SYBASE®

Administration Guide

Mirror Activator

12.6

DOCUMENT ID: DC20096-01-1260-02

LAST REVISED: June 2004

Copyright © 2004 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, the Sybase logo, AccelaTrade, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Server IQ, Adaptive Warehouse, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, AvantGo, AvantGo Application Alerts, AvantGo Mobile Delivery, AvantGo Mobile Document Viewer, AvantGo Mobile Inspection, AvantGo Mobile Marketing Channel, AvantGo Mobile Pharma, AvantGo Mobile Sales, AvantGo Pylon, AvantGo Pylon Application Server, AvantGo Pylon Conduit, AvantGo Pylon PIM Server, AvantGo Pylon Pro, Backup Server, BizTracker, ClearConnect, Client-Library, Client Services, Convoy/DM, Copernicus, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DB-Library, dbQueue, Developers Workbench, Direct Connect Anywhere, DirectConnect, Distribution Director, e-ADK, E-Anywhere, e-Biz Impact, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, EWA, Financial Fusion, Financial Fusion Server, Gateway Manager, GlobalFIX, iAnywhere, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InternetBuilder, iScript, Jaguar CTS, jConnect for JDBC, Mail Anywhere Studio, MainframeConnect, Maintenance Express, Manage Anywhere Studio, M-Business Channel, M-Business Network, M-Business Server, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, My AvantGo, My AvantGo Media Channel, My AvantGo Mobile Marketing, MySupport, Net-Gateway, Net-Library, New Era of Networks, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Orchestration Studio, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, PocketBuilder, Pocket PowerBuilder, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, PowerJ, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Rapport, RepConnector, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report-Execute, Report Workbench, Resource Manager, RW-DisplayLib, RW-Library, S-Designor, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILS, smart.partners, smart.parts, smart.script, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Financial Server, Sybase Gateways, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server and XP Server are trademarks of Sybase, Inc. 02/04

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

This product includes code licensed from RSA Security, Inc. Some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Mirror Activator 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 solution includes the following Sybase software products:

- Mirror Replication Agent™
- Replication Server®

Audience

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

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

Chapter 4, “Mirror Replication Agent Command Reference,” describes Mirror Replication Agent commands, including syntax, options, usage, and examples.

Chapter 5, “Mirror Replication Agent Configuration Parameters,” describes the Mirror Replication Agent configuration file and provides a configuration parameter reference.

Chapter 6, “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 system.

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

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

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

- Mirror Activator *Overview Guide* for more information about how Mirror Activator fits into an integrated disaster recovery solution
- Mirror Activator *Administration Guide* for an overview of the Mirror Activator solution, information about configuring and administering the Mirror Replication Agent component, and information about configuring other software components in the Mirror Activator system.
- Mirror Replication Agent *Installation Guide* for information about installing the Mirror Replication Agent software
- The Mirror Replication Agent release bulletin for last-minute information that was too late to be included in the books

Note A more recent version of the Mirror Replication Agent 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.

Make sure you have appropriate documentation for the Adaptive Server Enterprise version used with the Mirror Activator system.

Java environment

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

- The Mirror Replication Agent 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 Sybase Technical Library CD, and the Technical Library 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 Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).
- The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.
- The Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Updates, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Technical Library 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.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.

5 Click a Certification Report title to display the report.

❖ **Creating 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

❖ **Finding 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. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).

3 Select a product.

4 Specify a time frame and click Go.

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/MRA-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.

The HTML documentation has 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.

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.

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 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:

- At best, standby system resources are under-utilized, with frequent periods when they must be idle or offline.

  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.

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

- 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 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. 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.
- Reduced network bandwidth requirements because:
  - Transactions rolled back in the primary database need not be replicated.
  - Transaction replication can encode the replicated transactions and data in a more compact form than the actual log records or raw SQL for 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.

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.
  - There could be some data loss 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 solution

Mirror Activator 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.

Mirror Activator software

The Mirror Activator solution includes the following Sybase software products:

- Mirror Replication Agent
- Replication Server

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 on a separate platform 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 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.

Mirror Activator advantages

Compared to disaster recovery solutions that are based on either disk replication or transaction replication alone, advantages of the Mirror Activator 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 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 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 provides an integrated disaster recovery solution with:

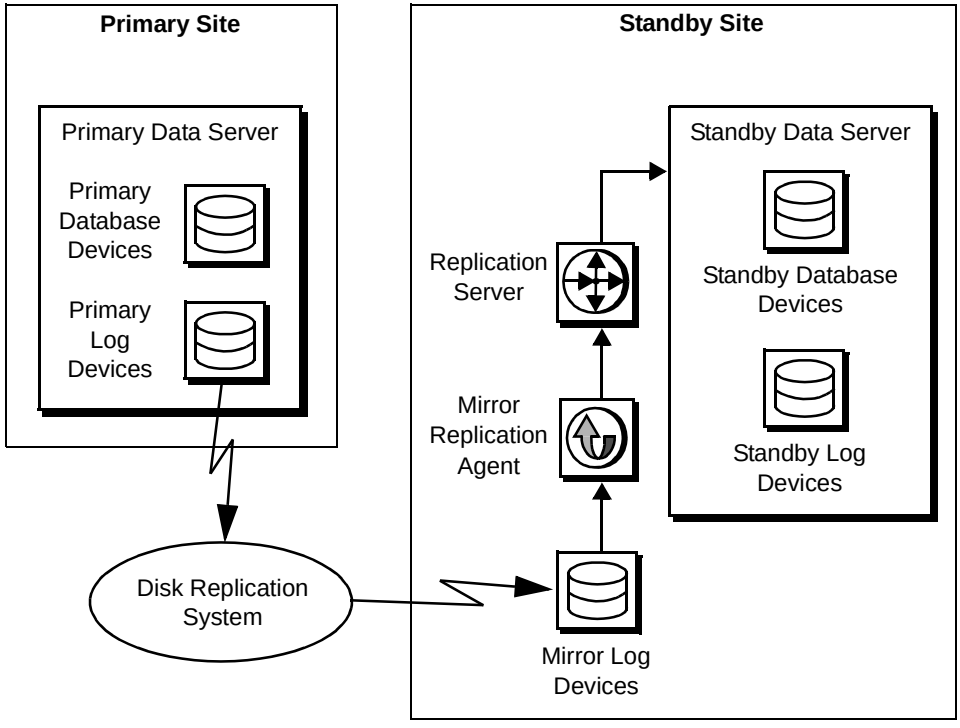
- Standby databases protected from disk corruption (by logical, not literal, replication)
- Synchronous replication, with zero data loss *and* guaranteed data integrity (transactional consistency)
- Complete coverage for databases, as well as non-database systems

How Mirror Activator works

A Mirror Activator 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 system configuration. Arrows illustrate the flow of mirrored log device data and replicated transactions during normal Mirror Activator system operation, that is, while replicating transactions from the primary database to the standby database.

Figure 1-1: Mirror Activator system



Note The disk replication system is the Mirror Activator 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 system components

The main components of a Mirror Activator system are:

- Primary database
- Disk replication system
- Mirror log devices
- Mirror Replication Agent
- Replication Server
- 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.

Standby database

The standby database is the destination of transactions that are replicated from the primary database. During normal Mirror Activator 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.

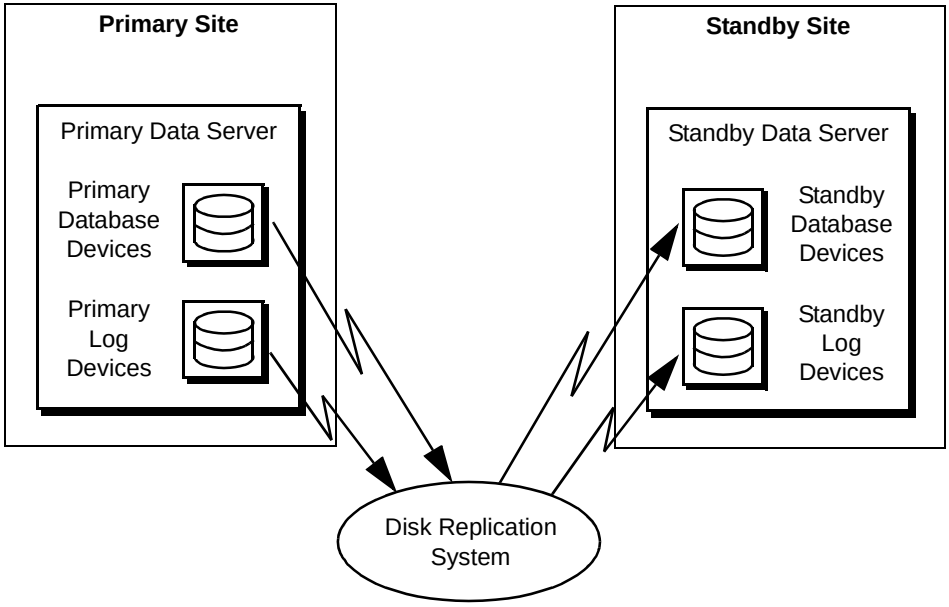
Materializing the standby database

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 the movement of snapshot data, 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

effectively 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. The disk replication system is then 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 system.

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

Once 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 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 system operation, the disk replication system mirrors primary database log devices synchronously, which guarantees zero data loss—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 in the event of network failures at the standby site, or a failure of the standby database itself.

The Mirror Activator 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 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 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 stand-alone process, independent of any other Mirror Activator 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 system component. The Mirror Replication Agent host must have local access to the mirror log devices.

Mirror Replication Agent instances

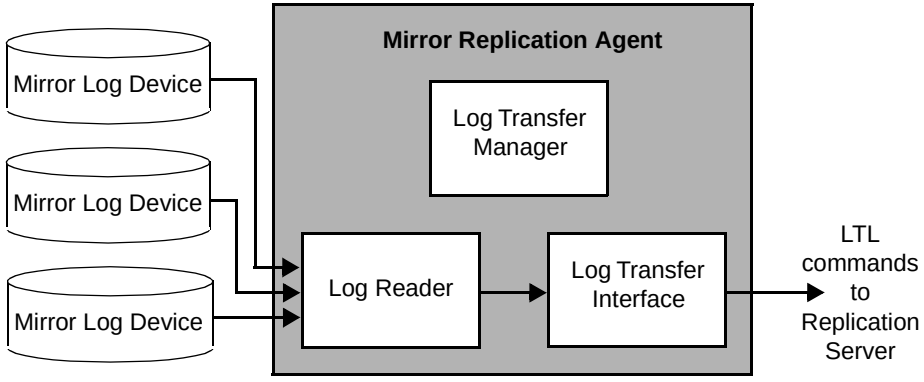
You must create one instance of the Mirror Replication Agent for each primary database that you want to replicate transactions from. Each Mirror Replication Agent instance is an independent process, with its own instance directories to house its configuration file, system log files, and system data repository.

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 – 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 – 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 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

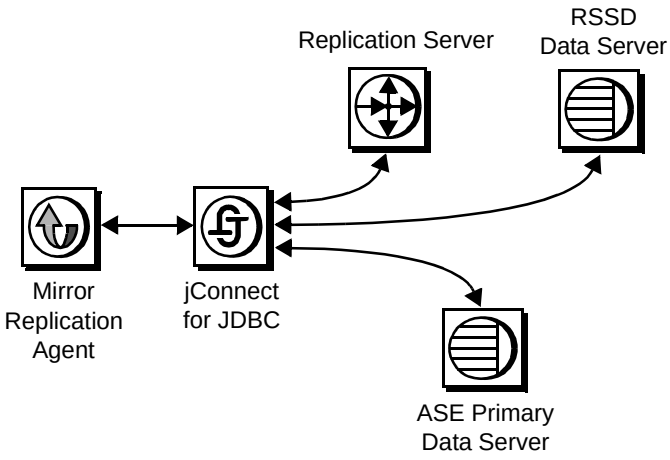
For network connections, Mirror Replication Agent uses the Java Database Connectivity (JDBC) protocol, as implemented by the Sybase JDBC driver, jConnect™ for JDBC™.

Each Mirror Replication Agent instance uses a single instance of jConnect for JDBC to communicate with all Open Client™ and Open Server™ applications, including the Adaptive Server® Enterprise primary data server.

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 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 system components:

- Primary data server
- Replication Server

Mirror Replication Agent maintains a connection with the primary data server to reserve the logscan context, which gives it control of the secondary truncation point in the primary database transaction log.

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 system.

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

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

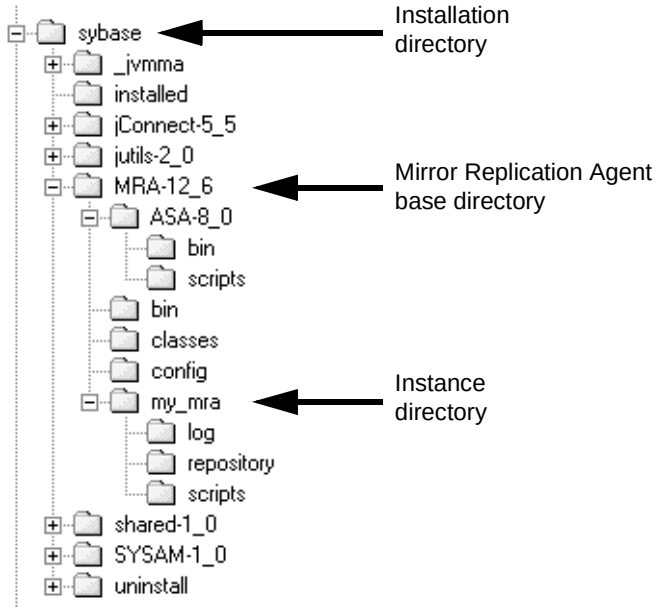
You can create and run more than one Mirror Replication Agent instance on a single host machine, but each instance must have a unique name and a unique port number.

For more information, see “Creating a Mirror Replication Agent instance” on page 20.

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

Using Mirror Replication Agent utilities

Two utilities are provided with Mirror Replication Agent:

- mra – starts a Mirror Replication Agent instance, or returns the Mirror Replication Agent software version number.
- mra_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.

Setting the SYBASE environment

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

- 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 make sure 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, enter:

  ```
  SYBASE
  ```

- On UNIX platforms, enter:

  ```
  source SYBASE.sh
  ```

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 mra utility

The Mirror Replication Agent mra utility provides the following functions:

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

See "Using the mra_admin utility" on page 18 for information about creating a Mirror Replication Agent instance.

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

Syntax

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

Parameters

-help

The option that returns command usage information.

Note You can also invoke mra 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 version number.

Example

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

```
mra -i my_mra -replicate
```

See "Starting the Mirror Replication Agent" on page 27 for more information about starting a Mirror Replication Agent instance.

See "Mirror Replication Agent states" on page 79 for more information about *Admin* and *Replicating* states.

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.

See Chapter 6, "Troubleshooting Mirror Replication Agent," for more information.

Using the mra_admin utility

The Mirror Replication Agent mra_admin utility provides the following functions:

- Create, copy, delete, and verify Mirror Replication Agent instances
- List all valid Mirror Replication Agent instances on the Mirror Replication Agent host machine
- Return the path of the Mirror Replication Agent installation directory

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

Syntax

mra_admin [option [create options]] [*inst_name*]

Note You can also invoke mra_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.

Note When you use the -c option to create a new Mirror Replication Agent instance, you must also use the -p option to specify a port number for the 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 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. See "Copying a Mirror Replication Agent configuration" on page 22 for more information.

-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 mra_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

-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.

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 mra_admin with the -c option.

The complete syntax is:

mra_admin -c *new_inst* -p *port_num*

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.

See "Copying a Mirror Replication Agent configuration" on page 22 for information about creating a Mirror Replication Agent instance based on the configuration of an existing instance.

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

Note You must set the SYBASE environment before you invoke the Mirror Replication Agent mra_admin utility. See "Setting the SYBASE environment" on page 15 for more information.

❖ **To create 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%\MRA-12_6\bin
  ```

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

- On UNIX platforms, enter:

  ```
  cd $SYBASE/MRA-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 mra_admin utility to create a new Mirror Replication Agent instance:

```
mra_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 mra_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 mra_admin with the -v option, and specify the name of the new Mirror Replication Agent instance:

  ```
  mra_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 mra_admin with the -l option:

  ```
  mra_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 mra_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 mra utility to start the instance so that it can be administered and configured. See “Starting the Mirror Replication Agent” on page 27 for more information.

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. See "Creating the Mirror Replication Agent administrator login" on page 33 for more information.

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 mra_admin with the -c option and -f option.

The complete syntax is:

mra_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. See "Creating a Mirror Replication Agent instance" on page 20 for information about creating a Mirror Replication Agent instance with the default configuration.

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 mra_admin utility. See "Setting the SYBASE environment" on page 15 for more information.

❖ **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, enter:

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

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

- On UNIX platforms, enter:

```
cd $SYBASE/MRA-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 mra_admin utility to create a new Mirror Replication Agent instance whose configuration is based on the configuration of an existing instance:

```
mra_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 mra_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 mra_admin with the -v option, and specify the name of the new Mirror Replication Agent instance:

```
mra_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 mra_admin with the -l (lowercase L) option:

  ```
  mra_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 mra_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).

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
rasd_backup_dir
rasd_database
rasd_mirror_tran_log
rasd_trace_log_dir
rasd_tran_log
rasd_tran_log_mirror
rs_source_db
rs_source_ds

See Chapter 5, “Mirror Replication Agent Configuration Parameters,” for more information.

After you create a Mirror Replication Agent instance, you can use the mra 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. See “Creating the Mirror Replication Agent administrator login” on page 33 for more information.

Deleting a Mirror Replication Agent instance

You can delete a Mirror Replication Agent instance at any time by invoking mra_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. See “Shutting down Mirror Replication Agent” on page 73 for more information.
- If the Mirror Replication Agent software is installed on a Microsoft Windows platform, make sure that none of the files in the instance subdirectories are open, and no application or window is accessing the instance subdirectories.

Note You must set the SYBASE environment before you invoke the Mirror Replication Agent mra_admin utility. See “Setting the SYBASE environment” on page 15 for more information.

❖ **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%\MRA-12_6\bin
```

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

- On UNIX platforms, enter:

  ```
  cd $SYBASE/MRA-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 mra_admin utility with the -d option to delete a Mirror Replication Agent instance:

```
mra_admin -d inst_name
```

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

After you invoke mra_admin with the -d option, 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 mra_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. See “Shutting down Mirror Replication Agent” on page 73 for more information.

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

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

  ```
  mra_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 mra_admin utility with the -l option:

  ```
  mra_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 mra_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

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).

There are two ways you can start a Mirror Replication Agent instance:

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

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

Starting an instance with the mra utility

When you start the Mirror Replication Agent with the mra utility, you can specify the instance start-up state. If you do *not* specify a start-up state when you invoke the mra 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 mra utility. See "Setting the SYBASE environment" on page 15 for more information.

❖ **To start Mirror Replication Agent with the *mra* 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%\MRA-12_6\bin
  ```

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

- On UNIX platforms, enter:

  ```
  cd $SYBASE/MRA-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 mra utility to start the Mirror Replication Agent instance:

```
mra -iinst_name
```

or

```
mra -iinst_name -state
```

where:

- *inst_name* is the server name of the 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_mra" in *Replicating* state, enter:

```
mra -i my_mra -replicate
```

See "Using the mra utility" on page 16 for more information.

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 mra 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 need not set the SYBASE environment before you invoke the *RUN* script, because the *RUN* script sets the Sybase environment before it starts the Mirror Replication Agent instance.

❖ **To start Mirror Replication Agent 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%\MRA-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/MRA-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_mra," enter:

  ```
  RUN_my_mra
  ```

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 software, and both isql and SQL Advantage are provided with the Adaptive Server software.

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_mra,” 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_mra]
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 31 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_mra," enter:

```
isql -Usa -P -Smy_mra
```

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 33 for more information.

Once you have successfully logged in to the administration port, you can use Mirror Replication Agent commands to administer the Mirror Replication Agent instance. See Chapter 4, "Mirror Replication Agent Command Reference," for more information.

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 1d and 1e on the "Installation and Setup Worksheet" in the Mirror Replication Agent *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 system components:

- Primary data server
- Replication Server
- RSSD

Note The term "RSSD" in this document refers to both RSSD and ERSSD, unless there is a difference.

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 Replication Agent *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, and to reserve the logscan context.

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 Adaptive Server to perform this procedure.

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

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Add the Mirror Replication Agent login name to the primary data server:

```
use master
sp_addlogin mra_pds_user, mra_pds_pwd, pdb
```

where:

- *mra_pds_user* is the Mirror Replication Agent user login name.
- *mra_pds_pwd* is the password for the user login name.
- *pdb* is the name of the primary database.

3 Grant the Replication role to the Mirror Replication Agent login name:

```
grant role replication_role to mra_pds_user
```

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

4 Add the Mirror Replication Agent user login name to the primary database:

```
use pdb
sp_adduser mra_pds_user
```

where:

- *pdb* is the name of the primary database.
- *mra_pds_user* is the Mirror Replication Agent user login name.

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.

Note If the Replication Server and databases were previously configured for a Replication Server warm standby application, the Mirror Replication Agent can use the Replication Server login that was created for the ASE RepAgent thread. See “Converting a warm standby application to a Mirror Activator system” on page 63 for more information.

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 mra_rs_user set password mra_rs_pwd
```

where:

- *mra_rs_user* is the Mirror Replication Agent user login name.
- *mra_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 mra_rs_user
```

where *mra_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 RSSD or ERSSD to obtain information about replication definitions.

The following sections describe procedures for:

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

See "Setting up the ERSSD user login for Mirror Replication Agent" on page 38 for information about setting up a user login name for the Mirror Replication Agent instance in an ERSSD managed by Adaptive Server Anywhere.

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

❖ **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 mra_rssd_user, mra_rssd_pwd, rssd_db
```

where:

- *mra_rssd_user* is the Mirror Replication Agent user login name.
- *mra_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 mra_rssd_user
```

where:

- *rssd_db* is the name of the primary database.
- *mra_rssd_user* is the Mirror Replication Agent user login name.

After you set up the Mirror Replication Agent user login in the RSSD data server, 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.

See "Setting up the RSSD user login for Mirror Replication Agent" on page 37 for information about setting up a user login name for the Mirror Replication Agent instance in an RSSD managed by Adaptive Server Enterprise.

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

❖ **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 mra_rssd_user
identified by mra_rssd_pwd
```

where:

- *mra_rssd_user* is the Mirror Replication Agent user login name.
- *mra_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 mra_rssd_user
```

where *mra_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 system component, it uses values stored in its configuration parameters to define the following 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.

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.

See Chapter 5, “Mirror Replication Agent Configuration Parameters,” for more information about the rs_source_db and rs_source_ds parameters.

You can use the “Installation and Setup Worksheet,” in the Mirror Replication Agent *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 27 for more information.

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

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 database name of the primary database.

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

   ```
   ra_config pds_username, mra_pds_user
   ```

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

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

   ```
   ra_config pds_password, mra_pds_pwd
   ```

   where *mra_pds_pwd* is the password for the user login name that 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 82 for more information.

Note If the Replication Server and databases were previously configured for a Replication Server warm standby application, the Mirror Replication Agent must use the same values for its rs_source_db and rs_source_ds parameters as the ASE RepAgent thread used for its connect database and connect dataserver parameters. See “Converting a warm standby application to a Mirror Activator system” on page 63 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 user login name for the Mirror Replication Agent instance:

   ```
   ra_config rs_username, mra_rs_user
   ```

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

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

   ```
   ra_config rs_password, mra_rs_pwd
   ```

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

6 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 Mirror Replication Agent uses in the LTL connect source command.

7 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 Mirror Replication Agent uses in the LTL connect source command.

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

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 RSSD host name:

```
ra_config rssd_hostname, rssd_host
```

where *rssd_host* is the network name of the RSSD host machine.

3 Specify the RSSD port number:

```
ra_config rssd_port_number, NNN
```

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

4 Specify the RSSD database name:

```
ra_config rssd_database_name, rssd_db
```

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

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

```
ra_config rssd_username, mra_rssd_user
```

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

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

```
ra_config rssd_password, mra_rssd_pwd
```

where *mra_rssd_pwd* is the password for the user login name that 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 82.

Setting up the Mirror Activator system

You have two options when setting up the Mirror Activator system:

- Setting up a *new* Mirror Activator system (that is, using a primary database that was *not* previously configured for a Replication Server warm standby application)
- Converting an existing Replication Server warm standby application to a Mirror Activator system

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

- Install Mirror Activator system components
- Configure Mirror Activator system components
- Materialize and set up synchronous replication to devices at the standby site:
  - For a new Mirror Activator 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.

- For a warm standby application that you are converting to a Mirror Activator system, you must materialize and set up synchronous replication to the mirror log devices.

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

If you are converting a warm standby application to a Mirror Activator system, the only Mirror Activator software you must install is the Mirror Replication Agent. The existing Replication Server and databases in a warm standby application can be reconfigured for the Mirror Activator system. See "Converting a warm standby application to a Mirror Activator system" on page 63 for more information.

Note The setup tasks for some components of the Mirror Activator 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 system, see Table 2-1 on page 49.
- If you are converting a warm standby application to a Mirror Activator system, see Table 2-2 on page 65.

Installation and basic configuration

For information about installing Mirror Activator 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 system components:

- Mirror Replication Agent *Installation Guide*
- Replication Server installation and configuration guides for your platform

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 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 the Adaptive Server Enterprise documentation.

Mirror Activator configuration requirements

This section describes general configuration requirements, by component, for *all* Mirror Activator systems, including both new Mirror Activator systems, and warm standby applications converted to Mirror Activator systems.

For detailed setup and configuration procedures:

- See “Setting up Mirror Replication Agent connectivity” on page 34 to configure the Mirror Replication Agent connection parameters
- See “Setting up a new Mirror Activator system” on page 48 to set up the primary and standby databases, Mirror Replication Agent, and Replication Server in a *new* Mirror Activator system
- See “Converting a warm standby application to a Mirror Activator system” on page 63 to set up the primary database, Mirror Replication Agent, and Replication Server in an existing Replication Server warm standby application that you are converting to a Mirror Activator 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 data server and primary database, with the Replication role granted
- Maintenance User login name (as specified in the Replication Server create connection command) added to the primary data server, with the Replication role granted

- Maintenance User login name added to the primary database
- Replication Server objects (tables and procedures) created and set up in the primary database, with appropriate permissions granted to the Maintenance User name in the primary database
- Initialized to mark the entire database for replication (and to set the secondary truncation point, if the primary database was *not* previously configured for a warm standby application)
- RepAgent thread stopped and disabled in the primary database (if the primary database was previously configured for a warm standby application)

See "Converting a warm standby application to a Mirror Activator system" on page 63 for more information.

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 (to materialize the standby database for a new Mirror Activator system)

Note If you are converting a warm standby application to a Mirror Activator system, you need not materialize the standby database.

- Ready to copy a snapshot image of all primary log devices to the mirror log devices at the standby site
- Ready to copy a snapshot image of all standby database data and log devices to the primary database devices (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 ra_init command to set up the RASD
- Paths correctly defined for access to all mirror log devices

Replication Server

The Replication Server must be configured as follows:

- Mirror Replication Agent user login name, with connect source permission granted
- Logical (warm standby) connection defined for the primary and standby databases
- Database connections defined for the primary and standby databases (as the *active* and *standby* for the logical connection)

Standby database

If you are converting a warm standby application to a Mirror Activator system, you need not change the standby database configuration.

The following list describes the *minimum* standby database configuration required to participate in the Mirror Activator 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 data server, with the Replication role granted
- Maintenance User login name added to the standby database
- 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

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). When 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 system

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

- Primary database
- Mirror Replication Agent
- Replication Server
- Standby database

Use the setup and configuration tasks in this section if you are setting up a *new* Mirror Activator system (that is, using a primary database that was *not* previously configured for a Replication Server warm standby application).

If you are converting an existing warm standby application to a Mirror Activator system, see “Converting a warm standby application to a Mirror Activator system” on page 63 for setup and configuration tasks.

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

The checklist in Table 2-1 assumes that:

- The primary database was *not* previously configured for a Replication Server warm standby application.
- 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 already completed all of the tasks described in “Setting up Mirror Replication Agent connectivity” on page 34.

Note If the primary database was previously configured for a Replication Server warm standby application, do *not* use the checklist in Table 2-1. You must instead use the setup and configuration tasks described in “Converting a warm standby application to a Mirror Activator system” on page 63.

When setting up a new Mirror Activator system, you must perform the tasks in Table 2-1 in the order they are shown. If you deviate from this sequence, the results may be unpredictable, and you may have to back out of the entire process and start over.

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

Task	Description
1	Set up the Mirror Activator system Maintenance User in the primary data server and in the primary database.
2	Set up the Mirror Activator system Maintenance User in the standby data server.
3	Set up the Replication Server database objects in the primary database.
4	Initialize the primary database using the Mirror Replication Agent pdb_init command. **Note** After the primary dabatase is initialized, you must *not* allow any DDL operations before it is quiesced in step 7.
5	Add a logical connection and database connections for the primary and standby databases to the Replication Server.
6	Shut down the standby data server. **Note** This step is required *only* when you use snapshot materialization for a standby database in Adaptive Server version 12.5.0.3.
7	Quiesce the primary database to suspend update activity.
8	Materialize the standby database data and log devices and the mirror log devices at the standby site, and set up the disk replication system for synchronous replication to the mirror log devices.
9	Initialize the Mirror Replication Agent using the ra_init command, and set the paths to the mirror log devices. **Note** You can initialize the Mirror Replication Agent concurrently with materialization (step 8).
10	Resume update activity on the primary database, *after* the device materialization and Mirror Replication Agent initialization are complete.
11	• If you use snapshot materialization for a standby database in Adaptive Server version 12.5.0.3, start the standby data server. • If you use the snapshot materialization mount database option, mount the standby database and bring it online.
12	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 the Maintenance User in the primary database

Setting up the Maintenance User involves:

- Adding the Maintenance User login to the primary data server
- Granting the Replication role to the Maintenance User login
- Adding the Maintenance User to the primary database

Use the following procedure to set up the Maintenance User login in the primary database.

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **To set up the Maintenance User in the primary database**

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Add the Maintenance User login name to the primary data server:

```
use master
sp_addlogin mra_maint, mra_maint_pwd, pdb
```

where:

- *mra_maint* is the Maintenance User login name.
- *mra_maint_pwd* is the password for the Maintenance User.
- *pdb* is the name of the primary database.

Note The Maintenance User login can be the same as the Mirror Replication Agent user login in the primary database. This can simplify managing user permissions and setting up the Mirror Activator system.

3 Grant the Replication role to the Maintenance User:

```
grant role replication_role to mra_maint
```

where *mra_maint* is the Maintenance User login name.

4 Add the Maintenance User to the primary database:

```
use pdb
sp_adduser mra_maint
```

where:

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

After you set up the Maintenance User in the primary data server, verify that the Maintenance User can log in to the primary data server and access the primary database.

Set up the Maintenance User in the standby data server

Setting up the Maintenance User involves:

- Adding the Maintenance User login to the standby data server
- Granting the Replication role to the Maintenance User login

You need not add the Maintenance User to the standby database, because when you materialize the standby database, users in the primary database are copied to the standby database.

Use the following procedure to set up the Maintenance User login in the standby data server.

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

❖ **To set up the Maintenance User in the standby data server**

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

2 Add the Maintenance User login to the standby data server:

```
use master
sp_addlogin mra_maint, mra_maint_pwd, sdb
```

where:

- *mra_maint* is the Maintenance User login name.
- *mra_maint_pwd* is the password for the Maintenance User.
- *sdb* is the name of the standby database.

3 Grant the Replication role to the Maintenance User:

```
grant role replication_role to mra_maint
```

where *mra_maint* is the Maintenance User login name.

After you set up the Maintenance User in the standby data server, verify that the Maintenance User can log in to the standby data server.

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
- Manage the secondary truncation point in the database
- Keep track of successfully replicated transactions

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 *primary database installation script*. The primary database installation script is a SQL script named:

- *rsinspri.sql* on Microsoft Windows platforms
- *rs_install_primary.sql* on UNIX platforms

The primary database installation script resides in the Replication Server *scripts* directory (for example, *%SYBASE%\REP-12_6\scripts* on Microsoft Windows platforms).

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **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 Adaptive Server with a System Administrator user role.

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

```
use pdb
grant all on rs_lastcommit to mra_maint
grant execute on rs_get_lastcommit to mra_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:

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

5 Disable replication for the rs_lastcommit table in the primary database:

```
use pdb
sp_setreplicate rs_lastcommit, false
```

where *pdb* is the name of the primary database.

6 Mark the rs_update_lastcommit procedure for replication:

```
use pdb
sp_setrepproc rs_update_lastcommit, "function"
```

where *pdb* is the name of the primary database.

7 Set the primary data server to ignore the secondary truncation point in the primary database log:

```
use pdb
dbcc settrunc("ltm", "ignore")
```

where *pdb* is the name of the primary database.

Initialize the primary database

You must initialize the primary database using the Mirror Replication Agent pdb_init command. Initializing the primary database does the following:

- Verifies that the primary database configuration is correct for the Mirror Activator system
- Marks the primary database for replication (equivalent to executing sp_reptostandby in the primary database)
- Sets the secondary truncation point to the end of the log

Note After you initialize the primary dabatase, you must *not* allow any DDL operations in the primary database before it is quiesced later in the setup procedure.

❖ **To initialize the primary database**

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
```

Warning! Do *not* use the pdb_init move_truncpt option when you initialize a primary database that was previously configured for a Replication Server warm standby application. See "Converting a warm standby application to a Mirror Activator system" on page 63 for more information.

2 Initialize the primary database:

```
pdb_init move_truncpt
```

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 the logical connection for the primary and standby databases:

```
create logical connection to
   logical_ds.logical_db
```

where:

- *logical_ds* is the name of the logical data server in a warm standby application.
- *logical_db* is the name of the logical database in a warm standby application.

The logical data server name (*logical_ds*) and logical database name (*logical_db*) need not be the same as the actual names of the primary data server and primary database. See the Replication Server *Administration Guide* for more information about logical connections.

3 Create a database connection for the primary database:

```
create connection to pds.pdb
   set error class rs_sqlserver_error_class
   set function string class
      rs_sqlserver_function_class
   set username "mra_maint"
   set password  "mra_maint_pwd"
   with log transfer on
   as active for logical_ds.logical_db
```

where:

- *pds* is the name of the primary data server.
- *pdb* is the name of the primary database.
- *mra_maint* is the Maintenance User login for the primary database.
- *mra_maint_pwd* is the password for the Maintenance User.
- *logical_ds* is the logical data server identified in the logical connection.
- *logical_db* is the logical database identified in the logical connection.

4 Create a database connection for the standby database:

```
create connection to sds.sdb
   set error class rs_sqlserver_error_class
   set function string class
      rs_sqlserver_function_class
   set username "mra_maint"
   set password  "mra_maint_pwd"
   with log transfer on
   as standby for logical_ds.logical_db
```

where:

- *sds* is the name of the standby data server.
- *sdb* is the name of the standby database.
- *mra_maint* is the Maintenance User login for the standby database.
- *mra_maint_pwd* is the password for the Maintenance User.
- *logical_ds* is the logical data server identified in the logical connection.
- *logical_db* is the logical database identified in the logical connection.

Shut down the standby data server

If you use snapshot materialization for a standby database in Adaptive Server version 12.5.0.3, you must shut down the standby data server to take the standby database offline for materialization.

Note Skip this procedure if you use the snapshot materialization mount database option for a standby database in Adaptive Server version 12.5.1 or later.

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 Adaptive Server to perform this procedure.

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

1 Log in to the standby Adaptive Server 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.

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **To quiesce the primary database**

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Quiesce the primary database.

- If you use snapshot materialization for a standby database in Adaptive Server version 12.5.0.3, use the following command to quiesce the primary database:

```
quiesce database MA_setup hold pdb
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *pdb* is the name of the primary database.

- If you use the snapshot materialization mount database option for a standby database in Adaptive Server version 12.5.1 or later, use the following command to quiesce the primary database:

```
quiesce database MA_setup hold pdb
for external dump to pdb_manifest
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *pdb* is the name of the primary database.
- *pdb_manifest* is the name of the manifest file.

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 Chapter 4, “Mirror Replication Agent Command Reference,” for more information about the ra_devicepath and ra_helpdevice commands.

Resume update activity on the primary database

After the standby database materialization and Mirror Replication Agent initialization are complete, release the quiesce hold 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.

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **To resume update activity on the primary database**

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Release the quiesce hold on the primary database:

```
quiesce database MA_setup release
```

where *MA_setup* is a user-defined tag that identifies the suspended database.

Start the standby database

If you use snapshot materialization for a standby database in Adaptive Server version 12.5.0.3, you must restart the standby data server to bring the standby database online for replication.

Note Skip this procedure if you use the snapshot materialization mount database option for a standby database in Adaptive Server version 12.5.1 or later. See “Mount the standby database and bring it online” on page 61 for more information.

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

Note You must have an operating system user login (such as the “sybase” user) with appropriate permissions in the Adaptive Server directories and database devices to perform this procedure. See the Adaptive Server *Configuration Guide* for your platform for more information.

❖ **To start the standby data server**

1 Log in to the standby Adaptive Server host machine with a user login that has appropriate permissions (for example, the "sybase" user).

2 Open an operating system command prompt window.

3 At the operating system prompt, navigate to the directory where the startup *RUN_server_name* script resides. For example, the default directory would be *%SYBASE%\ASE_12-5\install* on Microsoft Windows platforms, where *%SYBASE%* is the Adaptive Server installation directory.

- On Microsoft Windows platforms, enter:

```
cd %SYBASE%\ASE_12-5\install
```

- On UNIX platforms, enter:

```
cd $SYBASE/ASE_12-5/install
```

4 Start the standby data server using the RUN script:

```
RUN_sds
```

where *sds* is the name of the standby data server.

Note The standby data server will start up in "recovery" mode, and it may take a while for the standby database to come online.

Mount the standby database and bring it online

If you use the snapshot materialization mount database option for a standby database in Adaptive Server version 12.5.1 or later, you must:

- Use the mount command to mount the database on the standby data server, from the devices that were mirrored to the standby site
- Use the online database command to bring the standby database online after it is mounted on the standby data server

Note Skip this procedure if you use snapshot materialization for a standby database in Adaptive Server version 12.5.0.3. See "Start the standby database" on page 60 for more information.

See Appendix A, “Materializing Databases,” for more information about materialization options for the standby database.

Note You must have a System Administrator or Database Owner user role in the standby Adaptive Server to perform this procedure.

❖ **To mount the standby database and bring it online**

1 Log in to the standby Adaptive Server with a System Administrator or Database Owner user role.

2 Mount the database on the standby data server:

```
mount database all from pdb_manifest
```

where *pdb_manifest* is the manifest file created by the quiesce command at the primary database.

When you invoke mount, Adaptive Server performs all of the required supporting activities, including adding database devices and activating them, creating the catalog entries for the new database, and recovering the database.

3 After the database is recovered on the standby data server, bring the database online:

```
online database sdb
```

where *sdb* is name of the standby database.

The names of the standby database and primary database must be the same.

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 6, "Troubleshooting Mirror Replication Agent," for more information.

Resume the standby database connection

To start Mirror Activator 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 6, "Troubleshooting Mirror Replication Agent," or refer to the Replication Server *Troubleshooting Guide* for more information.

Converting a warm standby application to a Mirror Activator system

This section describes the setup and configuration tasks that are required to *convert* an existing Replication Server warm standby application to a Mirror Activator system.

If you are setting up a *new* Mirror Activator system (that is, using a primary database that was *not* previously configured for a warm standby application), see "Setting up a new Mirror Activator system" on page 48 for complete setup and configuration tasks.

Table 2-2 provides a checklist of the tasks required to configure software components and set up the Mirror Activator system for replication when you convert an existing warm standby application to a Mirror Activator system.

The checklist in Table 2-2 assumes that:

- The Replication Server and primary and standby databases are already configured for a warm standby application, and the warm standby application is functioning properly to replicate transactions from the primary database to the standby database.
- You need not materialize the standby database because it has been maintained by the Replication Server, and it already contains data and schema that are identical to the primary database.
- You have already completed all of the tasks described in "Setting up Mirror Replication Agent connectivity" on page 34, *except*:
  - Creating the Replication Server user login name (for the Mirror Replication Agent), and
  - Setting up the Mirror Replication Agent configuration parameters for the Replication Server connection.

Note If the Replication Server and primary and standby databases were *not* previously configured for a Replication Server warm standby application, do *not* use the task checklist in Table 2-2. You must instead use the setup and configuration tasks described in "Setting up a new Mirror Activator system" on page 48.

When converting an existing warm standby application to a Mirror Activator system, you must perform all of the tasks in Table 2-2 in the order they are shown. If you deviate from this sequence, the results may be unpredictable, and you may have to back out of the entire process and start over.

Table 2-2: Setup and configuration for converting a warm standby application to a Mirror Activator system

Task	Description
1	Materialize the mirror log devices, and set up the disk replication system for synchronous replication to the mirror log devices.
2	Set up the Mirror Replication Agent configuration parameters for the Replication Server connection.
3	Stop and disable the RepAgent thread in the primary database. **Note** You must preserve the secondary truncation point when you disable the RepAgent thread.
4	Initialize the primary database using the Mirror Replication Agent pdb_init command. **Note** After the primary dabatase is initialized, you must *not* allow any DDL operations before it is quiesced in step 5.
5	Quiesce the primary database.
6	Initialize the Mirror Replication Agent using the ra_init command, and set the paths to the mirror log devices.
7	Resume update activity in the primary database after Mirror Replication Agent initialization is complete.
8	Resume the Mirror Replication Agent to put it in *Replicating* state.

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

Materialize the mirror log devices

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

- 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

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.

Set up Mirror Replication Agent connection parameters for Replication Server

Complete all the tasks described in "Setting up Mirror Replication Agent connectivity" on page 34, *except* the following:

- Creating the Replication Server user login name
- Setting up the Mirror Replication Agent configuration parameters for the Replication Server connection

When connecting to the Replication Server, the Mirror Replication Agent can use the login name that was created for the primary database RepAgent thread.

Use the following procedure to find the values of the RepAgent thread configuration parameters.

Note You must have a System Administrator or Database Owner user role in the primary Adaptive Server to perform this procedure.

❖ **To find the Replication Server login name for the RepAgent thread**

1 Log in to the primary Adaptive Server with a System Administrator or Database Owner user role.

2 View the current values of the RepAgent thread configuration parameters in the primary database:

```
use pdb
sp_config_rep_agent pdb
```

where *pdb* is the name of the primary database.

Make a note of the current values returned for the following RepAgent thread parameters:

- rs username – Replication Server user login for the RepAgent thread
- connect dataserver – primary data server name in the Replication Server database connection
- connect database – primary database name in the Replication Server database connection

---

Note The password for the Replication Server user login is not displayed with the other RepAgent thread parameters. Consult the System Administrator or System Security Officer for the Replication Server to obtain the password that the Mirror Replication Agent must use.

Use the following procedure to set up the Mirror Replication Agent connection configuration for Replication Server.

❖ **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 user login name for the Mirror Replication Agent instance:

   ```
   ra_config rs_username, mra_rs_user
   ```

   where *mra_rs_user* is the value of the RepAgent thread rs username parameter.

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

   ```
   ra_config rs_password, mra_rs_pwd
   ```

   where *mra_rs_pwd* is the password you received from the System Administrator.

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

   ```
   ra_config rs_source_ds, pds
   ```

   where *pds* is the value of the RepAgent thread connect dataserver parameter.

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

```
ra_config rs_source_db, pdb
```

where *pdb* is the value of the RepAgent thread connect database parameter.

After you set up the Mirror Replication Agent connection configuration parameters, use the Mirror Replication Agent test_connection RS command to test connectivity between the Mirror Replication Agent and the Replication Server. See “Testing network connectivity” on page 82 for more information.

Stop and disable the RepAgent thread in the primary database

In the Mirror Activator system, the Mirror Replication Agent must control the secondary truncation point in the primary database. This requires you to stop and disable the RepAgent thread in the primary database when you convert an existing warm standby application to a Mirror Activator system.

Warning! You *must* use the preserve secondary truncpt option when you execute sp_config_rep_agent to disable the RepAgent thread. If you do not preserve the secondary truncation point in the primary database, you will have to re-materialize the standby database *before* you resume replication to prevent data loss.

Use the following procedure to stop and disable the RepAgent thread in the primary database.

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **To stop and disable the RepAgent thread**

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Stop the RepAgent thread in the primary database:

```
use pdb
sp_stop_rep_agent pdb
```

where *pdb* is the name of the primary database.

3 Disable the RepAgent thread in the primary database:

```
sp_config_rep_agent pdb, 'disable', 'preserve secondary truncpt'
```

where *pdb* is the name of the primary database.

Disabling the RepAgent thread allows the Mirror Replication Agent to reserve the logscan context in the primary database.

Initialize the primary database

You must initialize the primary database using the Mirror Replication Agent pdb_init command. Initializing the primary database does the following:

- Verifies that the primary database configuration is correct for the Mirror Activator system
- Marks the primary database for replication (equivalent to executing sp_reptostandby in the primary database)

Note After you initialize the primary dabatase, you must *not* allow any DDL operations in the primary database before it is quiesced (the next setup task).

❖ **To initialize the primary database**

1 Log in to the Mirror Replication Agent administration port.

Warning! Do *not* use the move_truncpt option of the pdb_init command when you initialize the primary database. If you do not preserve the secondary truncation point in the primary database, you will have to re-materialize the standby database *before* you resume replication to prevent data loss.

---

2 Initialize the primary database:

```
pdb_init
```

Quiesce the primary database

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

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **To quiesce the primary database**

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Quiesce the primary database to suspend update activities:

```
quiesce database MA_setup hold pdb
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *pdb* is the name of the primary database.

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.

❖ **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
```

You may need to alter the log device path(s) returned by the primary data server during Mirror Replication Agent 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 Chapter 4, “Mirror Replication Agent Command Reference,” for more information about the ra_devicepath and ra_helpdevice commands.

Resume update activity in the primary database

After the Mirror Replication Agent initialization is complete, release the quiesce hold to resume update activity in the primary database.

You must *not* release the quiesce hold on the primary database until the Mirror Replication Agent is initialized, with correct paths defined for all mirror log devices.

Note You must have a System Administrator user role in the primary Adaptive Server to perform this procedure.

❖ **To resume update activity on the primary database**

1 Log in to the primary Adaptive Server with a System Administrator user role.

2 Release the quiesce hold on the primary database:

```
quiesce database MA_setup release
```

where *MA_setup* is a user-defined tag that identifies the suspended 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 6, “Troubleshooting Mirror Replication Agent,” for more information.

CHAPTER 3

Administering Mirror Replication Agent

This chapter describes administrative tasks and procedures for Mirror Replication Agent.

See the Mirror Replication Agent *Installation Guide* for information about installing the Mirror Replication Agent software.

See Chapter 2, “Setup and Configuration,” for information about setting up the Mirror Activator system, including Mirror Replication Agent.

Note 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 log in to the Mirror Replication Agent administration port.

Shutting down Mirror Replication Agent

Each Mirror Replication Agent instance can be started and shut down independently of all other components in a Mirror Activator system, and independently of other Mirror Replication Agent instances.

See “Starting the Mirror Replication Agent” on page 27 for more information about how to start a Mirror Replication Agent instance.

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. See "Stopping replication in the Mirror Replication Agent" on page 76 for more information.

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.

- 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.

Starting replication in the Mirror Replication Agent

When you start replication in the Mirror Replication Agent:

- Mirror Replication Agent opens network connections to the primary database, Replication Server, and RSSD.
- The internal Log Reader and Log Transfer Interface components begin their 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.

See the description of the resume command in Chapter 4, “Mirror Replication Agent Command Reference,” for more detailed information about how Mirror Replication Agent starts its normal replication processing.

Note Before you attempt to replicate transactions from a mirror log device, use the checklist in “Setting up the Mirror Activator system” on page 43 to verify that your Mirror Activator system is set up properly.

The Mirror Replication Agent instance must be running before you can start replication. See “Starting the Mirror Replication Agent” on page 27 for more information.

❖ **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 6, “Troubleshooting Mirror Replication Agent,” for more information.

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.

Some administrative tasks require the Mirror Replication Agent instance to be in *Admin* state. In a functioning Mirror Activator 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. See “Quiescing the Mirror Replication Agent” on page 77 for more information.

- Suspend the Mirror Replication Agent instance to stop replication immediately. See “Suspending the Mirror Replication Agent” on page 78 for more information.

Quiescing the Mirror Replication Agent

Quiescing the Mirror Replication Agent instance stops replication gracefully:

- The Log Reader component stops reading operations from the mirror log devices when the current scan is complete. It continues to send change-set data to the Log Transfer Interface component until it finishes processing the last operation scanned from the log.
- The Log Transfer Interface component stops sending LTL commands to the Replication Server as soon as it finishes processing the last change set it receives 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 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 its transition to *Admin* state.

Suspending the Mirror Replication Agent

Suspending the Mirror Replication Agent instance stops replication 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 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
```

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, as shown in the following example:

```
State  Action
------ ----------------------------
ADMIN  Waiting for operator command
(1 row affected)
```

Mirror Replication Agent states

A Mirror Replication Agent instance can be in one of two discrete states:

- *Admin* – allows you to set or change any Mirror Replication Agent configuration parameter. No replication processing occurs when the instance is in *Admin* state.
- *Replicating* – indicates normal replication processing in the Mirror Replication Agent instance—scanning the transaction log, processing log records, and sending LTL commands to the Replication Server. When the state appears as *Replicating (Waiting at end of log)*, the instance is idle and waiting for transactions to be logged, ready to continue replication processing when new transactions appear in the log.

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 transactions (for example, a network error on the Replication Server connection), or
- Operator intervention (for example, invoking a command that changes the state).

From the time a state-changing event occurs until the moment the Mirror Replication Agent instance is actually in the new state, the instance is in *state transition*. During state transition, some Mirror Replication Agent commands (such as shutdown) 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 mra utility -admin option.
- The Mirror Replication Agent quiesce or suspend command is invoked.
- An unrecoverable replication error occurs while 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 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 commands that request 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 its connection to the Replication Server cannot be restored, the Mirror Replication Agent instance goes to *Admin* state.

To go from *Admin* state to *Replicating* state, use the Mirror Replication Agent resume command. See “Starting replication in the Mirror Replication Agent” on page 75 for more information.

Replicating state

In *Replicating* state, the Mirror Replication Agent instance maintains a connection to the Replication Server, and its Log Reader component scans the 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 reserve the logscan context

and update the secondary truncation point. 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.

Note There may be no replication throughput (that is, reading transactions from the log and sending LTL commands to the Replication Server) when the Mirror Replication Agent instance is in *Replicating* state. In that event, the Mirror Replication Agent state may appear as *Replicating (Waiting at end of log)*.

To go from *Replicating* state to *Admin* state, you can use either the quiesce or suspend command. See "Stopping replication in the Mirror Replication Agent" on page 76 for more information.

Getting Mirror Replication Agent statistics

The Log Reader and Log Transfer Interface components record information about their performance whenever the Mirror Replication Agent instance is in *Replicating* state. You can use this information when tuning Mirror Replication Agent performance, or troubleshooting problems.

To get statistics on current 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, the statistics counters are reset automatically.

See Chapter 4, "Mirror Replication Agent Command Reference," for more information about the ra_statistics command and Mirror Replication Agent statistics.

Testing network connectivity

Mirror Replication Agent provides a simple means of testing its 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 the connection specifications (server name, port number, user login, and password) recorded in the Mirror Replication Agent configuration parameters are not correct, the test_connection command returns a failure message.

See “Setting up Mirror Replication Agent connectivity” on page 34 for information about setting up connection configuration parameters.

Note You can test a specific Mirror Replication Agent connection (either the primary data server, or the Replication Server and its RSSD) by specifying the connection you want to test.

See Chapter 4, “Mirror Replication Agent Command Reference,” for more information about the test_connection command.

Maintaining the Replication Agent System Database

Mirror Replication Agent uses an embedded database, managed by Adaptive Server Anywhere, for its Replication Agent System Database (RASD).

There are five tasks you can perform to maintain the RASD:

- “Updating the RASD” on page 85
- “Updating the log device repository” on page 86
- “Backing up the RASD” on page 89
- “Restoring the RASD” on page 89
- “Truncating the RASD” on page 91

See the following section, “RASD overview,” for more general information about the RASD.

RASD overview

Each Mirror Replication Agent instance depends on the information in its Replication Agent System Database (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 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 system, you must initialize the Mirror Replication Agent instance after the instance is created and its connection configuration parameters are set. See Chapter 2, "Setup and Configuration," for more information.

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 system, and it has several prerequisites. See "Setting up the Mirror Activator system" on page 43 for more information about these tasks, and initializing the Mirror Replication Agent.

After the Mirror Replication Agent is initialized, the RASD cannot be updated without completely rebuilding the RASD (by re-initializing the Mirror Replication Agent).

During normal Mirror Activator system operation, the Mirror Replication Agent maintains its RASD automatically, without manual intervention.

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, Mirror Replication Agent updates its RASD automatically.

There are, however, some DDL commands and system procedures that cannot be replicated, and the Mirror Replication Agent does *not* update its RASD to reflect the schema changes produced by those commands and procedures.

Examples of DDL commands and system procedures that the Mirror Replication Agent cannot replicate include:

- select into
- update statistics
- sp_configure
- sp_dboption

If a DDL command or system procedure produces a change in the primary database schema, and the Mirror Replication Agent cannot replicate 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. See “Updating the RASD” on page 85 for more information.

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.

Mirror Replication Agent determines which version of each object to use by comparing the object’s version string with the LTM Locator value. If the LTM Locator value is greater than or equal to the value of the current version, the current object is used. If the LTM Locator value is less than the value of the current version, a previous version must be used.

Without periodic truncation, the size of the RASD can grow indefinitely, as more and more versions of objects are added. See "Truncating the RASD" on page 91 for more information.

Updating the RASD

Once 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 re-initialized and quiesced. The following procedure includes both of those tasks.

❖ **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 quiesce the Mirror Replication Agent instance:

```
quiesce
```

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 initialize the primary database:

```
pdb_init
```

Warning! Do *not* use the move_truncpt option when you re-initialize a primary database that was previously initialized by the Mirror Replication Agent.

If you do not preserve the existing secondary truncation point in the primary database, you will have to re-materialize the standby database before you can resume replication.

5 After you re-initialize the primary database, use the following command to quiesce the primary database:

```
pdb_quiesce hold
```

6 Use the following command to re-initialize the Mirror Replication Agent and force it to update its RASD:

```
ra_init force
```

7 After the Mirror Replication Agent is initialized, use the following command to release the quiesce hold on the primary database:

```
pdb_quiesce release
```

8 Use the following command to resume replication in the Mirror Replication Agent:

```
resume
```

9 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 6, “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:

- Deletes the entire log device repository in the RASD
- Queries the primary database for information about all of the log devices
- Re-populates its log device repository in the RASD with current information about the log devices

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 modify the path returned by the primary database and recorded in the RASD.

When you invoke ra_updatedevices, any log device path that you modified previously with ra_devicepath is overwritten with the current path specified in the primary database.

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 quiesce the Mirror Replication Agent instance:

```
quiesce
```

Note If the log device changes at the primary database are urgent, you can use suspend, instead of quiesce, to stop replication in the Mirror Replication Agent immediately.

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 modify the default path for a log device (that is, the device path returned by the primary data server), use ra_devicepath.

Use the following command to change a default log device path:

```
ra_devicepath device, dev_path
```

where:

- *device* is the device name or device 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 modify.

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 *after* you truncate the primary database log.

Mirror Replication Agent places RASD backup files in the directory identified by the rasd_backup_dir configuration parameter. See Chapter 5, "Mirror Replication Agent Configuration Parameters," for more information about the rasd_backup_dir parameter.

You need not 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 to 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.

Mirror Replication Agent retrieves the RASD backup files from the directory identified by the rasd_backup_dir configuration parameter. See Chapter 5,

"Mirror Replication Agent Configuration Parameters," for more information about the the rasd_backup_dir parameter.

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 quiesce the Mirror Replication Agent instance:

```
quiesce
```

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 completes successfully, the Mirror Replication Agent returns a message to confirm that the RASD restore was successful.

If the Mirror Replication Agent could not find the directory identified in the rasd_backup_dir parameter, or if it could not 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 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 6, "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 You should back up the RASD using rasd_backup *before* you truncate it. See "Backing up the RASD" on page 89 for more information.

The RASD stores definitions for two types of database objects:

- Articles – tables and stored procedures 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 need not 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.

Configuring Mirror Replication Agent

To set or change a Mirror Replication Agent configuration parameter, use the ra_config command.

Because 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.

See Chapter 4, “Mirror Replication Agent Command Reference,” for more information about the ra_config command.

See Chapter 5, “Mirror Replication Agent Configuration Parameters,” for more information about the configuration file and configuration parameters.

CHAPTER 4

Mirror Replication Agent Command Reference

This chapter describes all of the Mirror Replication Agent commands.

Table 4-1: Mirror Replication Agent commands

Command name	Description	Page
log_system_name	Returns the path to the Mirror Replication Agent system log file.	96
pdb_capabilities	Returns a list of Mirror Replication Agent capabilities.	97
pdb_date	Returns the current date and time from the primary data server.	97
pdb_execute_sql	Executes a SQL statement in the current database.	98
pdb_gen_id	Returns the current database generation ID; updates the database generation ID.	99
pdb_get_columns	Returns all the columns in the specified table.	100
pdb_get_databases	Returns all the databases in the primary data server.	102
pdb_get_primary_keys	Returns all the primary key columns in the specified table.	102
pdb_get_procedure_parms	Returns all the input parameters for the specified procedure.	103
pdb_get_procedures	Returns all the procedures in the specified database.	105
pdb_get_sql_database	Returns the name of the database used for SQL statement execution.	106
pdb_get_tables	Returns all the tables in the specified database.	107
pdb_init	Initializes the primary database for Mirror Replication Agent initialization.	108

Command name	Description	Page
pdb_quiesce	Quiesces the primary database for Mirror Replication Agent initialization.	110
pdb_set_sql_database	Specifies the database to be used for SQL statement execution.	112
pdb_version	Returns the type and version of the primary data server.	113
quiesce	Stops current Log Reader activity, processes data in internal queues, drops connections, and puts Mirror Replication Agent in *Admin* state.	113
ra_config	Returns help information for configuration parameter(s); sets the value of a configuration parameter.	114
ra_date	Returns the current date and time from the Mirror Replication Agent server.	116
ra_devicepath	Changes the path to a log device.	116
ra_help	Returns help information for Mirror Replication Agent command(s).	117
ra_helparticle	Returns information about primary database articles from the RASD.	118
ra_helpdb	Returns information about the primary database from the RASD.	120
ra_helpdevice	Returns information about log devices from the RASD.	120
ra_helpfield	Returns information about fields in articles from the RASD.	122
ra_helplocator	Returns LTM Locator field values.	123
ra_helpuser	Returns information about primary database users from the RASD.	124
ra_init	Initializes the Mirror Replication Agent instance.	126
ra_locator	Returns the current LTM Locator stored by the Mirror Replication Agent; retrieves a new LTM Locator from Replication Server.	128
ra_maintid	Returns the Maintenance User for the primary database connection.	129
ra_set_login	Sets the Mirror Replication Agent admin user login and password.	130

Command name	Description	Page
ra_statistics	Returns statistics for either a specified Mirror Replication Agent component, or all components; resets statistics for all components.	131
ra_status	Returns the current Mirror Replication Agent state.	136
ra_truncatearticles	Truncates older versions of primary database articles in the RASD.	137
ra_truncateusers	Truncates older versions of primary database users in the RASD.	138
ra_updatedevices	Updates the log device repository in the RASD.	139
ra_version	Returns the Mirror Replication Agent version.	140
ra_version_all	Returns Mirror Replication Agent, primary data server, Replication Server, and communications driver versions.	141
rasd_backup	Backs up the RASD.	142
rasd_restore	Restores the RASD.	142
resume	Starts replication for the current primary log and puts Mirror Replication Agent in *Replicating* state.	143
shutdown	Shuts down Mirror Replication Agent.	145
suspend	Immediately stops all Log Reader activity, drops connections, and puts Mirror Replication Agent in *Admin* state.	145
test_connection	Tests Mirror Replication Agent connections.	146
trace	Returns current trace flag settings; changes a specified trace flag.	148

The remaining sections in this chapter describe each Mirror Replication Agent command in detail.

log_system_name

Description — Returns the full path of the Mirror Replication Agent instance log file.

Syntax — log_system_name

Usage

- When you create a Mirror Replication Agent instance, a log directory is created automatically as part of the instance directory structure. The default value of the log_directory parameter points to that directory.
- The default path of the Mirror Replication Agent system log directory is:

  ```
  C:\%SYBASE%\MRA-12_6\inst_name\log\
  ```

  where:

  - *%SYBASE%* is the Mirror Replication Agent installation directory.
  - *inst_name* is the name of the Mirror Replication Agent instance.
- If you specify a valid directory path as the value of the log_directory parameter, the Mirror Replication Agent instance places its system log file in the directory you specify.

  If you change the value of the log_directory parameter with the ra_config command, the new value is recorded in the configuration file immediately, but you must shut down and restart the Mirror Replication Agent instance to make the new value take effect.

  See Chapter 5, “Mirror Replication Agent Configuration Parameters,” for more information.
- The system log file contains output from the following trace points:
  - Trace points identified as SYSTEM trace flags
  - Additional trace points enabled by using the trace command

  A list of all trace flags is provided in the description of the trace command beginning on page 148.
- The log_system_name command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also — ra_config, trace

pdb_capabilities

Description — Returns a list of Mirror Replication Agent capabilities, which is used by the Replication Server Manager.

Syntax — pdb_capabilities

Usage

- When pdb_capabilities is invoked, it returns a list of the capabilities of the Mirror Replication Agent instance.
- The purpose of the pdb_capabilities command is to support the Replication Server Manager.
- The pdb_capabilities command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

pdb_date

Description — Returns the current date and time from the primary data server.

Syntax — pdb_date

Usage

- When pdb_date is invoked, it returns the current date and time from the primary data server in the form of a Sybase datetime datatype, as follows:

```
Current PDB Date
 -----------------------------
     Jan 30 2004 12:09:47.310
(1 row affected)
```

- The pdb_date command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also — ra_date

pdb_execute_sql

Description — Executes a SQL statement in the current database at the primary data server.

Syntax — pdb_execute_sql *statement*

Parameters

statement

A SQL statement to be executed in the current database at the primary data server.

Usage

- The pdb_execute_sql command allows simple SQL queries to be executed in a database.
- The SQL statement specified in the pdb_execute_sql command must be a single SQL command enclosed in double quotes. For example:

  ```
  pdb_execute_sql "select * from Authors"
  ```

  No command terminator is required. No syntax or other validation is performed; the string is passed directly to the current database.
- When pdb_execute_sql is invoked, it executes the specified SQL statement against the current database.
- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.
- To set or change the current database, use the pdb_set_sql_database command.

Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_execute_sql command executes the SQL query against the primary database to which the Mirror Replication Agent instance is connected.

- Any results returned from execution of the SQL statement are passed to the Mirror Replication Agent client, by way of the Mirror Replication Agent administration port.
- To find the name of the current database, use the pdb_get_sql_database command.

Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_sql_database command returns the name of the default current database (the primary database to which the Mirror Replication Agent instance is connected).

- The pdb_execute_sql command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_get_sql_database, pdb_set_sql_database

pdb_gen_id

Description Returns the current value of the database generation ID, or updates the value of the database generation ID.

Syntax pdb_gen_id [*number*]

Parameters *number*

The value of the new database generation ID to be used when the database generation ID is updated.

Examples **Example 1**

```
pdb_gen_id
```

This command returns the current value of the database generation ID.

Example 2

```
pdb_gen_id 10
```

This command updates the database generation ID to the value 10.

Usage

- When pdb_gen_id is invoked with no option, it returns the current value of the database generation ID stored in the RASD. This is the value of the database generation ID at the primary database.
- When pdb_gen_id is invoked with the *number* option, it updates the value of the database generation ID in the RASD, and at the primary database. Changing the database generation ID takes effect immediately.
- The database generation ID is the first 2 bytes of the origin queue ID. The database generation ID is used by Replication Server to support recovery operations, which may require the Mirror Replication Agent to re-send transactions.

  During recovery, if the Mirror Replication Agent must re-send operations that the Replication Server has already processed, you can change the database generation ID to prevent the Replication Server from recognizing the operations as already processed.

For more information about the origin queue ID, see ra_helplocator on page 123.

- If the system database repository does not exist in the RASD (because the Mirror Replication Agent has not been initialized), the pdb_gen_id command returns an error.
- The pdb_gen_id command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_helplocator, ra_locator

pdb_get_columns

Description Returns a list of columns in tables in the current database at the primary data server.

Syntax pdb_get_columns [*ownername*, *tablename*[, *colname*]]

Parameters

ownername
The user name of the owner of the table specified in the *tablename* option. This option can be delimited with quote characters to specify character case.

tablename
The name of the table in the current database for which information is returned. This option can be delimited with quote characters to specify character case.

colname
The name of the column for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

Example 1

```
pdb_get_columns
```

This command returns a list of all of the columns in all of the user tables in the current database.

Example 2

```
pdb_get_columns bob, authors
```

This command returns a list of all of the columns in the table authors, owned by the user “bob” in the current database.

Example 3

```
pdb_get_columns bob, authors, au_fname
```

This command returns information about the column au_fname in the table authors, owned by the user "bob" in the current database.

Usage

- When pdb_get_columns is invoked with no option, it returns a result set that lists all of the columns in all of the user tables in the current database.
- When pdb_get_columns is invoked with the *ownername* and *tablename* options, it returns a result set that lists all of the columns in the specified table with the specified owner in the current database.
- When pdb_get_columns is invoked with the *ownername*, *tablename*, and *colname* options, it returns a result set with information about the specified column in the specified table with the specified owner in the current database.
- The pdb_get_columns command accepts the % wildcard character in the *ownername*, *tablename*, and *colname* options.
- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.

Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_columns command returns results from the primary database to which the Mirror Replication Agent instance is connected.

- To set or change the current database, use the pdb_set_sql_database command.
- To find the name of the current database, use the pdb_get_sql_database command.
- The pdb_get_columns command returns 0 rows if the specified table (with the specified owner) does not exist in the current database or if the specified column does not exist in the specified table.
- The pdb_get_columns command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

pdb_get_databases, pdb_get_primary_keys, pdb_get_procedure_parms, pdb_get_procedures, pdb_get_tables

pdb_get_databases

Description — Returns a list of all databases in the primary data server.

Syntax — pdb_get_databases

Usage

- When pdb_get_databases is invoked, it returns a result set that lists all of the databases in the primary data server.
- The pdb_get_databases command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also — pdb_get_columns, pdb_get_primary_keys, pdb_get_procedure_parms, pdb_get_procedures, pdb_get_tables

pdb_get_primary_keys

Description — Returns a list of primary key columns in a specified table in the current database at the primary data server.

Syntax — pdb_get_primary_keys *ownername*, *tablename*

Parameters

ownername
The user name of the owner of the table specified in *tablename*. This option can be delimited with quote characters to specify character case.

tablename
The name of the table in the current database for which primary key column information is returned. This option can be delimited with quote characters to specify character case.

Usage

- When pdb_get_primary_keys is invoked, it returns a result set that lists all of the columns that are defined as primary keys in the specified table with the specified owner in the current database.
- The pdb_get_primary_keys command accepts the % wildcard character in the *ownername* option, but not in the *tablename* option.
- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The current default database is identified by the pds_database_name configuration parameter.

Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_primary_keys command returns results from the primary database to which the Mirror Replication Agent instance is connected.

- To set or change the current database, use the pdb_set_sql_database command.
- To find the name of the current database, use the pdb_get_sql_database command.
- The pdb_get_primary_keys command returns 0 rows if the specified table with the specified owner does not exist in the current database.
- The pdb_get_primary_keys command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_get_columns, pdb_get_databases, pdb_get_procedure_parms, pdb_get_procedures, pdb_get_tables

pdb_get_procedure_parms

Description Returns a list of input parameters for procedures in the current database at the primary data server.

Syntax pdb_get_procedure_parms [*ownername*, *procname* [, *paramname*]]

Parameters

ownername
The user name of the owner of the procedure specified in *procname*. This option can be delimited with quote characters to specify character case.

procname
The name of the procedure in the current database for which information is returned. This option can be delimited with quote characters to specify character case.

paramname
The name of the input parameter for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

Example 1

```
pdb_get_procedure_parms
```

This command returns a list of all of the input parameters for all of the procedures in the current database.

Example 2

```
pdb_get_columns bob, sp_foo
```

This command returns a list of all of the input parameters for the procedure named sp_foo, owned by the user "bob" in the current database.

Example 3

```
pdb_get_columns bob, sp_foo, foo_count
```

This command returns information about the input parameter foo_count for the procedure sp_foo, owned by the user "bob" in the current database.

Usage

- When pdb_get_procedure_parms is invoked with no option, it returns a result set that lists all of the input parameters for all the procedures in the current database.
- When pdb_get_procedure_parms is invoked with the *ownername* and *procname* options, it returns a result set that lists all of the input parameters for the specified procedure with the specified owner in the current database.
- When pdb_get_procedure_parms is invoked with the *ownername*, *procname*, and *paramname* options, it returns a result set with information about the specified input parameter for the specified procedure with the specified owner in the current database.
- The pdb_get_procedure_parms command accepts the % wildcard character in both the *ownername* and *procname* options.
- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.

  Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_procedure_parms command returns results from the primary database to which the Mirror Replication Agent instance is connected.

- To set or change the current database, use the pdb_set_sql_database command.

- To find the name of the current database, use the pdb_get_sql_database command.
- The pdb_get_procedure_parms command returns 0 rows if the specified procedure (with the specified owner) does not exist in the current database.
- The pdb_get_procedure_parms command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

pdb_get_columns, pdb_get_databases, pdb_get_primary_keys, pdb_get_procedures, pdb_get_tables

pdb_get_procedures

Description

Returns a list of procedures in the current database at the primary data server.

Syntax

pdb_get_procedures [*ownername*, *procname*]

Parameters

ownername

The user name of the owner of the procedure specified in *procname*. This option can be delimited with quote characters to specify character case.

procname

The name of the procedure in the current database for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

Example 1

```
pdb_get_procedures
```

This command returns a list of all of the procedures in the current database.

Example 2

```
pdb_get_columns bob, sp_foo
```

This command returns information about the procedure named sp_foo, owned by the user "bob" in the current database.

Usage

- When pdb_get_procedures is invoked with no option, it returns a result set that lists all of the procedures in the current database.
- When pdb_get_procedures is invoked with the *ownername* and *procname* options, it returns a result set with information about the specified procedure with the specified owner in the current database.

- The pdb_get_procedures command accepts the % wildcard character in both the *ownername* and *procname* options.
- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.

Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_procedures command returns results from the primary database to which the Mirror Replication Agent instance is connected.

- To set or change the current database, use the pdb_set_sql_database command.
- To find the name of the current database, use the pdb_get_sql_database command.
- The pdb_get_procedures command returns 0 rows if the specified procedure (with the specified owner) does not exist in the current database.
- The pdb_get_procedures command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_get_columns, pdb_get_databases, pdb_get_primary_keys, pdb_get_procedure_parms, pdb_get_tables

pdb_get_sql_database

Description Returns the name of the current database, if any.

Syntax pdb_get_sql_database

Usage

- When pdb_get_sql_database is invoked, it returns the name of the current database.
- If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_sql_database command returns the default current database.
- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.

- To set or change the current database, use the pdb_set_sql_database command.
- The pdb_get_sql_database command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_execute_sql, pdb_set_sql_database

pdb_get_tables

Description Returns a list of user tables in the current database at the primary data server.

Syntax pdb_get_tables [*ownername*, *tablename*]

Parameters

ownername
The user name of the owner of the table specified in *tablename*. This option can be delimited with quote characters to specify character case.

tablename
The name of the table in the current database for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

Example 1

```
pdb_get_tables
```

This command returns a list of all of the user tables in the current database.

Example 2

```
pdb_get_columns bob, authors
```

This command returns information about the table authors, owned by the user "bob" in the current database.

Usage

- When pdb_get_tables is invoked with no option, it returns a result set that lists all of the user tables in the current database.
- When pdb_get_tables is invoked with the *ownername* and *tablename* options, it returns a result set with information about the specified table with the specified owner in the current database.
- The pdb_get_tables command accepts the % wildcard character in the both the *ownername* and *tablename* options.

- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.

  Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_tables command returns results from the primary database to which the Mirror Replication Agent instance is connected.

- To set or change the current database, use the pdb_set_sql_database command.
- To find the name of the current database, use the pdb_get_sql_database command.
- The pdb_get_tables command returns 0 rows if the specified table (with the specified owner) does not exist in the current database.
- The pdb_get_tables command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

pdb_get_columns, pdb_get_databases, pdb_get_primary_keys, pdb_get_procedure_parms, pdb_get_procedures

pdb_init

Description

Initializes the primary database in preparation for Mirror Replication Agent initialization.

Syntax

pdb_init [move_truncpt]

Parameters

move_truncpt

The optional keyword that sets the secondary truncation point to the end of the primary database transaction log.

Examples

Example 1

```
pdb_init
```

This command initializes the primary database, and if the secondary truncation point is not already set, it sets the secondary truncation point to the end of the transaction log.

Example 2

```
pdb_init move_truncpt
```

This command initializes the primary database and sets the secondary truncation point to the end of the log.

Usage

- The primary database must *not* be quiesced when you use pdb_init to initialize the primary database.
- After you initialize the primary dabatase with pdb_init, you must *not* allow any DDL operations in the primary database before it is quiesced for the Mirror Replication Agent initialization.
- When pdb_init is invoked with no option, Mirror Replication Agent does the following:
  - Verifies that:
    - The Mirror Replication Agent instance is in *Admin* state.
    - The primary database is not quiesced.
    - The RepAgent thread in the primary database is disabled.
  - Sets the secondary truncation point, only if it is *not* already set.
  - Marks all user tables in the primary database for replication in a warm standby application. This is equivalent to executing sp_reptostandby in the primary database.
- When pdb_init is invoked with the move_truncpt keyword, Mirror Replication Agent does the same things as described in the previous item, except that it sets the secondary truncation point to the end of the log.
- The pdb_init command returns an error if:
  - The primary database is quiesced.
  - The Mirror Replication Agent instance is in *Replicating* state.
- To replicate stored procedures in the primary database, you must mark the procedures for replication using the sp_setrepproc, "function" procedure in the primary database for each procedure that you want to replicate.

  If you mark a stored procedure using the sp_setreplicate or sp_setrepproc, "table" procedure, only the resulting data-changing operations on tables are replicated, not the procedure call itself.

Note Mirror Replication Agent 12.6 supports only request function delivery; it does *not* support applied function delivery.

- The pdb_init command is valid only when the Mirror Replication Agent is in *Admin* state.

See also pdb_quiesce, ra_init

pdb_quiesce

Description Quiesces the primary database in preparation for Mirror Replication Agent initialization.

Syntax pdb_quiesce hold [, for_dump [, *manifest_file*]]

pdb_quiesce release

Parameters hold
The keyword for the quiesce database hold action.

release
The keyword for the quiesce database release action.

for_dump
The optional keyword for the quiesce database hold for external dump action.

manifest_file
The name of the manifest file created by quiesce database.

Examples **Example 1**

```
pdb_quiesce hold
```

This command tells Mirror Replication Agent to invoke the quiesce database hold command for the primary database.

Example 2

```
pdb_quiesce hold, for_dump
```

This command tells Mirror Replication Agent to invoke the quiesce database hold for external dump command for the primary database.

Example 3

```
pdb_quiesce hold, for_dump, "%TEMP%\mypubs_file"
```

This command tells Mirror Replication Agent to invoke the quiesce database hold for external dump command, with the extension for creating for a manifest file.

Example 4

```
pdb_quiesce release
```

This command tells Mirror Replication Agent to invoke the quiesce database release command for the primary database.

Usage

- You *cannot* quiesce the primary database using pdb_quiesce, unless the Mirror Replication Agent user login in the primary database has a System Administrator role.
- When pdb_quiesce is invoked, Mirror Replication Agent invokes quiesce database for the primary database, using a generated tag name with the following format:

  ```
  ra_pdb_tag
  ```

  where *pdb* is the name of the primary database.

  Note If the length of the generated tag name exceeds the primary data server limit for identifier names, the tag name is truncated to the maximum allowed size.

- See the Adaptive Server Enterprise *Reference Manual* and *System Administration Guide* for more information about the quiesce database command.
- You must quiesce the primary database *before* you initialize the Mirror Replication Agent with the ra_init command.
- After you initialize the primary dabatase with pdb_init, you must *not* allow any DDL operations in the primary database before it is quiesced for the Mirror Replication Agent initialization.
- The pdb_quiesce command returns an error if the Mirror Replication Agent user login in the primary database does *not* have a System Administrator role.
- The pdb_quiesce command is valid when the Mirror Replication Agent is in either *Admin* or *Replicating* state.

See also pdb_init, ra_init

pdb_set_sql_database

Description Sets the current database to be used for SQL statement execution.

Syntax pdb_set_sql_database *database*

Parameters *database*

The name of the database in the primary data server against which SQL statements can be executed. This parameter can be delimited with quote characters to specify character case.

Usage

- When pdb_set_sql_database is invoked, it sets the database in which SQL statements can be executed.

  Note Some data servers and communication drivers do not allow you to change the current database.

- The database name you specify with the pdb_set_sql_database command is not validated. If you specify an invalid database name, no error is returned until the pdb_execute_sql command is invoked.
- To determine which database is currently specified for SQL statement execution, use the pdb_get_sql_database command.

  Note If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_get_sql_database command returns the default current database.

- The default current database is the primary database to which the Mirror Replication Agent instance is connected. The default current database is identified by the pds_database_name configuration parameter.
- If the pdb_set_sql_database command has not been invoked to set the current database, the pdb_execute_sql command executes the SQL query against the primary database (the default current database).
- The pdb_set_sql_database command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_execute_sql, pdb_get_sql_database

pdb_version

Description — Returns the type and version of the primary data server and the JDBC driver.

Syntax — pdb_version

Usage

- When pdb_version is invoked, it returns primary data server vendor name and version and the JDBC driver name and version for the primary database.
- If the primary database connection is down or not configured, the pdb_version command returns an error.
- The pdb_version command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also — ra_version, ra_version_all

quiesce

Description — Stops all Mirror Replication Agent processing in *Replicating* state, and puts the Mirror Replication Agent instance in *Admin* state.

Syntax — quiesce

Usage

- When the quiesce command is invoked, it stops all current replication processing:
  - The Log Reader component stops reading operations from the mirror log devices as soon as the current scan is complete. It continues to send change-set data to the Log Transfer Interface component until it finishes processing the last operation scanned from the log.
  - The Log Transfer Interface component stops sending LTL commands to the Replication Server as soon as it finishes processing the last change set it receives 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.

- If the Mirror Replication Agent internal queues are full when the quiesce command is invoked, the quiesce processing may take a while to complete, and there may be a delay before the Mirror Replication Agent instance completes its transition to *Admin* state.
- If the Mirror Replication Agent instance is in *Admin* state, the quiesce command returns an error.
- The quiesce command is valid only when the Mirror Replication Agent instance is in *Replicating* state.

Note The action of the suspend command is similar to that of the quiesce command, except that the suspend command stops Mirror Replication Agent processing immediately and flushes all data in the internal queues.

See also pdb_quiesce, ra_status, resume, shutdown, suspend

ra_config

Description Returns help information for Mirror Replication Agent configuration parameters, or sets the value of a specified configuration parameter.

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

Parameters

param
The name of a Mirror Replication Agent configuration parameter.

value
The value to be assigned to the configuration parameter specified in the *param* option. You can use the keyword default to set the specified parameter to its default value.

Examples

Example 1

```
ra_config use_rssd
```

This command returns the current value of the use_rssd configuration parameter.

Example 2

```
ra_config scan_sleep_max, 60
```

This command changes the value of the scan_sleep_max parameter to 60.

Usage

- If ra_config is invoked with no option, it returns a list of all Mirror Replication Agent configuration parameters. The following information is returned for each configuration parameter:
  - Parameter name
  - Parameter type – the datatype of the parameter's value (for example, string, numeric, or Boolean).
  - Current value – the value of the parameter in effect at the time ra_config is invoked.
  - Pending value – if different from the current value, the value to which the parameter was set by a previous invocation of the ra_config command, but which has not yet taken effect.
  - Default value – the value of the parameter when the Mirror Replication Agent instance configuration file is created.
  - Legal values – the values that are allowed for the parameter, for example, a range of numbers or a list of specific strings.
  - Category – this refers to the Mirror Replication Agent component affected by the value of the parameter.
  - Restart – this refers to parameters that require the Mirror Replication Agent instance to be shut down and restarted before a change in value takes effect.
- If ra_config is invoked with the *param* option, it returns information only for the specified configuration parameter.
- If ra_config is invoked with the *param* and *value* options, it changes the setting of the specified configuration parameter to the value specified in the *value* option.
- You can use the keyword default in place of the *value* option to reset a configuration parameter to its default value. For example:

  ```
  ra_config use_rssd, default
  ```

- When ra_config is invoked with either no option, or only the *param* option, the command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.
- If ra_config is invoked when the Mirror Replication Agent instance is in *Replicating* state, with the *param* and *value* options for a parameter that can be changed only in *Admin* state, it returns an error.

- When ra_config is invoked with the *param* and *value* options, the command is always valid when the Mirror Replication Agent instance is in *Admin* state.
- See Chapter 5, "Mirror Replication Agent Configuration Parameters," for more information.

See also ra_help, ra_set_login

ra_date

Description Returns the current date and time from the Mirror Replication Agent server.

Syntax ra_date

Usage

- When ra_date is invoked, it returns the current date and time from the Mirror Replication Agent server in the form of a Sybase datetime datatype, as follows:

```
Current RA Date
 ------------------------------
     Apr 30 2004 12:09:47.310
(1 row affected)
```

- The ra_date command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_date, ra_config

ra_devicepath

Description Changes the path to a log device recorded in the log device repository.

Syntax ra_devicepath *device*, *dev_path*

Parameters

device
The name or device ID of a primary database log device.

dev_path
The path to the primary database log device specified in the *device* option.

Examples

Example 1

```
ra_devicepath logdev1, c:\sybase\devices\ase1\log1.dat
```

This command specifies the path to the log device named logdev1 as:

```
c:\sybase\devices\ase1\log1.dat
```

Example 2

```
ra_devicepath 3, c:\sybase\devices\ase1\log1.dat
```

This command specifies the path to the log device ID 3 as:

```
c:\sybase\devices\ase1\log1.dat
```

Usage

- When ra_devicepath is invoked, Mirror Replication Agent records the specified path for the specified primary database log device in the RASD log device repository.
- To get information about log devices stored in the log device repository, use the ra_helpdevice command.
- If you invoke ra_updatedevices after you set a device path using ra_devicepath, you must use ra_devicepath again to re-set the path if you need to alter the default path for a log device. The default device path is the device path returned by the primary data server.
- If you invoke ra_devicepath when the Mirror Replication Agent instance is in *Replicating* state, it returns an error.
- The ra_devicepath command is valid only when the Mirror Replication Agent instance is in *Admin* state.

See also

ra_helpdevice, ra_init, ra_updatedevices

ra_help

Description

Returns help information for Mirror Replication Agent commands.

Syntax

ra_help [*command*]

Parameters

command

The name of a Mirror Replication Agent command for which you want to view help information.

Examples

Example 1

```
ra_help
```

This command returns help for all Mirror Replication Agent commands.

Example 2

```
ra_help pdb_gen_id
```

This command returns help for the pdb_gen_id command.

Usage

- If ra_help is invoked with no option, it returns help information for all Mirror Replication Agent commands.
- If ra_help is invoked with the *command* option, it returns help information only for the specified command.
- The ra_help command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

ra_config, ra_helparticle, ra_helpdb, ra_helpdevice, ra_helpfield, ra_helplocator, ra_helpuser

ra_helparticle

Description

Returns information about articles from the RASD.

Syntax

ra_helparticle [*article*, [*version*]]

Parameters

article

The name or object ID of an article (table or procedure) in the primary database. Article names can be qualified with an owner name in the following form:

```
owner.article
```

Owner qualification of article names is optional.

version

A hexadecimal locator value that identifies the version of the article specified in the *article* option.

Examples

Example 1

```
ra_helparticle
```

This command returns information about all versions of all articles in the RASD.

Example 2

```
ra_helparticle table1
```

This command returns information about the current version of the article named table1 in the RASD.

Example 3

```
ra_helparticle table1,
00000000000210a40000333400070000333400069994000d413c50000000000
```

This command returns information about version 00000000000210a40000333400070000333400069994000d413c50000000000 of the article named table1 in the RASD.

Usage

- The ra_helparticle command returns the following information for articles (tables and procedures):
  - Article object ID
  - Primary database name
  - Article owner name or alias
  - Article name
  - Article type (table or procedure)
  - Article status (Current, Archived, or Dropped)
  - Article version number

  All information except the article type, article status, and article version number are the values returned by the primary database when the Mirror Replication Agent is initialized with the ra_init command.
- If ra_helparticle is invoked with no option, it returns information for all versions of all articles (tables and procedures) in the RASD.
- If ra_helparticle is invoked with the *article* option, it returns information only for the current version of the specified article in the RASD.
- If ra_helparticle is invoked with the *article* and *version* options, it returns information only for the specified version of the specified article in the RASD.
- The ra_helparticle command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

ra_helpdb, ra_helpfield, ra_helplocator, ra_helpuser

ra_helpdb

Description — Returns information about the primary database from the RASD.

Syntax — ra_helpdb

Usage

- When ra_helpdb is invoked, it returns the following information about the primary database:
  - Database object ID
  - Database name

  The database ID and database name are the values returned by the primary database when the Mirror Replication Agent is initialized with the ra_init command.
- The ra_helpdb command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also — ra_devicepath, ra_helparticle, ra_helpdevice, ra_helpfield, ra_helplocator, ra_helpuser, ra_init, ra_updatedevices

ra_helpdevice

Description — Returns information about primary database log devices from the RASD log device repository.

Syntax — ra_helpdevice [*device*]

Parameters

device
The device name or device ID of the primary database log device.

Examples

Example 1

```
ra_helpdevice
```

This command returns information about all primary database log devices recorded in the log device repository.

Example 2

```
ra_helpdevice logdev1
```

This command returns information about the primary database log device named logdev1 in the log device repository.

Usage

- The ra_helpdevice command returns the following information for each primary database log device:

- Log device ID
- Primary database name
- Log device name
- Log device path

The primary database name, log device ID, and log device name are the values returned by the primary database when the Mirror Replication Agent is initialized with the ra_init command, or when the log device repository is updated with the ra_updatedevices command.

The log device path returned by ra_helpdevice is the current specification of that path in the RASD. The default path is the path returned by the primary database, when the Mirror Replication Agent is initialized with the ra_init command, or when the log device repository is updated with the ra_updatedevices command.

Each log device path can be altered by the ra_devicepath command, to "re-map" the path to point to the corresponding mirror log device at the standby site. Mirror Replication Agent uses the log device paths recorded in its RASD to find the mirror log devices, from which it reads the primary database transaction log.

Note The ra_helpdevice command does *not* return information about software mirror devices created with the Adaptive Server disk mirror command. See the Adaptive Server documentation for more information about software disk mirroring.

- If ra_helpdevice is invoked with no option, it returns information for all primary database log devices in the RASD log device repository.
- If ra_helpdevice is invoked with the *device* option, it returns information only for the specified primary database log device in the RASD log device repository.
- The ra_helpdevice command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_devicepath, ra_helpdb, ra_updatedevices

ra_helpfield

Description

Returns information about fields (columns in tables, or input parameters in stored procedures) from the RASD.

Syntax

ra_helpfield *article* [, *version* [, *field*]]

Parameters

article

The name or object ID of an article (table or procedure) in the primary database. Article names can be qualified with an owner name in the following form:

```
owner.article
```

Owner qualification of article names is optional.

version

A hexadecimal locator value that identifies the version of the specified article.

field

The name or object ID of a field (column or input parameter) in the specified article.

Examples

Example 1

```
ra_helpfield authors
```

This command returns information about all fields in the current version of the article named authors in the RASD.

Example 2

```
ra_helpfield authors,
00000000000210a400003334000700003334000699940000d413c50000000000
```

This command returns information about all fields in version 00000000000210a400003334000700003334000699940000d413c50000000000 of the article named authors in the RASD.

Example 3

```
ra_helpfield authors,
00000000000210a400003334000700003334000699940000d413c50000000000, au_fname
```

This command returns information about the field named au_fname in version 00000000000210a400003334000700003334000699940000d413c50000000000 of the article named authors in the RASD.

Usage

- The ra_helpfield command returns the following information for fields:
  - Field (column or input parameter) object ID
  - Field name
  - Field type ID
  - Field datatype (with precision, length, and scale)
  - Field NULL mode
  - Field IDENTITY status
  - Field primary key status

  All field information items are the values returned by the primary database when the Mirror Replication Agent is initialized with the ra_init command.
- If ra_helpfield is invoked with the *article* option, it returns information for all fields in the current version of the specified article in the RASD.
- If ra_helpfield is invoked with the *article* and *version* options, it returns information for all fields in the specified version of the specified article in the RASD.
- If ra_helpfield is invoked with the *article*, *version*, and *field* options, it returns information for the specified field in the specified version of the specified article in the RASD.
- The ra_helpfield command is valid when the Mirror Replication Agent is in either *Admin* or *Replicating* state.

See also

ra_config, ra_help, ra_helparticle, ra_helpdb, ra_helpdevice, ra_helplocator, ra_helpuser

ra_helplocator

Description

Returns information about fields in the LTM Locator value.

Syntax

ra_helplocator [*locator_value*]

Parameters

locator_value

The hexadecimal string value of the LTM Locator.

Examples

Example 1

```
ra_helplocator
```

This command returns information about fields in the current LTM Locator value.

Example 2

```
ra_helplocator locator_value
```

This command returns information about fields in the specified LTM Locator value.

Usage

- The ra_helplocator command returns the following information about the LTM Locator value:
  - Locator field names
  - Locator field hexadecimal values
  - Locator field decimal values
- If ra_helplocator is invoked with no option, it returns information about fields in the current LTM Locator value.
- If ra_helplocator is invoked with the *locator_value* option, it returns information about fields in the specified LTM Locator value.
- The ra_helplocator command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

ra_config, ra_help, ra_helparticle, ra_helpdb, ra_helpdevice, ra_helpfield, ra_helpuser, ra_locator

ra_helpuser

Description

Returns information about primary database users from the RASD.

Syntax

ra_helpuser [*user* [, *version*]]

Parameters

user

The name or user ID of a user in the primary database.

version

The version number of the database user in the RASD.

Examples

Example 1

```
ra_helpuser
```

This command returns information about all users in the RASD.

Example 2

```
ra_helpuser bob
```

This command returns information about all versions of the database user named “bob” in the RASD.

Example 3

```
ra_helpuser bob,
00000000000210a40000333400070000333400069994000d413c50000000000
```

This command returns information about version 00000000000210a40000333400070000333400069994000d413c50000000000 of the database user named “bob” in the RASD.

Usage

- The ra_helpuser command returns the following information about primary database users:
  - User ID
  - User name
  - User status (Current, Archived, or Dropped)
  - Primary database version (locator value)

  The user ID and user name are the values returned by the primary database when the Mirror Replication Agent is initialized with the ra_init command.
- If ra_helpuser is invoked with no option, it returns information about all users in the RASD.
- If ra_helpuser is invoked with the *user* option, it returns information about the specified user in all versions of the primary database in the RASD.
- If ra_helpuser is invoked with the *user* and *version* options, it returns information about the specified user in the specified version of the primary database in the RASD.
- The ra_helpuser command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

ra_config, ra_help, ra_helparticle, ra_helpdb, ra_helpdevice, ra_helpfield, ra_helplocator

ra_init

Description

Initializes the Mirror Replication Agent instance and populates the system data repository in the RASD.

Note The primary database must be quiesced before you invoke ra_init.

Syntax

ra_init [force]

Parameters

force

The optional keyword that forces the Mirror Replication Agent instance to re-initialize its system data repository in the RASD.

Examples

Example 1

```
ra_init
```

This command initializes the Mirror Replication Agent instance.

Example 2

```
ra_init force
```

This command re-initializes the Mirror Replication Agent instance, after it was initialized previously.

Usage

- Before you invoke ra_init to initialize the Mirror Replication Agent, the primary database must be:
  - Configured to accept a login from the Mirror Replication Agent instance,
  - Initialized with the Mirror Replication Agent pdb_init command, and
  - Quiesced to suspend update activity (marked as "in quiesce").

  If any of these conditions are not met by the primary database, ra_init returns an error.
- After you initialize the primary dabatase with pdb_init, you must *not* allow any DDL operations in the primary database before the database is quiesced for Mirror Replication Agent initialization.
- If ra_init is invoked with no option, Mirror Replication Agent does the following:
  - Queries the primary database to get the information it needs for the system data repository

- Populates the RASD with information returned by the primary database

Note You can invoke ra_init with no option *only* if the Mirror Replication Agent system data repository does *not* exist (that is, you have not previously initialized the Mirror Replication Agent instance).

- If ra_init is invoked with no option, and the Mirror Replication Agent already has data in its RASD (created previously by initializing the instance), it returns an error.
- If the Mirror Replication Agent already has data in its RASD (created previously by initializing the instance), you must invoke ra_init with the force keyword to delete all of the existing data from the RASD, and re-create the entire system data repository.
- If ra_init is invoked with the force keyword, Mirror Replication Agent does the following:
  - Deletes all of the data in its RASD (truncates all of the tables in the database)
  - Queries the primary database to get the information it needs for the system data repository
  - Re-populates the RASD with information returned by the primary database
- If you invoke ra_init when the Mirror Replication Agent instance is in *Replicating* state, it returns an error.
- The ra_init command is valid only when the Mirror Replication Agent instance is in *Admin* state.

See also pdb_init, pdb_quiesce

ra_locator

Description

Returns the current value of the LTM Locator recorded in the RASD, retrieves an LTM Locator value from the Replication Server, or sets the value of the LTM Locator to all zeros.

Syntax

ra_locator [update]

Parameters

update

The optional keyword that tells Mirror Replication Agent to retrieve a new LTM Locator value from the Replication Server.

Examples

Example 1

```
ra_locator
```

This command returns the current value of the LTM Locator, from the RASD, as follows:

```
Locator
 ------------------------------------------------
 0000000052000000000000527FFFFFFFFFFFFFF0022FB3B
 (1 row affected)
```

Example 2

```
ra_locator update
```

This command retrieves a new LTM Locator value from the Replication Server.

Usage

- When ra_locator is invoked with no option, it returns the current value of the LTM Locator stored in the RASD.

  Note The value of the LTM Locator that is maintained by the Mirror Replication Agent is also known as the *origin queue ID*.

- When ra_locator is invoked with the update keyword, it retrieves a new LTM Locator value from the Replication Server, and stores it in the RASD. The LTM Locator update operation is asynchronous and it may take a few seconds to complete.

  Note When the ra_locator command is invoked with the update keyword, the LTM Locator change takes effect *only* if the Mirror Replication Agent instance is in *Replicating* state.

- The LTM Locator contains information that Mirror Replication Agent uses to determine where to start reading the primary database transaction log.

Upon start-up, or recovery from a connection failure, Mirror Replication Agent automatically requests an LTM Locator value from the Replication Server.

- If the value of the LTM Locator returned from the Replication Server is zero, then Mirror Replication Agent uses the LTM Locator value stored in its RASD.
- If the value of the LTM Locator in the RASD is zero, then Mirror Replication Agent starts reading the primary database transaction log from the end of the log.

- If the value of the LTM Locator does not exist in the RASD, the ra_locator command returns a zero locator value.
- The ra_locator command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_gen_id, ra_helplocator

ra_maintid

Description Returns the name of the Maintenance User for the primary database connection.

Syntax ra_maintid

Usage

- Replication Server requires a Maintenance User name for each database connection. The Maintenance User name for a database connection is defined with the Replication Server create connection or alter connection command.

  When the primary database Maintenance User name is changed in the Replication Server (using the alter connection command), Replication Server sends the new Maintenance User name to the Mirror Replication Agent, if the Mirror Replication Agent is in *Replicating* state.

  Each time the Mirror Replication Agent goes into *Replicating* state, it automatically retrieves the primary database Maintenance User name from the Replication Server.
- When ra_maintid is invoked, it returns the name of the primary database Maintenance User currently stored in the RASD, as follows:

```
Maintenance User
 ----------------
 SYS
(1 row affected)
```

- If ra_maintid is invoked when the Mirror Replication Agent is in *Replicating* state, the correct Maintenance User name is always returned, because it is retrieved from the Replication Server when the Mirror Replication Agent goes to *Replicating* state.

  If ra_maintid is invoked when the Mirror Replication Agent is in *Admin* state, the value returned from the RASD may not be correct, because it could have been changed in the Replication Server after the last time the Mirror Replication Agent retrieved the value and stored it in the RASD.
- If ra_maintid is invoked when the primary database connection is down, it returns an error.
- The ra_maintid command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_config, ra_statistics

ra_set_login

Description Sets the Mirror Replication Agent administrator login and password.

Syntax ra_set_login *username*, *password*

Parameters *username*
The login name of the Mirror Replication Agent administrator.

password
The password of the Mirror Replication Agent administrator.

Examples
```
ra_set_login bob3, bug3wag
```

This command sets the Mirror Replication Agent administrator login to “bob3” and the password to “bug3wag.”

Usage
- The Mirror Replication Agent administrator login has permission to log in to the Mirror Replication Agent instance through the administration port.
- Only one Mirror Replication Agent administrator login name is valid at any time.

- Any change in the Mirror Replication Agent administrator login or password takes place immediately, and you must use the new login and password the next time you log in to the Mirror Replication Agent.
- The password specified for the administrator login is encrypted in the Mirror Replication Agent configuration file.
- The ra_set_login command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_config

ra_statistics

Description Returns performance-related statistics for Mirror Replication Agent components and the Java Virtual Machine (Java VM), or resets the statistics counters.

Syntax ra_statistics [*component*|reset]

Parameters

component

The optional keyword that identifies a Mirror Replication Agent component or the Java VM. Valid *component* keywords are:

LR – Log Reader
LTI – Log Transfer Interface
LTM – Log Transfer Manager
VM – Java Virtual Machine

reset

The optional keyword that resets the statistics counters.

Examples

Example 1

```
ra_statistics
```

This command returns performance statistics for the Mirror Replication Agent instance and the Java VM.

Example 2

```
ra_statistics reset
```

This command resets the statistics counters for the Mirror Replication Agent instance.

Usage

- If you invoke ra_statistics with no option, it returns statistics for all Mirror Replication Agent components and the Java VM.
- If you invoke ra_statistics with a *component* option, it returns statistics for the Log Transfer Manager component, as well as the component (or Java VM) you specify.
- Table 4-2 lists the statistics returned for the Java VM.

Table 4-2: Java VM statistics

Component	Statistic	Description
VM	VM maximum memory	Maximum memory (in bytes) available to the Java VM
VM	VM total memory allocated	Total memory (in bytes) allocated to the Java VM at start-up
VM	VM free memory	Memory (in bytes) allocated, but not used by the Java VM
VM	VM memory usage	Memory (in bytes) allocated and in use by the Java VM
VM	VM % max memory used	Percentage of the maximum memory available to the Java VM, currently in use by the Java VM

- Table 4-3 lists the statistics returned for the Log Transfer Manager component.

Table 4-3: Log Transfer Manager statistics

Component	Statistic	Description
LTM	Time statistics obtained	Day, date, and time ra_statistics was invoked and information returned
LTM	Time replication last started	Day, date, and time that *Replicating* state was entered
LTM	Time statistics last reset	Day, date, and time that statistics counters were reset
LTM	Items held in Global LRUCache	Number of object references in the internal Least Recently Used cache

- Table 4-4 lists the statistics returned for the Log Reader component.

Table 4-4: Log Reader statistics

Component	Statistic	Description
LR	Total pages read from log device	Number of pages read from log devices since last reset

Component	Statistic	Description
LR	Avg rows per page	Average number of rows per page read from log devices since last reset
LR	Last page read	Block and page number of most recent page read from log devices
LR	Total operations scanned	Number of operations read from log devices since last reset
LR	Total operations processed	Number of operations read from log devices and passed to LTI since last reset
LR	Total operations skipped	Number of operations read from log devices and not processed for any reason since last reset
LR	Total maintenance user operations filtered	Number of Maintenance User operations read from log devices and skipped since last reset
LR	Avg operation processing time	Average Log Reader operation processing time (in milliseconds) since last reset
LR	Total transactions processed	Number of transactions read from log devices since last reset
LR	Total transactions skipped	Number of transactions read from log devices and not processed for any reason since last reset
LR	Total transactions opened	Number of begin transaction commands read from log devices since last reset
LR	Total transactions closed	Number of commit and rollback commands read from log devices since last reset
LR	Total transactions committed	Number of commit commands read from log devices since last reset
LR	Total transactions aborted (rolled back)	Number of rollback commands read from log devices since last reset
LR	Total system transactions skipped	Number of system transactions read from log devices and skipped since last reset
LR	Avg ops per transaction	Average number of operations in each transaction read from log devices since last reset
LR	Current scan buffer size	Current size (in bytes) of the Log Reader scan buffer
LR	Current operation queue size	Current size (in bytes) of the Log Reader input queue
LR	Log reposition point locator	Locator value of reposition point in log device

Component	Statistic	Description
LR	Last processed operation locator	Locator value of most recently processed operation read from log devices
LR	Avg xlog operation wait time (ms)	Average time (in milliseconds) that Log Reader had to wait for each new operation to appear in the log since last reset
LR	Avg sender operation processing time (ms)	Average time (in milliseconds) that Log Reader sender took to process each operation since last reset
LR	Avg sender operation wait time (ms)	Average time (in milliseconds) that Log Reader sender had to wait to send each processed operation to the LTI input queue since last reset
LR	Avg ChangeSet send time (ms)	Average time (in milliseconds) that Log Reader sender took to send each processed operation to the LTI input queue since last reset
LR	Total sender operations processed	Number of operations that Log Reader sender processed since last reset

- Table 4-5 lists the statistics returned for the Log Transfer Interface component.

Table 4-5: Log Transfer Interface statistics

Component	Statistic	Description
LTI	Number of LTL commands sent	Total number of LTL commands sent to Replication Server since last reset
LTI	Avg LTL command size	Average size (in bytes) of each LTL command sent to Replication Server since last reset
LTI	Avg LTL commands/sec	Average number of LTL commands sent per second to Replication Server since last reset
LTI	Total bytes sent	Number of bytes sent to Replication Server since last reset
LTI	Avg Bytes/second during transmission	Average bytes per second sent over connection to Replication Server since last reset
LTI	Avg Rep Server turnaround time	Average time (in milliseconds) it takes Replication Server to acknowledge each LTL command buffer sent since last reset

Component	Statistic	Description
LTI	Avg data arrival time	Average time (in milliseconds) LTI waits between receiving change sets from Log Reader since last reset
LTI	Avg time to create distributes	Average time (in milliseconds) LTI takes to convert a change-set into LTL since last reset
LTI	Avg LTL buffer size	Average size (in bytes) of each LTL buffer sent to Replication Server since last reset
LTI	Avg LTM buffer utilization (%)	Average utilization (in percentage of LTL buffer size) of each LTL buffer sent to Replication Server since last reset
LTI	Avg LTL commands/buffer	Average number of LTL commands per buffer sent to Replication Server since last reset
LTI	Input queue size	Current number of change sets in the LTI input queue
LTI	Output queue size	Current number of distributes in the LTI output queue
LTI	Last QID sent	Hex value of most recent origin queue ID sent to Replication Server
LTI	Last transaction id sent	Hex value of most recent transaction ID sent to Replication Server

- Statistics counters are reset automatically each time the Mirror Replication Agent instance goes into *Replicating* state.
- If you invoke ra_statistics with the reset keyword, Mirror Replication Agent immediately resets all of the statistics, except the following:
  - Time statistics obtained (LTM)
  - Time replication last started (LTM)
  - Time statistics last reset (LTM)
  - Last QID sent (LTI)
  - Last transaction ID sent (LTI)
  - All Java VM statistics

Note All Java VM statistics are refreshed each time ra_statistics returns statistics values.

- The ra_statistics command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_status

ra_status

Description Returns the current state of the Mirror Replication Agent instance.

Syntax ra_status

Usage

- When ra_status is invoked, it returns the current state of the Mirror Replication Agent instance, and a brief description of the current state, as follows:

```
State  Action
 ------ ---------------------------
 ADMIN  Waiting for operator command
(1 row affected)
```

Note If the first word in the description is "Transitioning," the Mirror Replication Agent instance is in transition between states. Some commands are not valid when the Mirror Replication Agent instance is in state transition.

- Mirror Replication Agent states are:
  - *Admin* – in this state, the Mirror Replication Agent instance is running, but no connections are up. You can change any configuration parameter when the Mirror Replication Agent instance is in *Admin* state.
  - *Replicating* – in this state, the Mirror Replication Agent Log Reader component is scanning the transaction log for operations to replicate from the primary database. If there are operations to be replicated, the Log Transfer Interface component is sending LTL commands to the Replication Server.
  - *Replicating (Waiting at end of log)* – in this state, the Mirror Replication Agent Log Reader component has reached the end of the transaction log, and the Mirror Replication Agent has finished processing all operations in the transaction log and successfully sent LTL commands for all replicated operations to the Replication Server.

If the primary database is not quiesced or otherwise inactive, transactions may arrive in the log immediately after the state is returned by the Log Transfer Manager component, so even though the state is returned as *Replicating (Waiting at end of log)*, the Mirror Replication Agent may actually be in *Replicating* state and processing log records.

See "Mirror Replication Agent states" on page 79 for more information.

- The ra_status command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also: pdb_init, pdb_quiesce, quiesce, ra_init, ra_statistics, resume, shutdown, suspend

ra_truncatearticles

Description: Truncates older versions of primary database articles in the system data repository in the RASD.

Syntax: ra_truncatearticles *locator*

Parameters:

locator
The log locator value (LTM Locator) that identifies the cutoff point for truncating older versions of articles from the system data repository.

Usage:

- When ra_truncatearticles is invoked, it truncates all non-current versions of all primary database articles in the system data repository older than the version identified by the *locator* value.

  If the current (most recent) version of an article is older than the version identified by the *locator* value, it is not truncated.

- Most common data definition language (DDL) commands and stored procedures executed in the primary database (such as alter table) are recorded in the transaction log, and replicated to the standby database. When it processes those DDL transactions for replication, Mirror Replication Agent updates its RASD automatically, creating a new version of the affected primary database article(s).

  Use ra_truncatearticles as part of a periodic maintenance procedure to prevent the RASD from growing indefinitely. See "Maintaining the Replication Agent System Database" on page 82 for more information.

- You should back up the RASD using rasd_backup before you truncate it. See “Backing up the RASD” on page 89 for more information.
- The ra_truncatearticles command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_truncateusers

ra_truncateusers

Description Truncates older versions of primary database users in the system data repository in the RASD.

Syntax ra_truncateusers *locator*

Parameters *locator*

The log locator value (LTM Locator) that identifies the cutoff point for truncating older versions of database users from the system data repository.

Usage

- When ra_truncateusers is invoked, it truncates all non-current versions of all primary database users in the system data repository older than the version identified by the *locator* value.

  If the current (most recent) version of a user is older than the version identified by the *locator* value, it is not truncated.
- Most common data definition language (DDL) commands and stored procedures executed in the primary database (such as sp_modifylogin) are recorded in the transaction log, and replicated to the standby database. When it processes those DDL transactions for replication, Mirror Replication Agent updates its RASD automatically, creating a new version of the affected primary database user(s).

  Use ra_truncateusers as part of a periodic maintenance procedure to prevent the RASD from growing indefinitely. See “Maintaining the Replication Agent System Database” on page 82 for more information.
- You should back up the RASD using rasd_backup before you truncate it. See “Backing up the RASD” on page 89 for more information.
- The ra_truncateusers command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also ra_truncatearticles

ra_updatedevices

Description

Updates (re-initializes) the log device repository in the RASD.

Syntax

ra_updatedevices

Usage

- When ra_updatedevices is invoked, Mirror Replication Agent does the following:
  - Deletes all of the data in its log device repository
  - Queries the primary database for information about all of its log devices
  - Re-populates the log device repository in the RASD with current information about primary database log devices returned by the primary database
- If any log device is added, dropped, extended, or moved at the primary database, you must:
  - Stop replication (using quiesce or suspend) to put the Mirror Replication Agent instance in *Admin* state
  - Invoke ra_updatedevices to update the log device repository in the RASD

  See “Updating the log device repository” on page 86 for more information.

  Note The primary database need not be quiesced when you update the log device repository.
- If the primary data server writes to a new (or altered) log device before you update the log device repository, the Mirror Replication Agent instance will stop replication processing and go to *Admin* state.

  Sybase recommends that you coordinate all log device changes at the primary database with updating the Mirror Replication Agent log device repository.
- Because Mirror Replication Agent re-creates the entire log device repository when you invoke ra_updatedevices, any log device path that you modified previously (using ra_devicepath) is overwritten with the current log device information from the primary database.

Note If you need to alter the "default" path for a log device (that is, the log device path returned by the primary database), you must use the ra_devicepath command *after* you invoke ra_updatedevices.

- Each log device path can be altered by the ra_devicepath command, to "re-map" the current path recorded in the RASD to point to the corresponding mirror log device at the standby site. Mirror Replication Agent uses the log device paths recorded in its RASD to find the mirror log devices, from which it reads the primary database transaction log.
- The ra_updatedevices command is valid only when the Mirror Replication Agent instance is in *Admin* state.

See also ra_devicepath, ra_helpdevice

ra_version

Description Returns the version of the Mirror Replication Agent instance, the host operating system version, and the JRE version.

Syntax ra_version

Usage

- When ra_version is invoked, it returns the Mirror Replication Agent version string in a row, as follows:

```
Mirror Replication Agent/12.6.0.4042/P/generic/JDK
1.4.1/rax404perf/4042/VM: Sun Microsystems Inc.
1.4.1_05/OPT/Thu Feb 19 03:00:01 MST 2004
```

- The ra_version command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_version, ra_status, ra_version_all

ra_version_all

Description — Returns the name, type, and version of the Mirror Replication Agent instance, and version information for the primary data server, Replication Server, and communications drivers.

Syntax — ra_version_all

Usage

- When ra_version_all is invoked, it returns the following information:

```
Component              Version

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

 Instance:             mra_inst - Sybase Adaptive Server Enterprise
RepAgent:            Mirror Replication Agent/12.6.0.4042/P/generic/JDK
1.4.1/rax404perf/4042/VM: Sun Microsystems Inc. 1.4.1_05/OPT/Thu Feb 19
03:00:01 MST 2004
JRE:               Sun Microsystems Inc. Java(TM) 2 Runtime Environment,
Standard Edition/1.4.1_05-b01/Windows 2000 5.0/x86/32
RASD:             Sybase Adaptive Server Anywhere/8.0.2.4285/Windows2000
Primary Data Server:  Adaptive Server Enterprise/12.5.1/EBF 11428/P/NT
(IX86)/OS 4.0/ase1251/1823/32-bit/OPT/Wed Sep 17 11:10:54 2003
PDS JDBC Driver:      jConnect (TM) for JDBC(TM)/5.5(Build
25200)/P/EBF11473/JDK12/Tue Sep 16 15:18:30 2003
RepServer:         Replication Server/12.0/P/Sun_svr4/OS  5.6/1/OPT/Fri
Dec 10 19:07:38 PST 1999
RSSD:               Adaptive Server Enterprise/12.0.0.6/P/EBF 10627 ESD
1/Sun_svr4/OS 5.6/1918/32bit/FBO/Fri Oct 18 07:09:21 2002
Sybase JDBC Driver:   jConnect (TM) for JDBC(TM)/5.5(Build
25200)/P/EBF11473/JDK12/Tue Sep 16 15:18:30 2003

(9 rows affected)
```

- All information returned by the ra_version_all command is recorded in the RASD.
- The ra_version_all command returns information about the primary data server and communications driver, and the Replication Server and communications driver, only if the connections to those servers are properly configured in the Mirror Replication Agent.

  If a connection is not configured, or is configured incorrectly, ra_version_all returns `<unknown>` for that connection.

Note The ra_version_all command always returns information about the Mirror Replication Agent version and instance.

- The ra_version_all command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also pdb_version, ra_status, ra_version

rasd_backup

Description Backs up the Replication Agent System Database (RASD).

Syntax rasd_backup

Usage

- When rasd_backup is invoked, it starts the database backup process for the RASD.

Note Sybase recommends that you always back up the RASD after you truncate the primary database log.

- Mirror Replication Agent places RASD backup files in the directory identified by the rasd_backup_dir configuration parameter.

  When you create a Mirror Replication Agent instance, a RASD backup directory is created automatically as part of the instance directory structure. The default value of the rasd_backup_dir parameter points to that directory.

- The rasd_backup command is valid when the Mirror Replication Agent instance is in *Admin* or *Replicating* state.

See also ra_config, rasd_restore

rasd_restore

Description Restores the Replication Agent System Database (RASD).

Syntax rasd_restore

Usage

- When rasd_restore is invoked, it starts the restore process for the RASD.

- Mirror Replication Agent looks for the most recent RASD backup files in the directory identified by the rasd_backup_dir configuration parameter.

  When you create a Mirror Replication Agent instance, a RASD backup directory is created automatically as part of the instance directory structure. The default value of the rasd_backup_dir parameter points to that directory.
- If you invoke rasd_restore when the Mirror Replication Agent instance is in *Replicating* state, it returns an error.
- The rasd_restore command is valid only when the Mirror Replication Agent instance is in *Admin* state.

See also rasd_backup

resume

Description Starts replication processing in the Mirror Replication Agent instance.

Syntax resume

Usage

- When resume is invoked, the Mirror Replication Agent instance attempts to start replication and go to *Replicating* state, as follows:
  - Mirror Replication Agent attempts to open network connections to the primary database, Replication Server, and RSSD.

    Note If it fails to establish a connection to the primary database, the Mirror Replication Agent logs a warning message in its system log and attempts to retry the connection, based on its configuration parameters for the connection.
  - Mirror Replication Agent checks the RASD to make sure it contains a valid system data repository, and validates its read access to the mirror log devices.
  - Mirror Replication Agent requests the current LTM Locator value from the Replication Server, and it stores the LTM Locator value in the RASD.
  - The Log Reader component begins scanning the primary database transaction log on the mirror log devices, looking for operations to be

replicated. Log Reader begins scanning the transaction log at the point identified by the LTM Locator value.

- When it finds transactions to replicate, Log Reader passes them (as change-set data) to the input queue of the Log Transfer Interface component.
- The Log Transfer Interface component reads the change-set data from its input queue, generates LTL commands to send to the Replication Server, and passes the LTL commands to its output queue for transmission to the Replication Server.

- If any startup operation fails, except establishing a connection to the primary database, the Mirror Replication Agent instance returns to *Admin* state, and it logs the error.
- If the resume command is successful, the Mirror Replication Agent instance goes to *Replicating* state. To determine the current state of the Mirror Replication Agent, use the ra_status command.
- The resume command returns an error under any of the following conditions:
  - The Mirror Replication Agent instance is already in *Replicating* state.
  - The system data repository in the RASD does not exist.
  - The Mirror Replication Agent connection configuration parameters are not set correctly, or it fails otherwise to connect with the Replication Server.
  - The database connection for the primary database is not defined correctly in the Replication Server.
  - The Mirror Replication Agent connection configuration parameters are not set correctly.
- The resume command is valid only when the Mirror Replication Agent instance is in *Admin* state.

See also pdb_quiesce, quiesce, ra_status, shutdown, suspend

shutdown

Description — Shuts down the Mirror Replication Agent instance.

Syntax — shutdown [immediate]

Parameters — immediate

The optional keyword that shuts down the Mirror Replication Agent instance immediately.

Usage

- When shutdown is invoked with no option, a normal (graceful) shutdown is performed.

  In a normal shutdown, the Mirror Replication Agent first quiesces, and then the process terminates. For more information about quiescing the Mirror Replication Agent, see quiesce on page 113.

- When shutdown is invoked with the immediate keyword, an immediate shutdown is performed.

  In an immediate shutdown, the Mirror Replication Agent instance stops all processing, without regard to transactions in process or in transit, drops all connections immediately, and then terminates the application.

- The shutdown command with the immediate keyword is valid at any time, when the Mirror Replication Agent instance is in any state, including transition between states.

- The shutdown command with no keyword (normal shutdown) is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state, but not in state transition.

See also — quiesce, ra_status, resume, suspend

suspend

Description — Stops all replication processing and puts the Mirror Replication Agent instance in *Admin* state.

Syntax — suspend

Usage

- When suspend is invoked, it stops all current Mirror Replication Agent processing.

  - The Log Reader component stops scanning the transaction log immediately, and the Log Transfer Interface component stops sending LTL to the Replication Server immediately.
  - Any 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 immediately 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.
- If the Mirror Replication Agent instance is in *Admin* state, the suspend command returns an error.
- The suspend command is valid only when the Mirror Replication Agent instance is in *Replicating* state.

See also pdb_quiesce, quiesce, ra_status, resume, shutdown

test_connection

Description Tests all Mirror Replication Agent connections, or a specified connection.

Syntax test_connection [*conn_name*]

Parameters *conn_name*

The keyword for a connection to be tested. Valid keywords are:

- PDS – tests the connection to the primary data server.
- RS – tests the connection to the Replication Server and RSSD.

Examples **Example 1**

```
test_connection
```

This command tests all Mirror Replication Agent network connections, to the primary data server, Replication Server, and RSSD.

Example 2

```
test_connection RS
```

This command tests only connections to the Replication Server and RSSD.

Usage

- When test_connection is invoked with no option, Mirror Replication Agent tests all network connections.
- When test_connection is invoked with either the RS or PDS keyword, Mirror Replication Agent tests the specified network connection(s).
- If the value of the use_rssd parameter is true, the test_connection command tests the Mirror Replication Agent connection to the RSSD when it tests the Replication Server connection.
- Mirror Replication Agent tests each connection by attempting to log in to the corresponding server, using the connection parameters recorded in its configuration file. See "Setting up the connection configuration parameters" on page 39 for more information.

Note Even if test_connection returns a successful Replication Server connection, the Log Transfer Interface component still might not be able to successfully connect to the Replication Server. In that event, either the primary database connection does not exist (or was created incorrectly) in the Replication Server, or the Replication Server user login name for the Mirror Replication Agent instance does not have connect source permission.

- The test_connection command returns the connection type and its status.

  If the connection status is failed, it indicates one of the following:

  - The Mirror Replication Agent connection configuration parameters are not set correctly.
  - A network or communication error prevents the connection.
  - The specified server is down.
- If the connection status is failed, check the Mirror Replication Agent system log file to determine the cause of the failure. See Chapter 6, "Troubleshooting Mirror Replication Agent," for more information.
- The test_connection command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also

ra_statistics

trace

Description

Returns current trace flag settings, or changes trace flag settings for the Mirror Replication Agent instance.

Syntax

trace [*flag*|all, *switch*]

Parameters

flag
The name of the trace flag to change the setting for.

all
A keyword that allows you to apply a switch value to all of the trace flags at once.

switch
A Boolean (true or false) value that enables or disables tracing for the trace point identified in the *flag* option.

Usage

- The trace command is intended for use by Sybase Technical Support engineers when troubleshooting Mirror Replication Agent.
- When trace is invoked with no option, it returns the current settings for all Mirror Replication Agent trace flags.
- When trace is invoked with the *flag* and *switch* options, it changes the setting of the trace flag identified, and it returns the current (new) setting for the trace flag.
- When trace is invoked with the all keyword and a *switch* option, it sets all of the trace flags to the value specified in the *switch* option, and it returns the current (new) setting for all of the trace flags.
- Changes made with the trace command take effect immediately.
- When a trace flag is set to true, tracing is enabled for the trace point(s) identified by the flag. When set to false, tracing is disabled for the trace point(s).
- Normal output from trace points is sent to the Mirror Replication Agent system log file. Use the log_system_name command to find the name and path of the Mirror Replication Agent system log file.
- Output from the *LTITRACELTL* trace point is sent to a separate Log Transfer Language (LTL) output file named *LTITRACELTL.log*. If you want to view the contents of the *LTITRACELTL.log* file, your viewer must be capable of handling very long lines.

Note The *LTITRACELTL.log* file contains a human-readable representation of the LTL output, not the actual LTL sent to the Replication Server.

- Table 4-6 lists Mirror Replication Agent trace flags:

Table 4-6: Mirror Replication Agent trace flags

Trace flag	Description
BMGRTRACE	Traces "Bean Management" events.
CACHETRC	Traces internal global cache events.
LATRC	Traces Log Admin events.
LATRCSQL	Traces Log Admin conversations with primary database.
LICTRACE	Traces feature license check-in/checkout events.
LOGREADTRC	Traces ASE log-reading events.
LRTRACE	Traces general Log Reader events.
LTITRACE	Traces general Log Transfer Interface events.
LTITRACELTL	Records LTL output in *LTITRACELTL.log* file.
LTMCI	Traces Log Transfer Manager interface events.
LTMHL	Traces highlights in Log Transfer Manager execution.
LTMSC	Traces Mirror Replication Agent state changes.
RACONTRC	Traces connection and query execution.
RACONTRCSQL	Traces SQL statements sent to primary database.
RASDTRC	Traces RASD events.
RATRACE	Traces general Mirror Replication Agent events.
STMTRACE	Traces Log Transfer Manager state monitor events.
THREADTRC	Traces ThreadPool events.

- You *cannot* change the settings of SYSTEM trace flags.
- Output from SYSTEM trace points is sent to the system log file. Use the log_system_name command to find the name and path of the Mirror Replication Agent system log file.
- Table 4-7 lists Mirror Replication Agent SYSTEM trace flags:

Table 4-7: Mirror Replication Agent SYSTEM trace flags

Trace flag	Description
CONFIG	Configuration change event logged.
ERROR	Serious error; manual intervention may be needed to recover.
FATAL	Critical error; application shut down; manual intervention required to recover.

Trace flag	Description
INFORMATION	Information only; no action required.
WARNING	Minor error; operation not affected, or problem is recoverable.

- The trace command is valid when the Mirror Replication Agent instance is in either *Admin* or *Replicating* state.

See also log_system_name

CHAPTER 5

Mirror Replication Agent Configuration Parameters

This chapter describes the Mirror Replication Agent configuration file and configuration parameters.

Configuration parameter overview

Configuration parameters record the user-configurable settings that control how a Mirror Replication Agent instance operates. The current values of all configuration parameters are stored in the *configuration file* of each Mirror Replication Agent instance.

Mirror Replication Agent configuration file

The configuration file is created automatically when you create a Mirror Replication Agent instance. It resides in the instance subdirectory, under the Mirror Replication Agent base directory.

The configuration file is named after the Mirror Replication Agent instance, with the extension *.cfg* (for example, if the instance is named "my_mra," the configuration file is *my_mra.cfg*).

Each time a Mirror Replication Agent instance starts up, it reads the configuration file to get the configuration information it needs to run. After start-up, the only time the Mirror Replication Agent accesses the configuration file is when the ra_config or ra_set_login command is invoked to change the value of a configuration parameter.

When the value of a configuration parameter is changed, Mirror Replication Agent saves the new value, overwriting the entire configuration file.

Configuration file format

The configuration file is a flat ASCII file that contains configuration information for a single Mirror Replication Agent instance.

The first two lines in the configuration file identify the file as a Mirror Replication Agent configuration file and record the time that the file was last modified. For example:

```
#MRA Property File
 #Tue Feb 16 22:41:21 GMT+00:00 2004
```

Each configuration parameter name appears on a separate line, followed by the equal symbol (=) and the current value of the parameter. For example:

```
compress_ltl_syntax=true
```

If the Mirror Replication Agent instance is not running, you can view the configuration file to examine the current Mirror Replication Agent configuration.

Note Sybase recommends that you do *not* edit the configuration file, because Mirror Replication Agent overwrites the entire configuration file every time the ra_config or ra_set_login command is invoked to change a parameter value.

If the Mirror Replication Agent instance is running, use the ra_config command to view the current Mirror Replication Agent configuration.

Changing configuration parameters

To view, set, or change the current value of a Mirror Replication Agent configuration parameter, use the ra_config command.

To change the current Mirror Replication Agent administrator login (ltm_admin_user) or administrative user password (ltm_admin_pw), you must use the ra_set_login command.

Note The ltm_admin_user and ltm_admin_pw parameters cannot be changed with the ra_config command, and they do not appear in the parameter list returned by ra_config.

See Chapter 4, "Mirror Replication Agent Command Reference," for more information about using the ra_config and ra_set_login commands.

Copying a Mirror Replication Agent configuration

When you create a new Mirror Replication Agent instance with the mra_admin utility, you can specify that the new instance uses the same configuration parameter values as an existing Mirror Replication Agent instance.

Note When you copy an existing configuration for a new Mirror Replication Agent instance, certain configuration parameter values are not copied to the new configuration. See "Copying a Mirror Replication Agent configuration" on page 22 for more information.

If you do not copy an existing configuration when you create a new Mirror Replication Agent instance, the mra_admin utility creates a default configuration file, with default values for all configuration parameters. See "Using the mra_admin utility" on page 18 for more information.

Configuration parameter reference

Table 5-1 lists all of the Mirror Replication Agent configuration parameters, with a brief description of each parameter.

Table 5-1: Mirror Replication Agent configuration parameters

Parameter name	Description
admin_port	Port number for Mirror Replication Agent
column_compression	Use minimal column information
compress_ltl_syntax	Use abbreviated LTL syntax
connect_to_rs	Enable/disable Replication Server connection
dump_batch_timeout	Timeout to send incomplete LTL buffer
filter_maint_userid	Ignore Maintenance User operations

Parameter name	Description
log_backup_files	Number of log backup files kept
log_directory	System log file directory
log_trace_verbose	Verbose trace in log
log_wrap	Number of 1K blocks before log file wrap
lti_batch_mode	Send LTL in batch mode
lti_max_buffer_size	Number of change sets in LTI queues
lti_update_trunc_point	Number of LTL commands before Locator update
ltl_batch_size	Size of the LTL batch buffer
ltl_character_case	Character case of database object names in LTL
ltl_origin_time_required	LTL origin_time command tag
ltm_admin_pw	Password for Mirror Replication Agent
ltm_admin_user	User login for Mirror Replication Agent
max_ops_per_scan	Number of operations in Log Reader buffer
pds_charset	Character set on primary database connection
pds_connection_type	Type of connection to primary data server
pds_database_name	Name of primary database
pds_datasource_name	Data source name (DSN) of primary database
pds_host_name	Name of primary data server host machine
pds_password	Password for primary database
pds_port_number	Port number for primary data server
pds_retry_count	Number of primary database connection retries
pds_retry_timeout	Primary database connection retry timeout
pds_server_name	Server name of primary data server
pds_username	User login for primary database
ra_retry_count	Number of replication retries after a failure
ra_retry_timeout	Replication retry timeout
rasd_backup_dir	RASD backup file directory
rasd_database	RASD database file directory
rasd_mirror_tran_log	Enable/disable RASD log mirroring
rasd_trace_log_dir	RASD trace log directory
rasd_tran_log	RASD transaction log directory
rasd_tran_log_mirror	RASD mirror log directory
rs_charset	Character set on Replication Server connection
rs_host_name	Name of Replication Server host machine
rs_packet_size	Packet size on Replication Server connection
rs_password	Password for Replication Server
rs_port_number	Port number for Replication Server

Parameter name	Description
rs_retry_count	Number of Replication Server connection retries
rs_retry_timeout	Replication Server connection retry timeout
rs_source_db	Name of primary database in Replication Server database connection
rs_source_ds	Name of primary data server in Replication Server database connection
rs_username	User login for Replication Server
rssd_charset	Character set on RSSD connection
rssd_database_name	Database name of RSSD
rssd_host_name	Name of RSSD host machine
rssd_password	Password for RSSD
rssd_port_number	Port number for RSSD
rssd_username	User login for RSSD
scan_sleep_increment	Seconds to increase log scan wait timeout
scan_sleep_max	Maximum log scan wait timeout
skip_ltl_errors	Ignore LTL errors
structured_tokens	Use structured tokens in LTL
use_rssd	Use RSSD for replication definitions

The following sections list all Mirror Replication Agent configuration parameters in alphabetical order.

admin_port

The client socket port number of the Mirror Replication Agent.

Default

"" (empty string)

Value

A valid port number on the Mirror Replication Agent host machine.

Comments

- When you create a Mirror Replication Agent instance, you must specify a client socket port number for the instance administration port. Client applications use this port number to connect to the Mirror Replication Agent instance.
- You must specify a port number that does not conflict with any port numbers already in use on the Mirror Replication Agent host machine.

- If you change the value of the admin_port parameter with the ra_config command, the new value is recorded in the configuration file immediately, but you must shut down and restart the Mirror Replication Agent instance to make the new port number take effect.
- After you change the value of the admin_port parameter with the ra_config command, the next time you log in to the Mirror Replication Agent administration port, you must use the new port number.

column_compression

Determines whether the Log Transfer Interface component sends all columns in row after images, or only the columns that changed in an update operation.

Default: true

Values: true – Enables minimal column information (only changed columns in row after images) in LTL for update operations.

false – Disables minimal column information in LTL for update operations.

Comments:

- When the column_compression parameter is set to false, the Log Transfer Interface component sends complete row after images in LTL, including columns in which no data changed as a result of an update operation.
- When the column_compression parameter is set to true, the Log Transfer Interface component sends minimal column information in the row after images in LTL, with only the columns that changed as a result of an update operation. Columns in which no data changed as a result of the update are not sent in LTL.
- In general, setting the value of the column_compression parameter to true provides better Mirror Replication Agent throughput.

compress_ltl_syntax

Determines whether the Log Transfer Interface component compresses LTL commands using abbreviated syntax.

Default: true

Values: true – Enables LTL compression, using abbreviated LTL syntax.

false – Disables LTL compression.

Comments

- In general, setting the value of the compress_ltl_syntax parameter to true will provide better Mirror Replication Agent throughput.
- See the Replication Server *Administration Guide* and *Reference Manual* for more information about LTL commands and abbreviated LTL syntax.

connect_to_rs

Enables or disables the network connection to the Replication Server.

Default

true

Values

true – Enables the network connection to the Replication Server.

false – Disables the network connection to the Replication Server.

Comments

- When the value of the connect_to_rs parameter is false, the network connection from Mirror Replication Agent to Replication Server is disabled, and no replication can occur.
- When the network connection to Replication Server is disabled by the connect_to_rs parameter, the Mirror Replication Agent instance can still go to *Replicating* state, with the following limitations:
  - A “dummy” connection in the Mirror Replication Agent emulates a real connection to the Replication Server.
  - The value of the LTM Locator stored in the RASD is set to zero.
  - The Maintenance User name is set to an invalid user ID.

  Note Maintenance User operations cannot be filtered when the value of the connect_to_rs parameter is false.
- You can use the connect_to_rs parameter to temporarily disable the network connection to the Replication Server for testing.
- When the value of the connect_to_rs parameter is false, you can put the Mirror Replication Agent instance in *Replicating* state, set the value of the LTITRACELTL trace flag to true, and view a readable representation of the LTL that would have been sent to the Replication Server if the connection was not disabled.
- During normal Mirror Replication Agent operation, the value of the connect_to_rs parameter must be true.

dump_batch_timeout

The time interval to wait before sending the contents of the Log Transfer Interface buffer to the Replication Server, even though the buffer is not full.

Default: 5

Value: An integer from 1 to 60.

Comments:

- The value of the dump_batch_timeout parameter is the number of seconds from the time the previous Log Transfer Interface buffer was sent to the Replication Server until the next buffer will be sent.
- The dump_batch_timeout parameter has no effect if the value of the lti_batch_mode parameter is false.

filter_maint_userid

Determines whether operations applied by the Maintenance User are ignored.

Note Mirror Replication Agent ignores the value of the filter_maint_userid parameter it is in a Replication Server warm standby application.

Default: true

Values: true – Enables the Log Reader to ignore Maintenance User operations.

false – Disables the Log Reader filter to allow replicating Maintenance User operations.

Comments:

- The filter_maint_userid configuration parameter is provided to support bidirectional replication, in which the primary database also serves as a standby (or replicate) database that has transactions applied to it by a Replication Server Maintenance User.
- If the value of the filter_maint_userid parameter is true, database operations applied by the Maintenance User are *not* replicated. The Log Reader component filters out (ignores) operations applied by the Maintenance User when it reads the transaction log.
- If the value of the filter_maint_userid parameter is false, database operations applied by the Maintenance User are replicated. The Log Reader component replicates all operations on marked objects, regardless of the user that applied the operation.

- The Maintenance User login is specified when the database connection for the primary database is created in the Replication Server.

log_backup_files

The number of backup system log files kept in the *log* directory.

Default: 3

Values: An integer greater than or equal to 1.

Comments:

- When the system log wraps, Mirror Replication Agent copies the current log file to a backup file, with a generated number appended to the file's name.

  For example, if the system log file is named *my_mra.log*, the first backup file created when the system log wraps would be named *my_mra1.log*. The second backup file created would be named *my_mra2.log*, and so on.

- When the number of backup files exceeds the value of the log_backup_files parameter, the oldest backup file (that is, the one with the lowest generated number) is deleted from the *log* directory before the next backup file is created.

log_directory

The directory for Mirror Replication Agent system log files.

Default: The path to the *log* directory created when the Mirror Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

  ```
  %SYBASE%\MRA-12_6\inst_name\log
  ```

  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:

  ```
  $SYBASE/MRA-12_6/inst_name/log
  ```

  where:

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

Value

A valid path on the Mirror Replication Agent host machine.

Comments

- When a Mirror Replication Agent instance is created, the *log* directory is created as part of the instance directories. The default value of the log_directory parameter points to that directory.
- If you specify any valid path as the value of the log_directory parameter, the Mirror Replication Agent instance places its system log files in the directory you specify the next time it is started.
- If you specify the default value of the log_directory parameter by using the default keyword in the ra_config command, then the next time it is started, Mirror Replication Agent will place its system log files in the *log* directory that was created when the Mirror Replication Agent instance was created.
- If you change the value of the log_directory parameter with the ra_config command, the new value is recorded in the configuration file immediately, but you must shut down and restart the Mirror Replication Agent instance to make the new value take effect.

log_trace_verbose

Enables or disables additional diagnostic information in log files.

Default

false

Values

true – Enables detailed diagnostic information in log files.

false – Disables detailed diagnostic information in log files.

Comment

Detailed diagnostic information is intended for troubleshooting only, with assistance from Sybase Technical Support.

log_wrap

The maximum size of system log files before wrapping.

Default

1000

Value

An integer greater than or equal to 500.

Comments

- The value of the log_wrap parameter is the number of 1K blocks written by Mirror Replication Agent, before it wraps the system log file.
- Larger values for the log_wrap parameter allow more log history in each file. Smaller values produce smaller log files.
- When a system log file wraps, Mirror Replication Agent copies the current log file to a backup file, with a generated number appended to the file's name.

  For example, if the system log file is named *my_mra.log*, the first backup file created when the system log wraps would be named *my_mra1.log*. The second backup file created would be named *my_mra2.log*, and so on.

  When the number of backup files exceeds the value of the log_backup_files parameter, the oldest backup file (that is, the one with the lowest generated number) is deleted from the *log* directory before the next backup file is created.

lti_batch_mode

Enables or disables sending LTL in batch mode.

Default

true

Values

true – Enables sending LTL in batch mode.

false – Disables sending LTL in batch mode.

Comments

- If the value of the lti_batch_mode parameter is true, the Log Transfer Interface component fits as many LTL commands into its LTL batch buffer as it can before it sends any output to the Replication Server.

  The Log Transfer Interface component sends the LTL batch buffer contents to the Replication Server when the time interval specified in the dump_batch_timeout parameter expires, even if its LTL batch buffer is not full.
- If the value of the lti_batch_mode parameter is false, the Log Transfer Interface component sends individual LTL commands to the Replication Server for each change set in its input queue.
- When Mirror Replication Agent connects to a Replication Server, it determines the version of the Replication Server.
  - If the Replication Server version is earlier than 12.5, the size of the LTL batch buffer is set to 16K automatically.

    - If the Replication Server version is 12.5 or later, Mirror Replication Agent sets the size of the LTL batch buffer to the size specified by the ltl_batch_size parameter.
- If the Replication Server version is earlier than 12.5 and the value of the lti_batch_mode parameter is true, if any single LTL distribute command exceeds the 16K size of the LTL batch buffer, Replication Server returns an error and it drops its DSI connection with the Mirror Replication Agent.
- If the Replication Server version is 12.5 or later, you can use the Mirror Replication Agent ltl_batch_size parameter to set the size of the LTL batch buffer.
- In general, setting the value of the lti_batch_mode parameter to true provides better Mirror Replication Agent throughput.

lti_max_buffer_size

The maximum size of the Log Transfer Interface component's queues.

Default: 5000

Value: An integer in the range of 1000 to 100000.

Comments:

- The value of the lti_max_buffer_size parameter is the maximum number of operations that can be stored in the Log Transfer Interface component's inbound and outbound queues.
    - Operations in the inbound queue represent change sets received from the Log Reader component.
    - Operations in the outbound queue are the LTL commands to be sent to the Replication Server.
- The Log Transfer Interface component's inbound queue is a bounded buffer that blocks the processing of the Log Reader component when it gets full.

lti_update_trunc_point

The number of LTL commands sent before requesting a new LTM Locator from the Replication Server.

Default: 1000

Value

An integer from 1 to 100000.

Comments

- The value of the lti_update_trunc_point parameter is the number of LTL commands that Mirror Replication Agent sends to the Replication Server, before it requests a new LTM Locator (secondary truncation point) from the Replication Server.
- Lower numbers cause Mirror Replication Agent to request a new LTM Locator from the Replication Server more often.
- Each time Mirror Replication Agent requests a new LTM Locator from the Replication Server, the secondary truncation point in the Adaptive Server primary database is updated.
- The value of the lti_update_trunc_point parameter is a trade-off between better system performance and longer recovery time:
  - Lower values reduce the time it takes to recover from a replication failure, but they may have an adverse affect on overall system throughput.
  - Higher values improve overall system throughput, but they may increase the time it takes to recover from a replication failure.
- If Mirror Replication Agent is operating in an unreliable network environment, it may be prudent to set the lti_update_trunc_point parameter to a lower value to ensure faster recovery.

ltl_batch_size

The size of the Log Transfer Interface component's LTL batch buffer.

Default

40000

Value

An integer from 1 to 10485760.

Comments

- The value of the ltl_batch_size parameter is the size (in bytes) of the Log Transfer Interface component's LTL batch buffer.
- When Mirror Replication Agent connects to a Replication Server, it determines the version of the Replication Server.
  - If the Replication Server version is earlier than 12.5, the size of the LTL batch buffer is set to 16K automatically, and the value of the ltl_batch_size parameter is ignored.

- If the Replication Server version is 12.5 or later, Mirror Replication Agent sets the size of the LTL batch buffer to the size specified by the ltl_batch_size parameter.

- The Log Transfer Interface component uses the LTL batch buffer only if the value of the lti_batch_mode parameter is true. If the value of the lti_batch_mode parameter is false, the LTL batch buffer is not used.

ltl_character_case

The character case used for database object names in LTL.

Default

asis

Values

asis – Database object names in LTL are sent in the same character case as they are returned from the primary database, or (if the value of the use_rssd parameter is true) in the character case they are specified in replication definitions.

lower – Database object names in LTL are sent in all lowercase, regardless of how they are returned from the primary database or specified in replication definitions.

upper – Databse object names in LTL are sent in all uppercase, regardless of how they are returned from the primary database or specified in replication definitions.

Comments

- The ltl_character_case configuration parameter allows you to customize the handling of database object names in LTL to work with existing replication definitions that specify the object names differently than the way the primary database returns them.

  Note Mirror Replication Agent does *not* support replicating multiple tables (or multiple columns in a table) with names that differ only in character case (for example, mytable and MyTable, or mycol and MYCOL).

- If the value of the use_rssd parameter is false and the value of the ltl_character_case parameter is asis, database object names in LTL are sent in the character case returned from the primary database.

- If the value of the use_rssd parameter is true and the value of the ltl_character_case parameter is asis, database object names in LTL are sent in the character case specified in the replication definitions (stored in the RSSD).

- If database object names are specified in all lowercase in replication definitions, set the value of the ltl_character_case parameter to lower.
- If database object names are specified in all uppercase in replication definitions, set the value of the ltl_character_case parameter to upper.
- If you want to send database object names with mixed character case (for example, MyTable), set the value of the ltl_character_case parameter to asis.

ltl_origin_time_required

Enables or disables the LTL origin_time command tag.

Default: false

Values: true – Enables the origin_time command tag in LTL generated by the Log Transfer Interface component.

false – Disables the origin_time command tag in LTL generated by the Log Transfer Interface component.

Comments:

- If the value of the ltl_origin_time_required parameter is true, the Log Transfer Interface component includes the origin_time command tag in the LTL it generates.
- If a Replication Server function string checks for the origin_time command tag, set the value of the ltl_origin_time_required parameter to true.
- The datetime value placed in the LTL origin_time command tag is the time when the original primary database operation was recorded in the transaction log, not the time it was scanned and processed by the Log Reader component.
- Setting the value of the ltl_origin_time_required parameter to false provides better Mirror Replication Agent throughput.
- If you use Replication Server Manager to report latency, you must set the value of the ltl_origin_time_required parameter to true.

ltm_admin_pw

The Mirror Replication Agent administrator login password.

Default: "" (empty string)

Value

A valid password.

Comments

- The value of the ltm_admin_pw parameter is the password for the user name authorized to log in to the Mirror Replication Agent.
- The value of the ltm_admin_pw parameter is encrypted in the Mirror Replication Agent configuration file.
- To change the value of the ltm_admin_pw parameter, use ra_set_login.
- When you change the value of the ltm_admin_pw parameter with ra_set_login, the new value is recorded in the configuration file immediately, but you must shut down and restart the Mirror Replication Agent instance to make the new password take effect.

  After you change the value of the ltm_admin_pw parameter with ra_set_login, you must use the new password next time you log in to the Mirror Replication Agent.

ltm_admin_user

The Mirror Replication Agent administrator login name.

Default

sa

Value

A valid user name on the Mirror Replication Agent host machine.

Comments

- The value of the ltm_admin_user parameter is the user name authorized to log in to the Mirror Replication Agent.
- To change the value of the ltm_admin_user parameter, use the ra_set_login command.
- If you change the value of the ltm_admin_user parameter with the ra_set_login command, the new value is recorded in the configuration file immediately, but you must shut down and restart the Mirror Replication Agent instance to make the new administrator name take effect.
- After you change the value of the ltm_admin_user parameter with ra_set_login, you must use the new administrator name next time you log in to the Mirror Replication Agent.
- Only one administrator name is valid at any time.

max_ops_per_scan

The maximum number of operations the Log Reader component reads during each log scan operation.

Default: 1000

Values: An integer from 25 to 2147483647.

Comments:

- The value of the max_ops_per_scan parameter is the maximum number of operations that can be read from the mirror log devices during each Log Reader scan operation (the size of the Log Reader operation queue).
- The Log Reader component always reads at least one transaction in each scan, regardless of how many operations are in the transaction.

  For example, if the value of the max_ops_per_scan parameter is 1000, and a transaction contains 1200 operations, the Log Reader component reads all 1200 operations in one scan when it reads that transaction.
- See “Configuring Mirror Replication Agent” on page 92 for information about how the max_ops_per_scan parameter affects Mirror Replication Agent performance.

pds_charset

The character set used in communication with the primary data server.

Default: "" (empty string)

Value: Any valid character set supported by the Java VM on the Mirror Replication Agent host machine.

Comments:

- When the Mirror Replication Agent instance is created, the pds_charset parameter is set to its default value "" (empty string).
- When you initialize the Mirror Replication Agent instance, the value of the pds_charset parameter is automatically set to match the primary Adaptive Server default character set, which is identified by the Adaptive Server default character set id configuration parameter.
- Sybase recommends that you:
  - Configure the primary and standby data servers and the Replication Server to all use the same character set
  - Do *not* change the value of the pds_charset parameter configured during Mirror Replication Agent initialization.

- The value of the pds_charset parameter must match (or be compatible with) the primary Adaptive Server default character set, which is identified by the Adaptive Server default character set id configuration parameter.
- If you specify a valid character set for the value of the pds_charset parameter, the Mirror Replication Agent instance reads replicated transaction data from the mirror log devices in that character set.
- If you do *not* specify a valid character set name for the value of the pds_charset parameter (including the default pds_charset value ""), the Mirror Replication Agent instance reads replicated transaction data from the mirror log devices using the system default character set, specified by the Java VM file.encoding system property on the Mirror Replication Agent host machine.

pds_connection_type

The type of connectivity driver used on the primary database connection.

Default	JCONN
Value	Sybase recommends that you do *not* change the default value of the pds_connection_type parameter.
Comment	The value of the pds_connection_type parameter is set automatically when a Mirror Replication Agent instance is created.

pds_database_name

The name of the primary database.

Default	<not_configured>
Value	A valid database name.
Comments	• The value of the pds_database_name parameter is the name of the primary database for which Mirror Replication Agent reads the mirror log devices. • See “Setting up the connection configuration parameters” on page 39 for more information.

pds_datasource_name

The data source name (DSN) specified for the ODBC driver used on the primary database connection.

Note Mirror Replication Agent does not use the pds_datasource_name parameter. This parameter is implemented for future use.

Default: <not_configured>

Value: A valid ODBC data source name.

Comment: The value of the pds_datasource_name parameter is the data source name (DSN) of the ODBC driver on the primary database connection.

pds_host_name

The name of the primary data server host machine.

Default: <not_configured>

Value: A valid host name.

Comments:

- The value of the pds_host_name parameter is the network name of the host machine on which the primary data server resides.
- See "Setting up the connection configuration parameters" on page 39 for more information.

pds_password

The password that Mirror Replication Agent uses for primary data server access.

Default: "" (empty string)

Value: A valid password.

Comments:

- The value of the pds_password parameter is the password for the user name that Mirror Replication Agent uses to access the primary data server.
- The value of the pds_password parameter is encrypted in the Mirror Replication Agent instance configuration file.

- See “Setting up the connection configuration parameters” on page 39 for more information.

pds_port_number

The client socket port number for the primary data server.

Default: 1111

Value: A valid port number on the primary data server host machine.

Comments:

- The value of the pds_port_number parameter is the client socket port number for the primary data server.
- See “Setting up the connection configuration parameters” on page 39 for more information.

pds_retry_count

The number of times to retry establishing a connection to the primary database.

Default: 5

Value: An integer from 0 to 2147483647.

Comments:

- The value of the pds_retry_count parameter is the number of times that Mirror Replication Agent will try to establish a network connection to the primary database after a connection failure.
- Sybase recommends a setting of 5 for this parameter.

pds_retry_timeout

The number of seconds to wait between retry attempts to connect to the primary database.

Default: 10

Value: An integer from 0 to 3600.

Comment: The value of the pds_retry_timeout parameter is the number of seconds that Mirror Replication Agent will wait between its retry attempts to establish a network connection to the primary database after a connection failure.

pds_server_name

The server name of the primary data server.

Note Mirror Replication Agent does not use the pds_server_name parameter. This parameter is implemented for future use.

Default: <not_configured>

Value: A valid server name.

Comment: The value of the pds_server_name parameter is the server name of the primary data server.

pds_username

The user name that Mirror Replication Agent uses for primary data server access.

Default: <not_configured>

Value: A valid user name.

Comments:

- The value of the pds_username parameter is the user login name that Mirror Replication Agent uses to log in to the primary data server.
- The user name that Mirror Replication Agent uses to log in to the primary data server must be defined in the primary data server, with appropriate privileges or authority in the primary database.
- Mirror Replication Agent uses this user name to access system databases and primary database objects.
- See “Setting up the connection configuration parameters” on page 39 for more information.

ra_retry_count

The number of times to attempt to restart replication after a failure.

Default: 2

Value: An integer greater than 0.

Comments

- The value of the ra_retry_count parameter is the number of times that the Log Transfer Manager component will try to get the Mirror Replication Agent instance back into *Replicating* state after a failure causes the instance to go to *Admin* state.
- When a network connection failure occurs, Mirror Replication Agent attempts to re-establish the lost connection. Mirror Replication Agent uses the pds_retry_count and pds_retry_timeout parameter settings, if the network connection to the primary database is lost. Mirror Replication Agent uses the rs_retry_count and rs_retry_timeout parameter settings, if the connection to the Replication Server is lost.

  If Mirror Replication Agent is unable to re-establish a lost connection after the number of retries specified in either the pds_retry_count or rs_retry_count parameter, then the Mirror Replication Agent instance goes to *Admin* state and the Log Transfer Manager component attempts to return the Mirror Replication Agent instance to *Replicating* state, based on the settings of the ra_retry_count and ra_retry_timeout parameters.

ra_retry_timeout

The number of seconds to wait between attempts to restart replication after a failure.

Default: 10

Value: An integer greater than 0.

Comment: The value of the ra_retry_timeout parameter is the number of seconds that the Log Transfer Manager component will wait between its attempts to get the Mirror Replication Agent instance back into *Replicating* state after a failure causes the instance to go to *Admin* state.

rasd_backup_dir

The directory path for RASD backup files.

Default: The path to the RASD *backup* directory created automatically when the Mirror Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

  `%SYBASE%\MRA-12_6\inst_name\repository\backup`

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:

  $SYBASE/MRA-12_6/*inst_name*/repository/backup

  where:

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

Value

A valid path on the Mirror Replication Agent host machine.

Comments

- When you create a Mirror Replication Agent instance, a RASD *backup* directory is created automatically as part of the instance directory structure. The default value of the rasd_backup_dir parameter points to that directory.
- If you specify any valid path as the value of the rasd_backup_dir parameter, Mirror Replication Agent places its RASD backup files in that directory during RASD backup operations, and it looks in that directory for the RASD backup files during restore operations.

rasd_database

The directory path for the RASD database file.

Default

The path to the RASD database file created automatically when the Mirror Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

  %SYBASE%\MRA-12_6*inst_name*\repository*inst_name*.db

  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:

  $SYBASE/MRA-12_6/*inst_name*/repository/*inst_name*.db

  where:

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

Value

A valid path and RASD database file name on the Mirror Replication Agent host machine.

Comments

- When you create a Mirror Replication Agent instance, the *repository* directory and the RASD database file are created automatically. The default value of the rasd_database parameter points to the RASD database file in that directory.
- If you specify any valid path and RASD database file name as the value of the rasd_database parameter, the Mirror Replication Agent instance looks in that directory for its RASD database file, with the file name you specified.

rasd_mirror_tran_log

Enables or disables RASD transaction log mirroring.

Default

false

Values

true – Enables mirroring the RASD transaction log to another file.

false – Disables mirroring of the RASD transaction log.

Comment

Setting the value of the rasd_mirror_tran_log parameter to true provides additional recovery options in the event of a device failure on the Mirror Replication Agent host machine.

rasd_trace_log_dir

The directory path for the RASD trace log file.

Default

The path to the *repository* directory created automatically when the Mirror Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

  %SYBASE%\MRA-12_6*inst_name*\repository

  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:

  $SYBASE/MRA-12_6/*inst_name*/repository

  where:

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

Value

A valid path on the Mirror Replication Agent host machine.

Comments

- When you create a Mirror Replication Agent instance, the *repository* directory is created automatically as part of the instance directory structure. The default value of the rasd_trace_log_dir parameter points to that directory.
- If you specify any valid path as the value of the rasd_trace_log_dir parameter, the Mirror Replication Agent instance writes its RASD trace log file in that directory.

rasd_tran_log

The directory path for the RASD transaction log file.

Default

The path to the RASD transaction log file created automatically when the Mirror Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

%SYBASE%\MRA-12_6*inst_name*\repository*inst_name*.log

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:

```
$SYBASE/MRA-12_6/inst_name/repository/inst_name.log
```

where:

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

Value

A valid path on the Mirror Replication Agent host machine.

Comments

- When you create a Mirror Replication Agent instance, the *repository* directory and RASD transaction log file are created automatically. The default value of the rasd_tran_log parameter points to that transaction log file.
- If you specify any valid path and RASD transaction log file name as the value of the rasd_tran_log parameter, the Mirror Replication Agent instance looks in that directory for its RASD transaction log file, with the name you specified.

rasd_tran_log_mirror

The directory path for the RASD transaction log file mirror.

Default

The path to the RASD transaction log file mirror in the *tran_log_mirror* directory created automatically when the Mirror Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

```
%SYBASE%\MRA-12_6\inst_name\repository\tran_log_mirror\inst_name.log
```

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:

```
$SYBASE/MRA-12_6/inst_name/repository/tran_log_mirror/inst_name.log
```

where:

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

- *inst_name* is the name of the Mirror Replication Agent instance.

Value

A valid path on the Mirror Replication Agent host machine.

Comment

If you specify any valid path and transaction log file name as the value of the rasd_tran_log_mirror parameter, the Mirror Replication Agent instance looks in that directory for its RASD transaction log file mirror, with the name you specified.

rs_charset

The character set used in communication with the Replication Server.

Default

"" (empty string)

Value

Any valid character set supported by the Java VM on the Mirror Replication Agent host machine.

Comments

- Sybase recommends that you:
  - Configure the primary and standby data servers and the Replication Server to all use the same character set
  - Use the default rs_charset value "" (empty string)

  When you initialize the Mirror Replication Agent instance, the value of the pds_charset parameter is automatically set to match the primary Adaptive Server default character set, which is identified by the Adaptive Server default character set id configuration parameter.

  Using the default rs_charset value "" with a valid pds_charset value is equivalent to setting both parameters to the same value, and it has a distinct benefit; it reduces the opportunity for human error (such as entering an incorrect value for the rs_charset parameter).
- The value of the rs_charset parameter must match (or be compatible with) the Replication Server default character set, which is identified by the Replication Server RS_charset configuration parameter.
- If you specify a valid character set for the value of the rs_charset parameter, the Mirror Replication Agent instance sends replicated transaction data from the primary database to the Replication Server in that character set.
- If you do *not* specify a valid character set name for the value of the rs_charset parameter (including the default rs_charset value ""), the Mirror

Replication Agent instance sends replicated transaction data to the Replication Server using either:

- The primary database character set, specified in the Mirror Replication Agent pds_charset parameter, if the pds_charset value is a valid character set name, or
- The system default character set, specified by the Java VM file.encoding system property on the Mirror Replication Agent host machine, if the pds_charset value is not a valid character set name (including the default pds_charset value "").

- If the values of the rs_charset and pds_charset parameters are valid, but *not* the same value, Mirror Replication Agent converts the replicated transaction data from the primary database character set (pds_charset) to the Replication Server character set (rs_charset) before sending it to the Replication Server.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rs_host_name

The name of the Replication Server host machine.

Default: <not_configured>

Value: A valid host name.

Comments:

- The value of the rs_host_name parameter is the name of the host machine for the Replication Server.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rs_packet_size

The network packet size on the connection to the Replication Server.

Default: 2048

Value: An integer from 512 to 8192.

Comments:

- The value of the rs_packet_size parameter is the maximum size (in bytes) of the network packets handled by the TCP/IP network protocol.

- The Mirror Replication Agent rs_packet_size parameter is equivalent to the Replication Server rsi_packet_size parameter.
- When the network packet size is smaller, more packets must be processed to transmit a given amount of data to the Replication Server. When the network packet size is larger, more system resources are consumed to process the packets.
- The optimum value of the rs_packet_size parameter is based on the nature of the typical data replicated. If the typical operation is very large, a larger packet size is more efficient.
- A larger value of the rs_packet_size parameter is more efficient when the value of the lti_batch_mode parameter is true.

rs_password

The password that Mirror Replication Agent uses for Replication Server access.

Default: "" (empty string)

Value: A valid password.

Comments:

- The value of the rs_password parameter is the password for the user name that Mirror Replication Agent uses to log in to the Replication Server.
- The value of the rs_password parameter is encrypted in the Mirror Replication Agent instance configuration file.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rs_port_number

The client socket port number of the Replication Server.

Default: 1111

Value: A valid port number on the Replication Server host machine.

Comments:

- The value of the rs_port_number parameter is the client socket port number of the Replication Server.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rs_retry_count

The number of times to retry establishing a connection to the Replication Server.

Default: 5

Value: An integer greater than 0.

Comments:

- The value of the rs_retry_count parameter is the number of times that Mirror Replication Agent will try to establish a network connection to the Replication Server after a connection failure.
- Sybase recommends a setting of 5 for this parameter.

rs_retry_timeout

The number of seconds to wait between retry attempts to connect to the Replication Server.

Default: 10

Value: An integer greater than 0.

Comment: The value of the rs_retry_timeout parameter is the number of seconds that Mirror Replication Agent will wait between its retry attempts to establish a network connection to the Replication Server after a connection failure.

rs_source_db

The name of the database identified in the Replication Server primary database connection.

Default: <not_configured>

Value: A valid database name.

Comments:

- The value of the rs_source_db parameter is the name of the primary database by which the Replication Server recognizes the primary database transaction log.
- The value of the rs_source_db parameter must match the name of the database specified in the Replication Server create connection command for the primary database.

- See “Setting up the connection configuration parameters” on page 39 for more information.

rs_source_ds

The name of the data server identified in the Replication Server primary database connection.

Default: <not_configured>

Value: A valid server name.

Comments:

- The value of the rs_source_ds parameter is the name of the primary data server by which the Replication Server recognizes the primary database transaction log.
- The value of the rs_source_ds parameter must match the name of the data server specified in the Replication Server create connection command for the primary database.
- The value of the rs_source_ds parameter should *not* be the same as the name of the Mirror Replication Agent instance.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rs_username

The user name that Mirror Replication Agent uses for Replication Server access.

Default: <not_configured>

Value: A valid user name.

Comments:

- The value of the rs_username parameter is the user login name that Mirror Replication Agent uses to log in to the Replication Server.
- The value of the rs_password parameter is the password for the user name specified by the rs_username parameter.
- The user name that Mirror Replication Agent uses to log in to the Replication Server must have connect source permission in the Replication Server.

- See “Setting up the connection configuration parameters” on page 39 for more information.

rssd_charset

The character set used in communication with the RSSD.

Default: "" (empty string)

Value: Any valid character set supported by the Java VM on the Mirror Replication Agent host machine.

Comments:

- The value of the rssd_charset parameter must match (or be compatible with) the RSSD character set. The RSSD character set is usually the same as the Replication Server default character set identified by the Replication Server RS_charset configuration parameter.
- If you specify a valid character set for the value of the rssd_charset parameter, the Mirror Replication Agent instance communicates with the RSSD using that character set.
- If you do *not* specify a valid character set name for the value of the rssd_charset parameter (including the default rssd_charset value ""), the Mirror Replication Agent communicates with the RSSD using the system default character set, specified by the Java VM file.encoding system property on the Mirror Replication Agent host machine.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rssd_database_name

The database name of the RSSD.

Default: <not_configured>

Value: A valid database name.

Comments:

- The value of the rssd_database_name parameter is the database name of the RSSD.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rssd_host_name

The name of the RSSD host machine.

Default: <not_configured>

Value: A valid host name.

Comments:

- The value of the rssd_host_name parameter is the name of the host machine on which the RSSD resides.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rssd_password

The password that Mirror Replication Agent uses for RSSD access.

Default: "" (empty string)

Value: A valid password.

Comments:

- The value of the rssd_password parameter is the password for the user login name that Mirror Replication Agent uses to access the RSSD.
- The value of the rssd_password parameter is encrypted in the Mirror Replication Agent instance configuration file.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rssd_port_number

The client socket port number of the RSSD.

Default: 1111

Value: A valid port number on the RSSD host machine.

Comments:

- The value of the rssd_port_number parameter is the client socket port number of the RSSD data server.
- See “Setting up the connection configuration parameters” on page 39 for more information.

rssd_username

The user name that Mirror Replication Agent uses for RSSD access.

Default: <not_configured>

Value: A valid user login name in the RSSD data server.

Comments:

- The value of the rssd_username parameter is the user login name that Mirror Replication Agent uses to access the RSSD.
- See “Setting up the connection configuration parameters” on page 39 for more information.

scan_sleep_increment

The number of seconds to add to each wait before scanning the transaction log, after a previous scan yields no transaction to be replicated.

Default: 5

Value: An integer from 0 to 3600.

Comments:

- The value of the scan_sleep_increment parameter is the number of seconds added to each wait before the Log Reader component scans the mirror log device for a transaction to be replicated, after a previous scan yields no such transaction.
- The number of seconds specified by the scan_sleep_increment parameter is added to each wait until the wait time reaches the value specified by the scan_sleep_max parameter.
- For optimum Mirror Replication Agent performance, the value of the scan_sleep_increment parameter should be balanced with the average number of operations in the primary database over a period of time. In general, better performance results from reading more operations from the mirror log devices during each Log Reader scan.
- With a primary database that is infrequently updated, increasing the value of the scan_sleep_increment parameter may improve overall performance.
- If the database is continuously updated, the value of the scan_sleep_increment parameter may not be significant to Mirror Replication Agent performance.

scan_sleep_max

The maximum number of seconds to wait between transaction log scans.

Default: 60

Value: An integer from 5 to 86400.

Comments:

- The value of the scan_sleep_max parameter is the maximum number of seconds that can elapse before the Log Reader component scans the mirror log device for a transaction to be replicated, after a previous scan yields no such transaction.
- For less replication latency in an infrequently updated database, Sybase recommends lower number settings for the scan_sleep_max parameter.
- If the primary database is continuously updated, the value of the scan_sleep_max parameter may not be significant to Mirror Replication Agent performance.

skip_ltl_errors

Determines whether LTL error messages are ignored.

Warning! Using the skip_ltl_errors parameter incorrectly may cause data inconsistencies between the primary and standby databases.

Default: false

Values: true – Enables skipping LTL errors to continue replication.

false – Disables skipping LTL errors.

Comments:

- If the skip_ltl_errors configuration parameter is set to true, Mirror Replication Agent logs any LTL error message returned by Replication Server, along with the offending LTL command(s), and then it continues processing log records.
- If the skip_ltl_errors configuration parameter is set to false, Mirror Replication Agent stops replication and goes to *Admin* state when it receives an LTL error message and the error is unrecoverable.
- The skip_ltl_errors parameter is intended for troubleshooting only, with assistance from Sybase Technical Support.

structured_tokens

Determines whether to use LTL structured tokens.

Default: true

Values: true – Enables LTL structured tokens.

false – Disables LTL structured tokens.

Comments:

- If the structured_tokens configuration parameter is set to true, the Log Transfer Interface component uses LTL structured tokens when it generates LTL commands.
- Using structured tokens in the LTL can significantly improve replication system performance.
- To replicate columns that have one or more spaces in the column name, you must set the value of the structured_tokens parameter to true.

use_rssd

Determines whether Mirror Replication Agent uses replication definitions.

Default: true

Values: true – Enables using replication definitions.

false – Disables using replication definitions.

Comments:

- If the value of the use_rssd parameter is true, Mirror Replication Agent connects to the RSSD to retrieve replication definitions for the primary database whenever the resume command is invoked.
  - Each time it retrieves replication definitions, Mirror Replication Agent stores the information in a cache. Mirror Replication Agent uses replication definitions stored in its cache when it generates LTL commands.
  - If the Log Transfer Interface component encounters an operation on a database object for which it does not have a cached replication definition, Mirror Replication Agent reconnects to the RSSD to update its replication definition cache.
  - If a replication definition still cannot be found for the operation, the Mirror Replication Agent instance suspends operation and goes to the *Admin* state, unless it is in a warm standby application.

Note When Mirror Replication Agent is used in a Replication Server warm standby application, it will *not* suspend replication when it cannot find a replication definition, regardless of the value of the use_rssd parameter.

- Mirror Replication Agent uses information in table and function replication definitions (that is, replication definitions for individual primary database objects) stored in the RSSD to generate more efficient LTL, and thus improve throughput in the Log Transfer Interface component, and throughout the replication system.

  Accessing replication definitions in the RSSD enables the Log Transfer Interface component to improve performance by:

  - Omitting column names in LTL.

    When columns are sent in the order specified in the replication definition, column images can be sent without column names (headings), which reduces LTL overhead.

  - Omitting unneeded columns in LTL.

    When columns are sent as specified in the replication definition, images for unchanged columns need not be sent, which reduces LTL overhead.

  - Sending data for each column in the datatype specified by the replication definition.

    This allows data to be handled more efficiently all the way through the replication system.

  - Sending database object names in the same character case as defined in the replication definition.

- If the value of the use_rssd parameter is false, or no replication definition exists for the table, column data is sent in the datatype defined in the primary database.
- If you use owner-qualified table names for primary tables, you must:
  - Set the value of the use_rssd parameter to true
  - Specify an owner-qualified table name when you create the replication definition
  - Mark the table in the primary database using the owner_on clause of the sp_setreptable procedure

CHAPTER 6

Troubleshooting Mirror Replication Agent

This chapter describes basic troubleshooting procedures for Mirror Replication Agent and the Mirror Activator system.

Basic Mirror Replication Agent troubleshooting

This section contains basic procedures you can use to troubleshoot Mirror Replication Agent problems.

The following topics are in this section:

- Server name resolution
- Duplicate column names
- Changed or missing replication definitions
- File handle problems
- Missing or damaged configuration file
- Mirror Replication Agent initialization problems

Server name resolution

If you experience problems connecting Mirror Replication Agent with other servers in a Mirror Activator system because the server name cannot be resolved, try using the server's actual IP address.

Duplicate column names

Tables that contain two (or more) columns that have the same name, except for character case (for example, mycol and MyCol) cannot be replicated with Mirror Replication Agent. Columns identified in a replication definition are also subject to the same limitation.

Changed or missing replication definitions

If the value of the use_rssd configuration parameter is true, Mirror Replication Agent connects to the RSSD to retrieve replication definitions for the primary database whenever the resume command is invoked.

Note In this section, the term *replication definition* refers to both table replication definitions and function replication definitions.

Each time it retrieves replication definitions, Mirror Replication Agent stores that information in a cache. Mirror Replication Agent uses the replication definitions stored in its cache when generating LTL to send to the Replication Server.

If the Log Transfer Interface component encounters an operation for a database object that it does not have cached information for, Mirror Replication Agent reconnects to the RSSD to update its replication definition cache. If a replication definition cannot be found for the operation, the Log Transfer Interface generates unoptimized LTL for that operation, without the benefit of the replication definition, and continues operating normally.

If you drop a replication definition after Mirror Replication Agent has cached the RSSD information, a problem may occur. If the replication definition is dropped and then re-created, the Replication Server may suspend the Mirror Replication Agent connection when it receives LTL that is not formatted correctly for the object, as specified in the current replication definition in the RSSD.

To avoid this problem, you must:

1 Quiesce the Mirror Replication Agent instance.

2 Change the replication definition in Replication Server.

3 Resume the Mirror Replication Agent so that it has the correct version of the replication definition in its cache.

File handle problems

A connection failure error can occur when there is an insufficient number of file handles available to Mirror Replication Agent.

The following error message may indicate this problem:

```
java.lang.NoClassDefFoundError: ... ConnectFailedEvent
at ... FailedToConnect
```

If this error occurs while Mirror Replication Agent is otherwise functioning normally (that is, the entire Mirror Activator system seems to be working properly) and you cannot find another cause for the connection failure, it may indicate a file handles problem.

To correct this problem, increase the number of file handles available to Mirror Replication Agent.

Missing or damaged configuration file

If the Mirror Replication Agent configuration file is deleted or damaged, the Mirror Replication Agent instance cannot start correctly, and it will immediately shut down.

When the Mirror Replication Agent configuration file is deleted or damaged, you have two options for recovery:

- Copy the default configuration file to the Mirror Replication Agent instance directory and then re-create the missing configuration items.
- Delete the Mirror Replication Agent instance, create a new instance, and then go through the setup process to re-create the configuration.

Note You can use the information recorded on the "Installation and Setup Worksheet" in the Mirror Replication Agent *Installation Guide* to re-create a Mirror Replication Agent configuration file.

Use the following procedure to recover from a damaged or missing Mirror Replication Agent configuration file by copying the default configuration and re-creating the missing items.

❖ **To recover from a missing Mirror Replication Agent configuration file**

1 Make sure the Mirror Replication Agent instance is shut down. If it is not, use the shutdown command to terminate the Mirror Replication Agent instance.

2 Copy the default Mirror Replication Agent configuration file (such as *mra.cfg*) from the *MRA-12_6\config* directory to the Mirror Replication Agent instance directory.

For example, if the Mirror Replication Agent instance is named "my_mra," you would use the following command at the operating system prompt in the Mirror Replication Agent installation directory to copy the default Mirror Replication Agent configuration file:

```
copy config\mra.cfg my_mra\my_mra.cfg
```

3 Start the Mirror Replication Agent instance using the following command at the operating system prompt:

```
mra -i inst_name
```

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

4 Log in to the Mirror Replication Agent administration port using the following command at the operating system prompt:

```
isql -Usa -P -Sinst_name
```

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

5 Use the ra_config command to set the log directory for the Mirror Replication Agent instance:

```
ra_config log_directory, logpath
```

where *logpath* is the full path specification.

The following example is the path specified automatically when the Mirror Replication Agent instance is created:

```
C:%SYBASE%\MRA-12_6\inst_name\log\
```

where:

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

6 Use the ra_set_login command to reset the administrative user ID and password for the Mirror Replication Agent instance:

```
ra_set_login mra_admin_user, mra_admin_pwd
```

where:

- *mra_admin_user* is the Mirror Replication Agent administrator login.
- *mra_admin_pwd* is the corresponding password.

7 Use the ra_config command to set the values of the following Mirror Replication Agent configuration parameters for the primary database:

pds_database_name
pds_datasource_name
pds_host_name
pds_password
pds_port_number
pds_server_name
pds_username

8 Use the ra_config command to set the values of the following Mirror Replication Agent configuration parameters for the Replication Server:

rs_host_name
rs_password
rs_port_number
rs_source_db
rs_source_ds
rs_username

9 Use the ra_config command to set the values of the following Mirror Replication Agent configuration parameters for the RSSD:

rssd_database_name
rssd_host_name
rssd_password
rssd_port_number
rssd_username

10 Use the shutdown command to shut down the Mirror Replication Agent instance so you can re-start the Mirror Replication Agent instance with the correct administrative user login:

```
shutdown
```

11 Start the Mirror Replication Agent instance using the following command at the operating system prompt:

```
mra -i inst_name
```

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

12 Log in to the Mirror Replication Agent administration port using the following command at the operating system prompt:

```
isql -Umra_admin_user -Pmra_admin_pwd -Sinst_name
```

where:

- *mra_admin_user* is the Mirror Replication Agent instance administrator login.
- *mra_admin_pwd* is the password.
- *inst_name* is the name of the Mirror Replication Agent instance.

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

```
ra_status
```

14 Use the test_connection command to verify that the Mirror Replication Agent instance can communicate with the primary data server, the Replication Server, and the RSSD:

```
test_connection
```

15 Use the resume command to put the Mirror Replication Agent instance into *Replicating* state:

```
resume
```

16 Use the ra_status command to verify that the Mirror Replication Agent instance is in *Replicating* state:

```
ra_status
```

If the Mirror Replication Agent instance successfully enters *Replicating* state, then the information stored in the configuration file is correct for that Mirror Replication Agent instance.

If the Mirror Replication Agent instance remained in *Admin* state after you invoked the resume command, then you still need to verify and correct the following items:

- Configuration parameters for the primary database (see step 7 in the previous procedure)

- Configuration parameters for the Replication Server (see step 8 in the previous procedure)
- Configuration parameters for the RSSD (see step 9 in the previous procedure)

Mirror Replication Agent initialization problems

When you initialize the Mirror Replication Agent instance using ra_init, Mirror Replication Agent executes the Adaptive Server sp_special_columns procedure in the primary database, while the database is quiesced, to get the information it needs for its system data repository.

Therefore, the metadata caches for open indexes and open objects in the primary Adaptive Server must be large enough to execute sp_special_columns against each table marked for replication in the primary database. The default value of the number of open indexes and number of open objects configuration parameters is 500.

If metadata caches in the primary Adaptive Server are not large enough, the execution of ra_init may be blocked in the Mirror Replication Agent until the primary database is unquiesced (update activity is resumed).

To avoid this problem, increase the values of the number of open indexes and number of open objects configuration parameters in the primary Adaptive Server to accommodate the number of tables marked for replication in the primary database, *before* you initialize the Mirror Replication Agent instance.

After the Mirror Replication Agent instance is initialized, you can restore the normal number of open indexes and number of open objects parameter values in the primary Adaptive Server.

Basic Replication Server troubleshooting

This section contains basic procedures you can use to check the operation of Replication Server or troubleshoot Replication Server problems.

For more information, see the Replication Server *Troubleshooting Guide*.

Verify that Replication Server is running

Verify that the Replication Server is running:

- Use the Replication Server admin who_is_up command to find out which servers and processes are running.
- Use the Replication Server admin who_is_down command to find out which servers are down.

Check the Replication Server log

Check the Replication Server log for error and warning messages. The most recent messages are at the end of the log.

Check the 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.

See the Replication Server *Reference Manual* for information about the admin who command.

Correct the origin queue ID

If insert or update operations in the primary database are not being replicated, and if you do not find any errors in the Mirror Replication Agent system logs, the most likely cause of the problem is that the value of the origin queue ID (*qid*) sent to the Replication Server is lower than previous values of the *qid*. In that case, the new operations are ignored by Replication Server.

❖ **To correct the problem of incorrect *qid* values**

1 Turn on tracing for the Mirror Replication Agent LTITRACELTL trace flag.

You can view Mirror Replication Agent LTL output by examining the contents of the *LTITRACELTL.log* file.

2 Log in to the Replication Server and use the admin who, sqm command to determine if any duplicate operations are in the inbound queue of the Replication Server.

If you find duplicate operations in the inbound queue of the Replication Server, then Mirror Replication Agent is re-sending operations to the Replication Server as part of a recovery process. Finding duplicate operations indicates that the value of the Mirror Replication Agent origin queue ID is not the problem.

If you find no duplicate operations in the inbound queue of the Replication Server, continue this procedure to reset the *qid* value.

3 Log in to the Mirror Replication Agent administration port.

4 Use the Mirror Replication Agent quiesce command to put the Mirror Replication Agent instance in *Admin* state.

```
quiesce
```

5 Use the Mirror Replication Agent ra_locator command to set the value of the LTM Locator stored in the RASD to all zeros.

```
ra_locator zero
```

6 Log in to the RSSD and execute the rs_zeroltm stored procedure.

```
rs_zeroltm pds, pdb
```

where:

- *pds* is the name of the primary data server.
- *pdb* is the name of the primary database, as specified in the Replication Server create connection command.

7 Restart replication using the Mirror Replication Agent resume command.

```
resume
```

When the value of the LTM Locator is set to all zeros in both the RASD and the RSSD, Mirror Replication Agent starts scanning the transaction log at the end of the log.

Replication failure troubleshooting

This section contains basic procedures you can use to diagnose and troubleshoot replication system failures.

Diagnostic procedures and tips

Use the procedures in the following sections to diagnose the source of replication system failures:

- Verify database names in Replication Server
- Verify the Mirror Replication Agent login in Replication Server
- Check the Mirror Replication Agent logs
- Check the standby database
- Check the error logs of the data servers and RSSD

Verify database names in Replication Server

Check the Replication Server log to verify that the values for *data_server* and *database* specified in the create connection command exactly match the values specified for the Replication Server RS_source_ds and RS_source_db configuration parameters, and that they exactly match the values specified for the Mirror Replication Agent rs_source_ds and rs_source_db configuration parameters.

Use the Replication Server admin who command to look for the Mirror Replication Agent instance in the list of replication system servers.

Verify the Mirror Replication Agent login in Replication Server

Executing the Replication Server connect source lti command accomplishes three things:

- Verifies that the database connection from the Mirror Replication Agent instance to the Replication Server exists
- 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 displays 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 value of the rs_username configuration parameter.

Refer to the "Installation and Setup Worksheet" in the Mirror Replication Agent *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 you specified for the Mirror Replication Agent rs_source_ds configuration parameter.
- *pdb* is the value you 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 Replication Agent *Installation Guide* for the values of the rs_source_ds and rs_source_db parameters.

For the value of the LTL version number, you can enter `999`, and the Replication Server will return the highest numbered version of LTL that it supports.

3 Disconnect from the Replication Server as rs_username, and then invoke the Mirror Replication Agent resume command.

See the Replication Server *Design Guide* and *Reference Manual* for more information about the connect source lti command.

Check 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 Replication Server and the primary database.

Check the standby database

Use the Replication Server admin who_is_down command to determine whether the connection to the standby database is down, or has been suspended by Replication Server.

Check the error logs of the data servers and RSSD

The Adaptive Server error logs may contain warning and error messages.

Troubleshooting procedures

Use the procedures in the following sections to troubleshoot replication system failures.

Replication fails to start after resuming standby connection

When replication fails to start after you resume the standby connection in Replication Server, there are two probable causes:

- The standby DSI thread does not recognize the “enable replication” marker.
- The secondary truncation point was moved while there was an open transaction in the stable queue.

Problem with enable replication marker

If the Replication Server standby database connection is resumed before the Mirror Replication Agent is resumed, the standby DSI thread may not recognize the “enable replication” marker in the inbound queue. In that event, the standby DSI continues to ignore messages in the inbound queue, waiting for the marker to appear. The following symptoms indicate this problem:

- Mirror Replication Agent processes the transaction log and sends LTL to the Replication Server for distribution, but Replication Server sends no transactions to the standby database after the standby connection is resumed.
- The admin logical_status command reports the following standby connection state:

  ```
  Active/Waiting for Enable Marker
  ```

- The admin who, dsi command reports the following standby DSI Ignoring Status:

  ```
  Ignoring
  ```

- The standby database row in the RSSD rs_databases table contains a value of 0x8 in the dist_status column, and a non-zero value in the enable_seq column.

Note To avoid this problem, *always* resume replication in the Mirror Replication Agent *before* you resume the Replication Server standby connection.

❖ **To correct the enable replication marker problem**

1 Suspend the standby connection in Replication Server.

2 Suspend the Mirror Replication Agent.

3 Purge all open transactions in the Replication Server stable queue.

See the Replication Server *Reference Manual* for information about the sysadmin purge_all_open and sysadmin purge_first_open commands.

4 Resume the Mirror Replication Agent.

5 Resume the standby connection in Replication Server.

Problem with truncation point moved

If the secondary truncation point moves while an open transaction remains in the Replication Server stable (inbound) queue, no transactions are sent to the standby database. When this occurs:

- Replication will not start, because Replication Server will not send any new transactions to the standby database until it receives operations to complete an open transaction.
- Eventually, the stable queue will run out of space, because Replication Server will not send any transactions after you resume replication, and

new transactions arriving in the stable queue will not be processed and deleted.

This problem may occur under the following conditions:

- You suspend the Mirror Replication Agent after a transaction begin record is processed, but before the corresponding commit or abort record is processed, and then move the secondary truncation point (for example, using pdb_init move_truncpt).
- The primary database goes offline after a transaction begin record is written to the mirror log device, but before the corresponding commit or abort record is written, triggering a failover to the standby database and requiring the primary database to be re-materialized.

Note To avoid this problem, do *not* move the secondary truncation point while an open transaction remains in the stable queue.

If you need to re-initialize the primary database, in preparation for re-initializing the Mirror Replication Agent, do *not* move the secondary truncation point.

❖ **To correct the moved truncation point problem**

- Purge all open transactions in the stable queue *before* resuming the Replication Server standby connection.

  See the Replication Server *Reference Manual* for information about the sysadmin purge_all_open and sysadmin purge_first_open commands.

Note If the secondary truncation point must be moved (for example, when the primary database is re-materialized in a failback procedure), you must purge all open transactions in the stable queue *before* resuming the standby connection.

APPENDIX A

Materializing Databases

This appendix describes how to materialize the databases in a Mirror Activator system.

The procedures in this appendix assume that:

- You have already installed the Mirror Replication Agent software and Replication Server software, as described in the Mirror Replication Agent *Installation Guide* and the Replication Server installation and configuration guides for your platform.
- You have an existing, operational Adaptive Server Enterprise (ASE) database, and you have configured that database as the primary database in a Mirror Activator system, as described in Chapter 2, "Setup and Configuration."

Materialization options

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 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, the rs_subcmp utility, or 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 system.

The Mirror Activator 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 system relies on mirror log devices, which are exact copies of the primary database log devices, a device-level database materialization technique works best with the Mirror Activator system.

Although there are other techniques you could use to materialize a database, Sybase recommends *snapshot materialization*, using the disk replication system to capture a snapshot (or point-in-time copy) of all source database data and log devices, and then send the snapshot to corresponding devices at the target database site.

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.

The Mirror Activator system setup procedures in Chapter 2, "Setup and Configuration," and the failover and failback procedures in Appendix B, "Failover and Failback with Mirror Activator," are based on using the snapshot materialization technique.

Note Snapshot materialization requires the primary database to be quiesced. To minimize primary database downtime, refer to the appropriate task checklist in the following sections and plan your materialization procedures carefully.

Materializing databases in ASE 12.5.0.3

This section describes two materialization procedures for Mirror Activator databases in Adaptive Server version 12.5.0.3:

- Materializing the standby database (required when the Mirror Activator system is set up)
- Re-materializing the primary database for failback

Both of these procedures use the snapshot materialization technique, described in "Materialization options" on page 203.

You can use the following procedures to materialize a database in any supported version of Adaptive Server (version 12.5.0.3, or version 12.5.1 or later). However:

- If the databases in the Mirror Activator system reside in Adaptive Server version 12.5.0.3, you must use the procedures in this section.
- If the databases in the Mirror Activator system reside in Adaptive Server version 12.5.1 or later, you can use either the procedures in this section, or the procedures in "Materializing databases in ASE 12.5.1 or later" on page 211.

Materializing the standby database in ASE 12.5.0.3

Materializing the standby database is one of the tasks required to set up the Mirror Activator system. Therefore, the following Mirror Activator system setup tasks (which are not strictly materialization tasks) must be performed during the standby database materialization procedure:

- Materialize the mirror log devices at the standby site, and set up the disk replication system for synchronous replication to the mirror log devices
- Initialize the Mirror Replication Agent

The snapshot materialization procedure mentions these setup tasks, but it does not describe them in detail. See "Setting up a new Mirror Activator system" on page 48 for detailed information about Mirror Activator system setup tasks.

Note *Before* you materialize a standby database, you must create the Replication Server database objects in the primary database, and then initialize the primary database using the Mirror Replication Agent pdb_init command. See "Setting up a new Mirror Activator system" on page 48 for more information.

Table A-1 provides a checklist of the snapshot materialization tasks for a standby database in Adaptive Server version 12.5.0.3.

The checklist in Table A-1 assumes that the standby Adaptive Server is already installed, and configured identically to the primary Adaptive Server.

Table A-1: Materializing a standby database in ASE 12.5.0.3

Task	Description
1	In the standby Adaptive Server, create the same server logins and roles that are defined in the primary Adaptive Server.
2	In the standby Adaptive Server, create an empty (or shell) database, with the same data and log devices, database name, and database options as the primary database.
3	Shut down the standby Adaptive Server.
4	Quiesce the primary database to suspend all update activity.
5	Use the disk replication system to copy a snapshot of all primary database data and log devices to the standby database devices. While the primary database is suspended during Mirror Activator system setup, you must also: • Copy a snapshot of the primary database log devices to the mirror log devices, and configure synchronous replication from the primary log devices to the mirror log devices • Initialize the Mirror Replication Agent
6	Resume update activity in the primary database after all procedures in step 5 are complete.
7	Start the standby Adaptive Server to recover the standby database and bring it online.

Use the following procedure for snapshot materialization of a standby database in Adaptive Server version 12.5.0.3.

❖ **To materialize a standby database in ASE 12.5.0.3**

1 In the standby Adaptive Server, create all server logins and roles defined in the primary Adaptive Server.

2 Create an empty (or shell) database in the standby Adaptive Server.

a Initialize the standby database data and log devices in the standby Adaptive Server, using the same disk init options as the primary database devices in the primary Adaptive Server. For example:

```
disk init name = "pdb_data",
physname = "data_dev",
size = "dddM"
```

where:

- *pdb_data* is the device name used in the primary Adaptive Server.
- *data_dev* is the path to the device in the standby Adaptive Server.

- *ddd* is the size of the device, in megabytes, in the primary Adaptive Server.

Note The only device options that can be different between the primary and standby servers are the paths to the physical devices. All other device options must be identical on the primary and standby servers.

b Create the standby database in the standby Adaptive Server, using the same create database options as the database in the primary Adaptive Server. For example:

```
create database pdb
on pdb_data = "dddM"
log on pdb_log = "lllM"
```

where:

- *pdb* is the database name used in the primary Adaptive Server.
- *pdb_data* is the database data device name used in the primary Adaptive Server.
- *ddd* is the size of the database data device, in megabytes, in the primary Adaptive Server.
- *pdb_log* is the database log device name used in the primary Adaptive Server.
- *lll* is the size of the database log device, in megabytes, in the primary Adaptive Server.

3 Shut down the standby Adaptive Server using the shutdown command.

4 Quiesce the primary database to suspend all update activity.

Log in to the primary Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup hold pdb
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *pdb* is the name of the primary database.

5 Using the disk replication system facilities:

- Capture a snapshot (or point-in-time) image of all of the primary database data and log devices

- Transfer the snapshot to the standby database devices you initialized in the standby Adaptive Server

While the primary database is suspended during Mirror Activator system setup, you must also:

- Transfer the snapshot of the primary database log devices to the mirror log devices, and configure the disk replication system for synchronous replication from the primary log devices to the mirror log devices
- Initialize the Mirror Replication Agent, using the ra_init command

  See "Setting up a new Mirror Activator system" on page 48 for more information about initializing the Mirror Replication Agent.

Refer to the documentation provided by your disk replication system vendor for more information about the procedures in this step.

6 Resume update activity in the primary database after all procedures in step 5 are complete.

Note You must initialize the Mirror Replication Agent *before* you resume update activity in the primary database.

Log in to the primary Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup release
```

where *MA_setup* is a user-defined tag that identifies the suspended primary database.

7 Start the standby Adaptive Server to recover the standby database and bring it online.

Open an operating system command prompt window on the standby Adaptive Server host, and use the RUN script to start the standby Adaptive Server:

```
$SYBASE/ASE_12-5/install/RUN_sds
```

where:

- *$SYBASE* is the path to the Adaptive Server installation directory.
- *sds* is the name of the standby Adaptive Server.

The standby data server will start up in "recovery" mode, and it may take a while for the standby database to come online.

Re-materializing the primary database for failback in ASE 12.5.0.3

Normally, the primary database is the source of all data and transactions replicated in the Mirror Activator system. Re-materializing the primary database is required only as part of a failback procedure, to restore normal system operations after a failover.

After a failover event, the standby database becomes the "active" database, and the primary database must be re-materialized from the standby database to restore the normal operating condition, and resume replication from the primary database to the standby database.

Note When you re-materialize a primary database, the primary database is the *target* and the standby database is the *source* in the materialization procedure.

Table A-2 provides a checklist of the snapshot re-materialization tasks for a primary (target) database in Adaptive Server version 12.5.0.3.

The checklist in Table A-2 assumes that:

- The primary (target) Adaptive Server is already installed, and configured identically to the standby (source) Adaptive Server.
- The primary (target) database exists in the primary (target) Adaptive Server, and its database options and devices are configured identically to the standby (source) database.
- All server logins defined in the standby (source) Adaptive Server exist in the primary (target) Adaptive Server, with identical suid values and names, and all server roles defined in the standby (source) Adaptive Server exist in the primary (target) Adaptive Server.

Table A-2: Re-materializing a primary database in ASE 12.5.0.3

Task	Description
1	Shut down the primary (target) Adaptive Server.
2	Quiesce the standby (source) database to suspend all update activity. The standby (source) database will remain suspended during failback, until after the materialization procedure is complete for the primary (target) database.
3	Use the disk replication system to copy a snapshot of all standby (source) database data and log devices to the primary (target) database devices.
4	Start the primary (target) Adaptive Server to recover the primary (target) database and bring it online.

After you complete the primary database re-materialization, you must perform additional tasks to complete the failback procedure. See Appendix B, "Failover and Failback with Mirror Activator," for more information about failback procedures.

Use the following procedure for snapshot re-materialization of a primary database in Adaptive Server version 12.5.0.3.

❖ **To re-materialize a primary database in ASE 12.5.0.3**

1 Shut down the primary (target) Adaptive Server using the shutdown command.

2 Quiesce the standby (source) database to suspend all update activity.

Log in to the standby (source) Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup hold sdb
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *sdb* is the name of the standby (source) database.

Note The standby (source) database will remain suspended during failback, until after the materialization procedure is complete for the primary (target) database.

3 Using the disk replication system facilities:

- Capture a snapshot (or point-in-time) image of all of the standby (source) database data and log devices
- Transfer the snapshot to the primary (target) database devices you initialized in the primary (target) Adaptive Server

While the standby (source) database is suspended during failback, you must also:

- Transfer the snapshot of the standby (source) database log devices to the mirror log devices
- Configure the disk replication system for synchronous replication from the primary (target) log devices to the mirror log devices (to prepare for normal system operation after failback)

Refer to the documentation provided by your disk replication system vendor for more information about the procedures in this step.

4 Start the primary (target) Adaptive Server to recover the primary (target) database and bring it online.

Open an operating system command prompt window on the primary Adaptive Server host, and use the RUN script to start the primary Adaptive Server:

```
$SYBASE/ASE_12-5/install/RUN_pds
```

where:

- *$SYBASE* is the path to the Adaptive Server installation directory.
- *pds* is the name of the primary Adaptive Server.

The primary data server will start up in "recovery" mode, and it may take a while for the primary database to come online.

Materializing databases in ASE 12.5.1 or later

This section describes two materialization procedures for Mirror Activator databases in Adaptive Server version 12.5.1 or later:

- Materializing the standby database (required when the Mirror Activator system is set up)
- Re-materializing the primary database for failback

Both of these procedures use the snapshot materialization technique, which is described in "Materialization options" on page 203.

The materialization procedures in this section take advantage of the Adaptive Server mount command, introduced in version 12.5.1. Using mount simplifies the materialization procedure by allowing you to "create" devices at the target site, using the disk replication system's snapshot (or point-in-time copy) feature, and then mount those devices in the target Adaptive Server, without initializing the devices, creating an empty database, and shutting down and restarting the server.

To materialize a database in Adaptive Server version 12.5.1 or later, you can use either the procedures in this section, or the procedures in "Materializing databases in ASE 12.5.0.3" on page 204.

Note You *cannot* use the procedures in this section to materialize a database in Adaptive Server version 12.5.0.3. You must use the procedures in "Materializing databases in ASE 12.5.0.3" on page 204.

Materializing the standby database in ASE 12.5.1 or later

Materializing the standby database is one of the tasks required to set up the Mirror Activator system. Therefore, the following Mirror Activator system setup tasks (which are not strictly materialization tasks) must be performed during the standby database materialization procedure:

- Materialize the mirror log devices at the standby site, and set up the disk replication system for synchronous replication to the mirror log devices
- Initialize the Mirror Replication Agent

The snapshot materialization procedure mentions these setup tasks, but it does not describe them in detail. See "Setting up a new Mirror Activator system" on page 48 for detailed information about Mirror Activator system setup tasks.

Note *Before* you materialize a standby database, you must create the Replication Server database objects in the primary database, and then initialize the primary database using the Mirror Replication Agent `pdb_init` command. See "Setting up a new Mirror Activator system" on page 48 for more information.

Table A-3 provides a checklist of the snapshot materialization tasks for a standby database in Adaptive Server version 12.5.1 or later.

The checklist in Table A-3 assumes that the standby Adaptive Server is already installed, and configured identically to the primary Adaptive Server.

Table A-3: Materializing a standby database in ASE 12.5.1 or later

Task	Description
1	In the standby Adaptive Server, create the same server logins and roles that are defined in the primary Adaptive Server.
2	Quiesce the primary database to suspend update activity and generate a manifest file for the primary database.
3	Use the disk replication system to copy a snapshot of all primary database data and log devices to the standby site, on devices accessible to the standby Adaptive Server. While the primary database is suspended during Mirror Activator system setup, you must also: • Copy a snapshot of the primary database log devices to the mirror log devices, and configure synchronous replication from the primary log devices to the mirror log devices • Initialize the Mirror Replication Agent
4	Resume update activity in the primary database after all procedures in step 3 are complete.
5	In the standby Adaptive Server, mount the standby database devices (created in step 3) and bring the standby database online.

Use the following procedure for snapshot materialization of a standby database in Adaptive Server version 12.5.1 or later.

❖ **To materialize a standby database in ASE 12.5.1 or later**

1 In the standby Adaptive Server, create all server logins and roles defined in the primary Adaptive Server.

2 Quiesce the primary database to suspend all update activity, and generate a manifest file for the primary database.

Log in to the primary Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup hold pdb
for external dump to pdb_manifest
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *pdb* is the name of the primary database.
- *pdb_manifest* is the name of the manifest file.

The standby Adaptive Server uses the manifest file to mount the devices created by the disk replication system in the following step.

3 Using the disk replication system facilities:

- Capture a snapshot (or point-in-time) image of all of the primary database data and log devices
- Transfer the snapshot to the standby devices at the standby site

The standby devices must be accessible to the standby Adaptive Server, for use as database devices.

While the primary database is suspended during Mirror Activator system setup, you must also:

- Transfer the snapshot of the primary database log devices to the mirror log devices, and configure the disk replication system for synchronous replication from the primary log devices to the mirror log devices
- Initialize the Mirror Replication Agent, using the ra_init command

  See "Setting up a new Mirror Activator system" on page 48 for more information about initializing the Mirror Replication Agent.

Refer to the documentation provided by your disk replication system vendor for more information about the procedures in this step.

4 Resume update activity in the primary database after all procedures in step 3 are complete.

Note You must initialize the Mirror Replication Agent *before* you resume update activity in the primary database.

Log in to the primary Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup release
```

where *MA_setup* is a user-defined tag that identifies the suspended primary database.

5 Mount the standby devices in the standby Adaptive Server to recover the standby database, and then bring it online.

a Log in to the standby Adaptive Server with a System Administrator user role, and execute the following command:

```
mount database all from pdb_manifest
with listonly
```

where *pdb_manifest* is the manifest file created by the quiesce command at the primary database.

The mount command with listonly option returns the device paths specified at the primary Adaptive Server for all primary database data and log devices.

If necessary, invoke the mount command to "re-map" the device paths to the standby devices in the standby Adaptive Server. For example:

```
mount database all from pdb_manifest
using "sdb_path" = "pdb_data"
```

where:

- *pdb_manifest* is the manifest file created by the quiesce command at the primary database.
- *sdb_path* is the path to the standby database data device.
- *pdb_data* is the device name of the primary database data device specified in the primary Adaptive Server.

When you invoke mount, Adaptive Server performs all of the required supporting activities, including adding database devices and activating them, creating the catalog entries for the new database, and recovering the database.

b After the standby Adaptive Server completes the mount processing, use the following command to bring the standby database online:

```
online database sdb
```

where *sdb* is the name of the standby database.

The names of the standby database and primary database must be the same.

Re-materializing the primary database for failback in ASE 12.5.1 or later

Normally, the primary database is the source of all data and transactions replicated in the Mirror Activator system. Re-materializing the primary database is required only as part of a failback procedure, to restore normal system operations after a failover.

After a failover event, the standby database becomes the "active" database in the system, and the primary database must be re-materialized from the standby

database to restore the normal operating condition, and resume replication from the primary database to the standby database.

Note When you re-materialize a primary database, the primary database is the *target* and the standby database is the *source* in the materialization procedure.

Table A-4 provides a checklist of the snapshot re-materialization tasks for a primary (target) database in Adaptive Server version 12.5.1 or later.

The checklist in Table A-4 assumes that:

- The primary (target) Adaptive Server is already installed, and configured identically to the standby (source) Adaptive Server.
- The primary (target) database exists in the primary (target) Adaptive Server, and its database options and devices are configured identically to the standby (source) database.
- All server logins defined in the standby (source) Adaptive Server exist in the primary (target) Adaptive Server, with identical suid values and names, and all server roles defined in the standby (source) Adaptive Server exist in the primary (target) Adaptive Server.

Table A-4: Re-materializing a primary database in ASE 12.5.1 or later

Task	Description
1	Quiesce the standby (source) database to suspend all update activity and generate a manifest file for the standby (source) database. The standby (source) database will remain suspended during failback, until after the materialization procedure is complete for the primary (target) database.
2	Use the disk replication system to copy a snapshot of all standby (source) database data and log devices to the primary (target) database devices.
3	In the primary (target) Adaptive Server, mount the primary (target) database devices (created in step 3) and bring the primary (target) database online.

After you complete the primary database re-materialization, you must perform additional tasks to complete the failback procedure. See Appendix B, “Failover and Failback with Mirror Activator,” for more information about failback procedures.

Use the following procedure for snapshot re-materialization of a primary database in Adaptive Server version 12.5.1 or later.

❖ **To re-materialize a primary database in ASE 12.5.1 or later**

1 Quiesce the standby (source) database to suspend all update activity, and generate a manifest file for the standby (source) database.

Log in to the standby (source) Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup hold sdb
for external dump to sdb_manifest
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *sdb* is the name of the standby (source) database.
- *sdb_manifest* is the name of the manifest file.

The primary (target) Adaptive Server uses the manifest file to mount the devices created by the disk replication system in the following step.

Note The standby (source) database will remain suspended during failback, until after the materialization procedure is complete for the primary (target) database.

2 Using the disk replication system facilities:

- Capture a snapshot (or point-in-time) image of all of the standby (source) database data and log devices
- Transfer the snapshot to the primary (target) devices at the primary (target) site

While the standby (source) database is suspended during failback, you must also:

- Transfer the snapshot of the standby (source) database log devices to the mirror log devices
- Configure the disk replication system for synchronous replication from the primary (target) log devices to the mirror log devices (to prepare for normal system operation after failback)

Refer to the documentation provided by your disk replication system vendor for more information about the procedures in this step.

3 Mount the primary (target) devices in the primary (target) Adaptive Server to recover the primary (target) database, and then bring it online.

a Log in to the primary (target) Adaptive Server with a System Administrator user role, and execute the following command:

```
mount database all from sdb_manifest
with listonly
```

where *sdb_manifest* is the manifest file created by the quiesce command at the standby (source) database.

The mount command with listonly option returns the device paths specified at the standby (source) Adaptive Server for all standby (source) database data and log devices.

If necessary, invoke the mount command to "remap" the device paths to the primary (target) devices in the primary (target) Adaptive Server. For example:

```
mount database all from sdb_manifest
using "pdb_path" = "sdb_data"
```

where:

- *sdb_manifest* is the manifest file created by the quiesce command at the standby (source) database.
- *pdb_path* is the path to the primary (target) database data device.
- *sdb_data* is the device name of the standby (source) database data device specified in the standby (source) Adaptive Server.

When you invoke mount, Adaptive Server performs all of the required supporting activities, including adding database devices and activating them, creating the catalog entries for the new database, and recovering the database.

b After the primary (target) Adaptive Server completes the mount processing, use the following command to bring the primary (target) database online:

```
online database pdb
```

where *pdb* is the name of the primary (target) database.

The names of the primary database and standby database must be the same.

APPENDIX B

Failover and Failback with Mirror Activator

This appendix describes tasks that you must incorporate in failover and failback procedures for the Mirror Activator 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 system.

This appendix describes only failover and failback tasks that are specific to the Mirror Activator 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 system components are set up and properly configured, as described in Chapter 2, “Setup and Configuration,” and when functioning normally, the Mirror Activator 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 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 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 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 system. General system failover procedures are not covered in this document.

Note The failover procedure described in this section does *not* use the Replication Server warm standby switch active feature. See the Replication Server *Administration Guide* for information about switching the active and standby databases in a warm standby application.

Table B-1 provides a checklist of the tasks that you must include in failover procedures for the Mirror Activator 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 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. **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: • Treat the standby database data and log devices as "primary" devices. • 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 system failover tasks.

❖ **To fail over the Mirror Activator 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:

```
suspend
```

After you invoke suspend, use the ra_status command to verify that the Mirror Replication Agent is in *Admin* state.

See "Stopping replication in the Mirror Replication Agent" on page 76 and "Determining current Mirror Replication Agent status" on page 79 for more information.

2 Verify that the Replication Server has finished distributing the last committed transaction to the standby database.

Log in to the Replication Server and execute the following command:

```
admin who, sqt
```

When Replication Server has finished distributing the last committed transaction, the admin who, sqt output shows the value 0 (zero) in the Closed column for the SQT thread associated with the standby database connection.

See the Replication Server *Reference Manual* for more information about the admin who, sqt command.

3 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.

See the Replication Server *Reference Manual* for more information about the admin who command.

4 Fail over the disk replication system.

Reconfigure the disk replication system to:

- Treat the standby database data and log devices as “primary” (or source) devices
- Mirror all subsequent changes on the standby database data and log devices to another set of standby (or 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.

5 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 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 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 system failback tasks

Task	Description
1	Quiesce the standby database to suspend update activity.
2	Fail back the disk replication system to: • Re-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.
3	Bring the primary database online.
4	Initialize the primary database using the pdb_init move_truncpt command to set the secondary 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 hold 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 hold 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 system failback tasks.

❖ **To fail back the Mirror Activator system**

1 Quiesce the standby database to suspend update activity.

The quiesce method you use depends on whether you want to use the snapshot materialization mount database option (for databases in Adaptive Server version 12.5.1 or later).

- If the Mirror Activator databases are in Adaptive Server version 12.5.0.3, log in to the standby Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_fback hold sdb
```

where:

- *MA_fback* is a user-defined tag that identifies the database.
- *sdb* is the name of the standby database.

- If you want to use the snapshot materialization mount database option for databases in Adaptive Server version 12.5.1 or later, log in to the standby Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_fback hold sdb
for external dump to sdb_manifest
```

where:

- *MA_fback* is a user-defined tag that identifies the database.
- *sdb* is the name of the standby database.
- *sdb_manifest* is the name of the manifest file.

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.

See Appendix A, “Materializing Databases,” for more information about re-materializing a primary database.

3 Bring the primary database online.

4 Initialize the primary database using the Mirror Replication Agent pdb_init move_truncpt command. This command marks the database for replication with sp_reptostandby, and sets the secondary truncation point to the end of the log.

See “Setting up the Mirror Activator system” on page 43 for more information about initializing the primary database.

5 Quiesce the primary database.

Log in to the primary Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup hold pdb
```

where:

- *MA_setup* is a user-defined tag that identifies the database.
- *pdb* is the name of the primary database.

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).

See “Updating the RASD” on page 85 for more information.

7 Release the quiesce hold on the primary database, *after* the Mirror Replication Agent initialization is complete.

Log in to the primary Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_setup release
```

where *MA_setup* is a user-defined tag that identifies the suspended primary database.

8 Switch access for users and client applications from the standby database to the primary database.

9 Release the quiesce hold on the standby database, after client access is switched to the primary database.

Log in to the standby Adaptive Server with a System Administrator user role, and execute the following command:

```
quiesce database MA_fback release
```

where *MA_fback* is a user-defined tag that identifies the suspended standby 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.

See "Starting replication in the Mirror Replication Agent" on page 75 for more information.

Glossary

This glossary describes Mirror Activator 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 DBMS footprint, which is ideal for embedded applications and mobile device applications.

See also **DBMS** 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.

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** 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. See also **transaction**. Contrast with **rollback**.

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 also 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**, **DBMS**, and **RDBMS**.

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 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**.

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. See also **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. See also **failback**.

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 heterogeneous replication, in which the primary and standby 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 programming language developed by Sun Microsystems. A platform-independent, "write once, run anywhere" programming language.

Java VM The Java Virtual Machine. The Java VM (or JVM) is the part of the Java Runtime Environment (JRE) that is responsible for interpreting 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 **data server** and **Java**.

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. The JRE must be installed on a machine to run Java applications, such as the Mirror Replication Agent. See also **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.

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.

See also **disk replication** and **transaction replication**.

LOB An abbreviation for *large object*. A LOB is a type of data element that is 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** and **Log Transfer Manager**.

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** and **Log Transfer Manager**.

Log Transfer Manager — An internal component of the Mirror Replication Agent that interacts with the other Mirror Replication Agent internal components to control and coordinate Mirror Replication Agent operations. 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**.

materialization — The process of copying the data from a primary database to a standby database, initializing the standby database so that the Mirror Activator system can begin replicating transactions. See also **atomic materialization**, **bulk materialization**, and **non-atomic materialization**.

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 **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** and **stable queue**.

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 **Mirror Replication Agent**, **primary database**, and **Replication Server**.

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 database that is the source of replicated data in a replication system. Sometimes called the *active database*. Contrast with **standby database**. See also **primary data**.

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**, **replicated transaction**, 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 updates have been propagated to their destinations. Some Mirror Replication Agent and Replication Server commands require that you first quiesce the replication system.

In a database, a state in which all data updates are suspended so that transactions cannot change any data and the data and log devices are stable.

This term is interchangeable with *quiesced* and *in quiesce*. See also **quiesce**.

RCL — An abbreviation for *Replication Command Language.* RCL is the command language used to manage 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. 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 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** and **Replication Server**.

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 **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 **disk replication** and **transaction replication**.

rollback An instruction to a database to back out of the changes requested in a unit of work (called 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**, **Replication Server**, and **route**.

standby data The data managed by a standby database, which is the destination (or target) of a replication system. See also **data replication** and **standby database**.

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 data in a replication system. Sometimes called the *replicate database*. Contrast with **primary database**. See also **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** and **Replication Server**.

table — In a relational DBMS, 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 for the table. See also **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 **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 **Replication Server**.

transaction replication — A data replication method that copies data-changing operations from a primary database transaction log to a standby database. See also **data replication** and **disk replication**.

transactional consistency — A condition in which all transactions in the primary database are applied in the standby database, in the same order that they were applied in the primary database.

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

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W