by Oracle and others interested in providing sequence coordination between multiple sites:

1. Assuming the sequence is being used to generate primary key values, the sequence at each site can be concatenated with something unique to the site. For example, use a sequence number concatenated with the database name, site name or something similar. This technique allows each site to maintain a unique *range* of sequence of numbers. Each site having a unique range would mean there would be no value in sending (replicating) changes of one site's range to another site.

2. Similar to concatenating, each site could obtain a different range of numbers by having different starting points, or increment values, for the same sequence.

   As an example, the sequence at one site could start at one and increment by two to generate odd numbers (1, 3, 5) while the other site starts at two an generates even numbers (2, 4, 6). Again, each site would have a unique range and could avoid any need for replication.

3. A third option is available to standby solutions, where the standby site is for read-only, and does not access the sequence value until fail-over. Rather than continually replicate a sequence's value, the value of the sequence at the standby site could be updated as part of the failover tasks. After failover and before the standby allows connection to client applications, a script or procedure could query the last used sequence value (based on the last table to use it for a primary key) and update or redefine the sequence one, based on that calculated value.

## Mirror Replication Agent objects in the Oracle primary database

Mirror Replication Agent creates objects in the Oracle primary database to assist with replication tasks.

The Mirror Replication Agent objects are created by invoking the pdb_init command. When you invoke this command, Mirror Replication Agent generates a SQL script that contains the SQL statements for the objects created or modified in the primary database. This script is stored in the *partinit.sql* file in the *MRO-12_6\inst_name\scripts\xlog\installed* directory. The objects must be created before any primary database objects can be marked for replication.

---

**Note** The generated scripts are for informational purposes only and *cannot* be run manually to initialize the primary database. This is also true for the procedure marking and unmarking scripts that are generated when you use pdb_setrepproc. Scripts are no longer generated when you mark and unmark tables with pdb_setreptable.

---

### Mirror Replication Agent object names

There are two variables in the transaction log component database object names shown in this chapter:

- prefix – represents the one- to three-character string value of the pdb_xlog_prefix parameter (the default is ra_).
- xxx – represents an alphanumeric counter, a string of characters that is (or may be) added to a database object name to make that name unique in the database.

The value of the pdb_xlog_prefix parameter is the prefix string used in all Mirror Replication Agent object names.

The value of the pdb_xlog_prefix_chars parameter is a list of the non-alphanumeric characters allowed in the prefix string specified by pdb_xlog_prefix. This list of allowed characters is database-specific. For example, in Oracle, the only non-alphanumeric characters allowed in a database object name are the $, #, and _ characters.

You can use the ra_helpsysinfo command to view the names of Mirror Replication Agent transaction log components in the primary database.

❖ **To find the names of the objects created**

- At the Mirror Replication Agent administration port, invoke the ra_helpsysinfo command with no keywords:

  ```
  ra_helpsysinfo
  ```

  The ra_helpsysinfo command returns a list of the transaction log base objects.

## Table objects

Table C-3 lists the tables that are considered Mirror Replication Agent objects. No permissions are granted on these tables when they are created.

***Table C-3: Mirror Replication Agent transaction log base tables***

| Table | Database name |
|---|---|
| Transaction log system table | *prefix*XLOG_SYSTEM_ |
| Procedure-active table | *prefix*PROCACTIVE_[*xxx*] |

## Procedure objects

Table C-4 lists the procedure objects that are considered Mirror Replication Agent objects. No permissions are granted when these procedures are created.

***Table C-4: Mirror Replication Agent marker procedures and shadow tables***

| Procedure/Table | Database name |
|---|---|
| Transaction log marker procedure | RS_MARKER[*xxx*] |
| Dump marker procedure | RS_DUMP[*xxx*] |
| Transaction log marker shadow table | *prefix*SH_RS_MARKER_[*xxx*] |
| Dump marker shadow table | *prefix*SH_RS_DUMP_[*xxx*] |

## Sequences

Table C-5 lists the Oracle sequences that are considered Mirror Replication Agent base objects. No permissions are granted when these procedures are created.

***Table C-5: Mirror Replication Agent sequences***

| Sequence | Database name |
|---|---|
| Assign procedure call | *prefix*PCALL_[*xxx*] |

## Marked procedures

Table C-6 lists the Mirror Replication Agent objects that are created for each primary procedure that is marked for replication. These objects are created only when a procedure is marked for replication.

***Table C-6: Mirror Replication Agent objects for each marked procedure***

| Object | Database name |
|---|---|
| Shadow table | *prefix*SH_*xxx* |

# Archiving the Oracle redo log

To ensure that data in the Oracle redo logs is not archived and overwritten before it is processed for replication, Mirror Replication Agent requires that automatic archiving by Oracle is disabled. The Oracle archiving process is handled by the following Mirror Replication Agent log truncation processing.

Mirror Replication Agent provides features for both automatic and manual log truncation.

Mirror Replication Agent provides two options for automatic transaction log truncation:

- Periodic truncation, based on a time interval you specify
- Automatic truncation whenever Mirror Replication Agent receives a new LTM Locator value from the primary Replication Server

You also have the option to switch off automatic log truncation. By default, automatic log truncation is enabled and is set to truncate the log whenever Mirror Replication Agent receives a new LTM locator value from the primary Replication Server.

## Transaction log truncation

Each time Mirror Replication Agent performs log truncation (either scheduled or manual) it issues the alter system command with the archive log sequence keywords. The command uses the log sequence number of the redo log file whose contents have been processed by the Mirror Replication Agent and are ready to be archived.

**Note** The alter system command syntax in Oracle allows redo log files to be archived in addition to the single log sequence specified in the command. To avoid the possibility of unintentional archiving, Mirror Replication Agent only issues this command when it is processing the redo log file whose status is `CURRENT`.

### Automatic transaction log truncation

You can specify the automatic truncation option you want (including none) by using the ra_config command to set the value of the truncation_type configuration parameter.

If you want to truncate the transaction log automatically based on a time interval, use the ra_config command to set the value of the truncation_interval configuration parameter.

### Manual transaction log truncation

At any time, you can truncate the Mirror Replication Agent transaction log manually by invoking the pdb_truncate_xlog command at the Mirror Replication Agent administration port.

Mirror Replication Agent creates objects in the Oracle primary database to assist with replication tasks.

## Mirror Replication Agent setup test scripts

Mirror Replication Agent provides a set of test scripts that automate the process of creating a replication test environment that you can use to verify the installation and configuration of the Mirror Replication Agent software and the basic function of the other components in your replication system.

The Mirror Replication Agent test scripts are located in the *scripts* subdirectory under the Mirror Replication Agent base directory. For example: *MRO-12_6\scripts*.

The Mirror Replication Agent test scripts perform the following tasks:

1. Create a primary table.
2. Create a replicate table that corresponds to the primary table.
3. Apply Heterogeneous datatype support objects to Replication Server.
4. Create the primary data server connection in the primary Replication Server.
5. Create the replication definition in the primary Replication Server.
6. Create the subscription in the replicate Replication Server.
7. Create the Mirror Replication Agent transaction log in the primary database.
8. Modify the primary table.
9. Clean up and remove the replication test environment created by the other scripts.

## Before you begin

Before running the test scripts, make sure that you have completed the following tasks:

- Installed Oracle data servers instances to act as primary and standby databases.
- Installed and running ECDA for Oracle, Replication Server, and Mirror Replication Agent.
- Created a Mirror Replication Agent instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, asis
  - ra_config rs_source_ds, MRO
  - ra_config rs_source_db, TEST

  **Note** These recommended configuration parameter values are for this test only. Values for a production environment (or even for a different test environment) may be different.

- Configured all the pds, rs, and rssd connection parameters and tested them successfully with the Mirror Replication Agent test_connection command

- Confirmed that all servers are running
- Confirmed that the Mirror Replication Agent instance is in the *Admin* state

## Create the primary table

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_create_test_primary_table.sql* script to create the primary table in the primary database.

## Create the replicate table

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle replicate data server and execute the same script *(oracle_create_test_primary_table.sql*) to create a matching table in the replicate database.

## Apply Heterogeneous Datatype Support objects to Replication Server

Replication Server requires changes to the RSSD to support Oracle datatypes. To set up Replication Server changes in the RSSD, execute the following Replication Server scripts against the RSSD database. You must edit these scripts to supply the correct RSSD database name.

```
$SYBASE/REP-12_6/scripts/hds_oracle_udds.sql
$SYBASE/REP-12_6/scripts/hds_oracle_funcstrings.sql
$SYBASE/REP-12_6/scripts/hds_clt_ase_to_oracle.sql
$SYBASE/REP-12_6/scripts/hds_clt_oracle_to_ase.sql
```

The scripts in the Replication Server directory have not been updated for this release. Apply the following additional script from the Mirror Replication Agent installation to your RSSD:

```
$SYBASE/MRO-12_6/scripts/hds_oracle_new_udds.sql.sq
```

The *oracle_error_class* script does not exist by default in Replication Server. Create it using the following scripts from the Mirror Replication Agent installation:

```
Apply script $SYBASE/MRO-12_6
/scripts/oracle_create_error_class_1_rs.sql to
Replication Server
```

```
Apply script $SYBASE/MRO-12_6
/scripts/oracle_create_error_class_2_rssd.sql to the
RSSD
```

```
Apply script $SYBASE/MRO-12_6
/scripts/oracle_create_error_class_3_rs.sql to
Replication Server
```

## Create Replication Server objects in the replicate database

Use the Replication Server script to create required objects in the replicate Oracle database. Use the ECDA connection through isql to apply the script, logged in as the replicate database maintenance user:

```
$SYBASE/REP-12_6/scripts
/hds_oracle_setup_for_replicate.sql
```

The scripts in the Replication Server directory have not been updated for this release. Instead, apply the following scripts from the Mirror Replication Agent installation:

```
$SYBASE/MRO-12_6/scripts
/hds_oracle_new_setup_for_replicate.sql
```

Use your Oracle database access tool (such as sqlplus) to create the procedure in the replicate Oracle database required for sequence replication:

```
$SYBASE/MRO-12_6/scripts
/oracle_create_replicate_sequence_proc.sql
```

## Create the Replication Server connection for Mirror Replication Agent

Edit the *oracle_create_rs_primary_connection.sql* script, changing "{pds}.{pdb}" to "MRO.TEST," and changing "sys" and "sys_pwd" to a valid user and password for your primary Oracle database.

Use isql or another query processor to log in to the primary Replication Server, and execute the edited script to create the Replication Server connection to the primary database.

## Create the replicate data server connection

Edit the *oracle_create_rs_standby_connection.sql* script, changing "{rds}.{rdb}" to "ecdaconn.ecdaconn," where "edcdaconn" is the name of the connection string your defined to ECDA to connect to the replicate Oracle database. Also, change "rs_maint_user" and "rs_maint_user_pwd" to the maintenance user and password for your replicate Oracle database.

Use isql or another query processor to log in to the primary Replication Server, and execute the edited script to create the Replication Server connection to the replicate database.

## Create the replication definition

Edit the *oracle_create_rs_db_repdef.sql script,* changing "{pds}.{pdb}" to "MRO.TEST." Edit the *oracle_create_rs_sequence_repdef.sql* script, making the same changes. Use isql or another query processor to log in to the primary Replication Server, and apply both edited scripts to create the Replication Server database replication definition and function replication definition.

## Create the subscription

Edit the *oracle_create_rs_db_sub.sql* script, changing "{pds}.{pdb}" to "MRO.TEST'"and "{rds}.{rdb}'"to "ecdaconn.ecdaconn'"where "edcdaconn" is the name of the connection string your defined to ECDA to connect to the replicate Oracle database.

Use isql or another query processor to log in to the replicate Replication Server, and apply the edited script to create the Replication Server database subscription.

## Initialize the Mirror Replication Agent

If you have not already created the Mirror Replication Agent transaction log in the primary database, create the transaction log now.

❖ **To create the Mirror Replication Agent transaction log**

1 Use isql or another query processor to log in to the Mirror Replication Agent administration port.

2 Use the following command to initialize the transaction log:

```
pdb_init move_truncpt
ra_init
```

For more information about creating the Mirror Replication Agent transaction log, see "Initializing the Mirror Replication Agent transaction log" on page 29.

## Mark a primary table for replication

Mark the test primary table that was created in the Oracle database by the *oracle_create_test_primary_table.sql* script (rax_test).

❖ **To mark the test primary table for replication**

1 Use isql or another query processor to log in to the Mirror Replication Agent administration port.

2 Use the following command to mark the rax_test table for replication:

```
pdb_setreptable rax_test, mark
```

For more information about marking objects in the primary database, see "Marking and unmarking tables" on page 107.

**Note** Make sure that replication is enabled for the rax_test table.

## Start replication

If you have not already put the Mirror Replication Agent instance in the *Replicating* state, put the Mirror Replication Agent instance in the *Replicating* state now.

❖ **To start test replication**

1 Use isql or another query processor to log in to the Mirror Replication Agent administration port.

2 Use the following command to put the Mirror Replication Agent instance in the *Replicating* state:

```
resume
```

For more information about starting replication, see "Starting the Mirror Replication Agent" on page 58.

## Execute the test transaction script in Oracle

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server, and execute the *oracle_primary_test_transactions.sql* script to generate transactions in the primary table in the primary database.

## Check results of replication

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server, and execute the *oracle_select_test_primary.sql* script to view the changes in the test primary table in the primary database.

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server, and execute the *oracle_select_test_primary.sq*l script to view the changes in the test replicate table in the replicate database.

## Clean up the test environment

Use the following procedure to clean up and remove the replication test environment that you created in the previous sections.

❖ **To clean up and remove the replication test environment**

1. Log in to the Mirror Replication Agent administration port using isql (or another query processor).
2. Use the following command to *suspend* the Mirror Replication Agent instance:

   ```
   suspend
   ```
3. Use the following command to remove the Mirror Replication Agent transaction log and unmark the test primary table (rax_test) in the Oracle database:

   ```
   ra_deinit, force
   ```

4 Edit the *oracle_drop_rs_db_sub.sql* and *oracle_drop_rs_standby_connection.sql* scripts, changing "{pds}.{pdb}" to "MRO.TEST" and '"{rds}.{rdb}" to "ecdaconn.ecdaconn," where "edcdaconn" is the name of the connection string you defined to ECDA to connect to the replicate Oracle database.

   Use isql or another query processor to log in to the replicate Replication Server, and execute the edited *oracle_drop_rs_db_sub.sql* script followed by the *oracle_drop_rs_standby_connection.sql* script to remove the Replication Server connection to the replicate database.

5 Log in to the replicate data server, and execute the script to drop the test replicate table:

```
oracle_drop_test_primary_table.sql
```

6 Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server, and execute the *oracle_drop_test_primary_table.sql* script to drop the test primary table.

# Glossary

This glossary describes Mirror Activator for Oracle and Mirror Replication Agent terms used in this book.

**Adaptive Server**

The brand name for Sybase relational database management system (RDBMS) software products.

- *Adaptive Server Enterprise* manages multiple, large relational databases for high-volume online transaction processing (OLTP) systems and client applications.
- *Adaptive Server IQ* manages multiple, large relational databases with special indexing algorithms to support high-speed, high-volume business intelligence, decision support, and reporting client applications.
- *Adaptive Server Anywhere* manages relational databases with a small RDBMS footprint, which is ideal for embedded applications and mobile device applications.

See also **database** and **RDBMS**.

**atomic materialization**

A materialization method that copies subscription data from a primary database to a standby database in a single, atomic operation. No changes to primary data are allowed until the subscription data is captured at the primary database. See also **bulk materialization** and **nonatomic materialization**.

**BCP utility**

A bulk copy transfer utility that provides the ability to load multiple rows of data into a table in a target database. See also **bulk copy**.

**bulk copy**

An Open Client interface for the high-speed transfer of data between a database table and program variables. It provides an alternative to using SQL insert and select commands to transfer data. See also **BCP utility** and **materialization**.

| | |
|---|---|
| **bulk materialization** | A materialization method whereby subscription data in a standby database is initialized outside of the replication system. You can use bulk materialization for subscriptions to table replication definitions or function replication definitions. See also **atomic materialization**, **materialization**, and **nonatomic materialization**. |
| **client** | In client/server systems, the part of the system that sends requests to servers and processes the results of those requests. See also **client application**. |
| **client application** | Software that is responsible for the user interface, including menus, data entry screens, and report formats. See also **client**. |
| **commit** | An instruction to the DBMS to make permanent the changes requested in a transaction. Contrast with **rollback**. See also **DBMS** and **transaction**. |
| **data client** | A client application that provides access to data by connecting to a data server. See also **client**, **client application**, and **data server**. |
| **data distribution** | A method of locating (or placing) discrete parts of a single set of data in multiple systems or at multiple sites. Data distribution is distinct from data replication, although a data replication system can be used to implement or support data distribution. Contrast with **data replication**. |
| **data replication** | The process of copying data to remote locations, and then keeping the replicated data synchronized with the primary data. Data replication is distinct from data distribution. Replicated data is stored copies of data at one or more remote sites throughout a system, and it is not necessarily distributed data. Contrast with **data distribution**. See also **disk replication** and **transaction replication**. |
| **data server** | A server that provides the functionality necessary to maintain the physical representation of a table in a database. Data servers are usually database servers, but they can be any data repository with the interface and functionality a data client requires. See also **client**, **client application**, and **data client**. |
| **database** | A collection of data with a specific structure (or schema) for accepting, storing, and providing data for users. See also **data server** and **relational database**. |
| **database connection** | A connection that allows Replication Server to manage the database and distribute transactions to the database. Each database in a replication system can have only one database connection defined in Replication Server. See also **Replication Server** and **route**. |
| **datatype** | A keyword that identifies the characteristics of stored information on a computer. Some common datatypes are: char, int, smallint, date, time, numeric, and float. Different data servers support different datatypes. |

| | |
|---|---|
| **DBMS** | An abbreviation for *database management system*. A DBMS is a computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The DBMS can include the user interface for using the database, or it can be a stand-alone data server system. Compare with **RDBMS**. See also **database**. |
| **disaster recovery** | A method or process used to restore the critical business functions interrupted by a catastrophic event. A disaster recovery (or business continuity) plan defines the resources and procedures required for an organization to recover from a disaster, based on specified recovery objectives. |
| **disk replication** | A data replication method that copies blocks or pages from a primary disk device to a standby device. Sometimes referred to as *disk mirroring*. See also **data replication** and **transaction replication**. |
| **failback** | A procedure that restores the normal user and client access to a primary database, after a failover procedure switched access from the primary database to a standby database. Contrast with **failover**. |
| **failover** | A procedure that switches user and client access from a primary database to a standby database, particularly in the event of a failure that interrupts operations at the primary database, or access to the primary database. Failover is an important fault-tolerance feature for systems that require high availability. Contrast with **failback**. |
| **ERSSD** | An abbreviation for embedded *Replication Server System Database*. The ERSSD manages replication system information for a Replication Server. |
| **function** | A Replication Server object that represents a data server operation, such as insert, delete, or begin transaction. Replication Server distributes operations to standby databases as functions. See also **function string**. |
| **function string** | A string that Replication Server uses to map a function and its parameters to a data server API. Function strings allow Replication Server to support replication between (homogeneous) non-Sybase data servers, and heterogeneous replication, in which the primary and replicate databases are different types, with different SQL extensions and different command features. See also **function**. |
| **gateway** | Connectivity software that allows two or more computer systems with different network architectures to communicate. |
| **inbound queue** | A stable queue managed by Replication Server to spool messages received from a Mirror Replication Agent. See also **outbound queue** and **stable queue**. |

| | |
|---|---|
| **interfaces file** | A file containing information that Sybase Open Client and Open Server applications need to establish connections to other Open Client and Open Server applications. See also **Open Client** and **Open Server**. |
| **isql** | An interactive SQL client application that can connect and communicate with any Sybase Open Server application, including Adaptive Server, Mirror Replication Agent, and Replication Server. See also **Open Client** and **Open Server**. |
| **Java** | An object-oriented, platform-independent, "write once, run anywhere" programming language developed by Sun Microsystems. The Mirror Replication Agent is a Java application. |
| **Java VM** | The Java Virtual Machine. The Java VM (or JVM) is the part of the Java Runtime Environment (JRE) that interprets Java byte codes. See also **Java** and **JRE**. |
| **JDBC** | An abbreviation for *Java Database Connectivity*. JDBC is the standard communication protocol for connectivity between Java clients and data servers. See also **client**, **data server**, and **Java**. |
| **jConnect** | The Sybase JDBC driver that Mirror Replication Agent uses to connect to Replication Server and the RSSD. |
| **JRE** | An abbreviation for *Java Runtime Environment*. The JRE consists of the Java Virtual Machine (Java VM or JVM), the Java Core Classes, and supporting files. To run a Java application, such as the Mirror Replication Agent, a JRE must be installed on the machine. See also **Java** and **Java VM**. |
| **LAN** | An abbreviation for "local area network." A local area network is a computer network located on the user's premises and covering a limited geographical area (usually a single site). Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary can be subject to some form of regulation. Contrast with **WAN**. |
| **latency** | In transaction replication, the time it takes to replicate a transaction from a primary database to a standby database. Specifically, latency is the time elapsed between committing an original transaction in the primary database and committing the replicated transaction in the standby database.<br><br>In disk replication, latency is the time elapsed between a disk write operation that changes a block or page on a primary device and the disk write operation that changes the replicated block or page on a mirror (or standby) device.<br><br>See also **disk replication** and **transaction replication**. |

| | |
|---|---|
| **LOB** | An abbreviation for *large object*. A LOB is a type of data element (or datatype) associated with a column that contains extremely large quantities of data. |
| **Log Reader** | An internal component of the Mirror Replication Agent that interacts with the primary database and mirror log devices to capture transactions for replication. See also **Log Transfer Interface**. |
| **Log Transfer Interface** | An internal component of the Mirror Replication Agent that interacts with Replication Server to forward transactions for distribution to a standby database. See also **Log Reader**. |
| **Log Transfer Language** | The proprietary protocol used between Mirror Replication Agent and Replication Server to replicate data from the primary database to Replication Server. See also **Log Reader** and **Log Transfer Interface**. |
| **Maintenance User** | A special user login name in the standby database that Replication Server uses to apply replicated transactions to the database. See also **Replication Server** and **standby database**. |
| **materialization** | The process of copying the data from a primary database to a standby database, initializing the standby database so that the Mirror Activator for Oracle system can begin replicating transactions. See also **atomic materialization**, **bulk materialization**, and **non-atomic materialization**. |
| **Mirror Replication Agent** | An application that reads a primary database transaction log to acquire information about data-changing transactions in the primary database, processes the log information, and then sends it to a Replication Server for distribution to a standby database. See also **primary database**, **Replication Server**, and **standby database**. |
| **nonatomic materialization** | A materialization method that copies subscription data without a lock on the primary database. Changes to primary data are allowed during data transfer, which may cause temporary inconsistencies between the primary and standby databases. Contrast with **atomic materialization**. See also **bulk materialization**. |
| **ODBC** | An abbreviation for *Open Database Connectivity*. ODBC is an industry standard communication protocol for clients connecting to data servers. See also **client**, **data server**, and **JDBC**. |
| **Open Client** | A Sybase product that provides customer applications, third-party products, and other Sybase products with the interfaces needed to communicate with Open Server applications. See also **Open Server**. |
| **Open Client application** | An application that uses Sybase Open Client libraries to implement Open Client communication protocols. See also **Open Client** and **Open Server**. |

| | |
|---|---|
| **Open Server** | A Sybase product that provides the tools and interfaces required to create a custom server. See also **Open Client**. |
| **Open Server application** | A server application that uses Sybase Open Server libraries to implement Open Server communication protocols. See also **Open Client** and **Open Server**. |
| **outbound queue** | A stable queue managed by Replication Server to spool messages to a standby database. See also **inbound queue**, **stable queue**, and **standby database**. |
| **primary data** | The version of a set of data that is the source used for replication. Primary data is stored and managed by the primary database. See also **primary database**. |
| **primary database** | The database that contains the data to be replicated to another database (the standby database) through a replication system. The primary database is the source of replicated transactions and data in a replication system. Sometimes called the *active database*. Contrast with **standby database**. See also **primary data** and **replicated transaction**. |
| **primary key** | The column or columns whose data uniquely identify each row in a table. |
| **primary site** | The location or facility at which primary data servers and primary databases are deployed to support normal business operations. Sometimes called the *active site* or *main site*. See also **primary database** and **standby site**. |
| **primary table** | A table used as a source for replication. Primary tables are defined in the primary database schema. See also **primary data** and **primary database**. |
| **primary transaction** | A transaction that is committed in the primary database and recorded in the primary database transaction log. See also **primary database** and **transaction log**. |
| **quiesce** | To cause a system to go into a state in which further data changes are not allowed. See also **quiescent**. |
| **quiescent** | In a replication system, a state in which all data-changing operations have been propagated to their destinations. Some Mirror Replication Agent and Replication Server commands require that you quiesce the replication system.<br><br>In a database, a state in which all data-changing operations are suspended so that transactions cannot change any data, and the data and log devices are stable.<br><br>This term is interchangeable with *quiesced* and *in quiesce*. See also **quiesce**. |

| | |
|---|---|
| **RASD** | An abbreviation for *Replication Agent System Database*. Information in the RASD is used by the primary database to recognize database structure or schema objects in the transaction log. |
| **RCL** | An abbreviation for *Replication Command Language*. RCL is the command language used to manage Replication Server. See also **Replication Server**. |
| **RDBMS** | An abbreviation for *relational database management system*. An RDBMS is an application that manages and controls relational databases. Compare with **DBMS**. See also **relational database**. |
| **relational database** | A collection of data in which data is viewed as being stored in tables, which consist of columns (data items) and rows (units of information). Relational databases can be accessed by SQL requests. Compare with **database**. See also **SQL**. |
| **replicated data** | A set of data that is replicated from a primary database to a standby database by a replication system. See also **primary database**, **replication system**, and **standby database**. |
| **replicated transaction** | A primary transaction that is replicated from a primary database to a standby database by a transaction replication system. See also **primary database**, **primary transaction**, **standby database**, and **transaction replication**. |
| **replication definition** | A description of a table or stored procedure in a primary database, for which subscriptions can be created. The replication definition, maintained by Replication Server, includes information about the columns to be replicated and the location of the primary table or stored procedure. See also **Replication Server** and **subscription**. |
| **Replication Server** | The Sybase software product that provides the infrastructure for a robust transaction replication system. See also **Mirror Replication Agent**. |
| **RSSD** | An abbreviation for *Replication Server System Database*. The RSSD manages replication system information for a Replication Server. See also **Replication Server**. |
| **replication system** | A data processing system that replicates data from one location to another. Data can be replicated between separate systems at a single site, or from one or more local systems to one or more remote systems. See also **data replication**, **disk replication**, and **transaction replication**. |
| **rollback** | An instruction to a database to reverse the data changes requested in a unit of work (a transaction). Contrast with **commit**. See also **transaction**. |

| | |
|---|---|
| **SQL** | An abbreviation for *Structured Query Language*. SQL is a non-procedural programming language used to process data in a relational database. ANSI SQL is an industry standard. See also **transaction**. |
| **stable queue** | A disk device-based, store-and-forward queue managed by Replication Server. Messages written into the stable queue remain there until they can be delivered to the appropriate process or standby database. Replication Server provides a stable queue for both incoming messages (the inbound queue) and outgoing messages (the outbound queue). See also **database connection** and **Replication Server**. |
| **standby data** | The data managed by a standby database, which is the destination (or target) of a replication system. Contrast with **primary data**. See also **standby database** and **replication system**. |
| **standby database** | A database that contains data replicated from another database (the primary database) through a replication system. The standby database is the database that receives replicated transactions and/or data in a replication system. Sometimes called the *replicate database*. Contrast with **primary database**. See also **replicated transaction**, **replication system**, and **standby data**. |
| **standby site** | The location or facility at which standby data servers and standby databases are deployed to support disaster recovery, and normal business operations during scheduled downtime at the primary site. Sometimes called the *alternate site* or *replicate site*. Contrast with **primary site**. See also **standby database**. |
| **subscription** | A request for Replication Server to maintain a replicated copy of a table, or a set of rows from a table, in a standby database at a specified location. See also **replication definition**, **Replication Server**, and **standby database**. |
| **table** | In a relational database, a two-dimensional array of data, or a named data object that contains a specific number of unordered rows composed of a group of columns that are specific to the table. See also **database** and **relational database**. |
| **transaction** | A unit of work in a database that can include zero, one, or many operations (including insert, update, and delete operations), and that is either applied or rejected as a whole. Each SQL statement that modifies data can be treated as a separate transaction, if the database is so configured. See also **replicated transaction** and **SQL**. |

| | |
|---|---|
| **transaction log** | Generally, the log of transactions that affect the data managed by a data server. Mirror Replication Agent reads the transaction log to identify and acquire the transactions to be replicated from the primary database. See also **Mirror Replication Agent**, **primary database**, and **transaction**. |
| **transaction replication** | A data replication method that copies data-changing operations from a primary database to a standby database. See also **data replication**, **primary database**, and **standby database**. |
| **transactional consistency** | A condition in which all transactions in the primary database are applied in the standby database, and in the same order that they were applied in the primary database. See also **primary database**, **standby database**, and **transaction**. |
| **WAN** | An abbreviation for "wide area network." A wide area network is a system of local-area networks (LANs) connected together with data communication lines. Contrast with **LAN**. |

# Index