Primary Database Guide

Mirror Replication Agent

15.0

[Linux, Microsoft Windows, and UNIX]

DOCUMENT ID: DC00712-01-1500-01

LAST REVISED: March 2007

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

Contents

About This Book

Sybase® Mirror Replication Agent™ version 15.0 extends the capabilities of Replication Server® by supporting Sybase and non-Sybase primary data servers in a Sybase replication system.

Mirror Replication Agent is the software solution for replicating transactions from a primary database in one of the following data servers:

- Adaptive Server Enterprise (ASE)
- Oracle

Audience

This book is for anyone who needs to administer a Sybase replication system with Sybase and non-Sybase primary data servers.

If you are new to Sybase replication technology, refer to the following documents:

- The Replication Server *Design Guide* for an introduction to basic data replication concepts and Sybase replication systems
- The Replication Server *Heterogeneous Replication Guide* for an introduction to heterogeneous replication concepts and the issues peculiar to Sybase replication systems with non-Sybase data servers.

How to use this book

Refer to this book when you need detailed information about Mirror Replication Agent support for Sybase and non-Sybase data servers.

This book is organized as follows:

Chapter 1, “Mirror Replication Agent for ASE,” describes the characteristics of the Mirror Replication Agent that are unique to the Mirror Replication Agent for ASE implementation.

Chapter 2, “Mirror Replication Agent for Oracle,” describes replication system issues that are specific to Oracle, and details of the Mirror Replication Agent for Oracle.

Appendix A, “Migration for Mirror Replication Agent for ASE,” describes how to upgrade and downgrade ASE and Mirror Replication Agent.

Appendix B, “Migration for Mirror Replication Agent for Oracle,” describes how to upgrade and downgrade Oracle and Mirror Replication Agent.

Related documents

A Sybase Replication Server—Heterogeneous Replication Options system comprises several components. You may find it helpful to have the following documentation available.

Replication Agent

- The Replication Server—Heterogeneous Replication Options *Administration Guide* introduces replication concepts and Sybase replication technology, and describes Mirror Replication Agent features and operations, and how to set up, administer, and troubleshoot the Replication Agent software.
- The Mirror Replication Agent *Reference Manual* describes all Mirror Replication Agent commands and configuration parameters in detail, including syntax, examples, and usage notes.
- The Mirror Replication Agent *Installation Guide* describes how to install the Mirror Replication Agent software. It includes an installation and setup worksheet that you can use to collect all of the information you need to complete the software installation and Mirror Replication Agent setup.
- The Mirror Replication Agent 15.0 release bulletin contains last-minute information that was too late to be included in the books.

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

Java environment

Mirror Replication Agent 15.0 requires a Java Runtime Environment (JRE) on the machine that acts as the Mirror Replication Agent host.

- The Mirror Replication Agent 15.0 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 your Java environment.

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

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

Replication Server

- *Administration Guide* – includes information and guidelines for creating and managing a replication system, setting up security, recovering from system failures, and improving performance.

- *Configuration Guide* for your platform – describes configuration procedures for Replication Server and related products, and explains how to use the rs_init configuration utility.
- *Design Guide* – contains information about designing a replication system and integrating non-Sybase data servers into a replication system.
- *Getting Started with Replication Server* – provides step-by-step instructions for installing and setting up a simple replication system.
- *Heterogeneous Replication Guide* – describes how to implement a Sybase replication system with heterogeneous or non-Sybase data servers.
- *Reference Manual* – contains the syntax and detailed descriptions of Replication Server commands in the Replication Command Language (RCL); Replication Server system functions; Replication Server executable programs; and Replication Server system tables.
- *Troubleshooting Guide* – contains information to aid in diagnosing and correcting problems in the replication system.

Primary data servers

Sybase recommends that you or someone at your site be familiar with the software and database administration tasks for the following data servers supported by Mirror Replication Agent:

- ASE
- Oracle

Sybase Adaptive Server

If your replication system includes databases in Adaptive Server® Enterprise, make sure that you have documentation appropriate for the version of Adaptive Server Enterprise that you use.

More information about Adaptive Server Enterprise can be found at the following URL:

```
http://www.sybase.com/support/manuals/
```

Other sources of information

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

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

- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

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

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

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

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ To find the latest information on product certifications

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click Certification Report.
3. In the Certification Report filter, select a product, platform, and time frame, and then click Go.
4. Click a Certification Report title to display the report.

❖ To find the latest information on component certifications

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.
2. Either select the product family and product under Search by Base Product, or select the platform and product under Search by Platform.
3. Select Search to display the availability and certification report for the selection.

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

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

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ To find the latest information on EBFs and software maintenance

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.
4. Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

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

Conventions

The following sections describe the style, syntax, and character case conventions used in this book.

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:

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

  Specify the value of *value* to mark the table.

- 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 text_connection command to mark a table for replication.

  - Names of database objects (such as tables, columns, stored procedures) appear like this:

    Check the price column in the widgets table.

  - Names of datatypes appear like this:

    Use the date or datetime datatype.

  - Names of files and directories appear like this:

    Log files are located in the *$SYBASE/MA-15_0/inst_name/log* subdirectory.

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.

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

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, RA_CONFIG, Ra_Config, and ra_config 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 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.

Sybase Mirror Replication Agent version 15.0 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

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

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

For a Section 508 compliance statement for Sybase Mirror Replication Agent version 15.0, see Sybase Accessibility at http://www.sybase.com/products/businesscontinuity/mirroractivator.

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 Mirror Replication Agent for ASE

Mirror Replication Agent for ASE refers to an instance of the Mirror Replication Agent version 15.0 software that is configured for an ASE database.

This chapter describes the characteristics of the Mirror Replication Agent that are unique to the Mirror Replication Agent for ASE implementation.

Note For information on the basic features and operation of Mirror Replication Agent version 15.0, refer to the Mirror Activator *Administration Guide* and Mirror Replication Agent *Reference Manual*.

ASE Database-specific issues

This section describes general issues and considerations that are specific to using Mirror Replication Agent version 15.0 with the ASE Database.

The following topics are included in this section:

- ASE Specific issues
- Converting a warm standby application to a Mirror Activator system
- Materializing databases in ASE 12.5.1 or later

ASE Specific issues

The following topics are included in this section:

- Mirror Replication Agent connectivity
- Mirror Replication Agent permissions

- Format of origin queue ID

Mirror Replication Agent connectivity

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 for ASE 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.

Mirror Replication Agent permissions

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. This permission can be obtained by granting replication_role to the Mirror Replication Agent user ID that is used to access the primary ASE database.

Grant the replication role to the Mirror Replication Agent login name as in the following example:

```
grant role replication_role to ma_pds_user
```

where ma_pds_user is the Mirror Replication Agent user login name assigned to configuration parameter pds_username.

Format of origin queue ID

Each record in the transaction log is identified by an origin queue ID that consists of 64 hexadecimal characters (32 bytes). The format of the origin queue ID is determined by the Mirror Replication Agent instance, and it varies according to the primary database type.

Table 1-1 illustrates the format of the origin queue ID for the Mirror Replication Agent for ASE

Table 1-1: Mirror Replication Agent for ASE origin queue ID

Character	Bytes	Description
0-3	2	Database generation ID
4-15	6	Log page timestamp for the current record
16-27	6	Row ID of the current row. Row ID = page number (4 bytes) + row number (2 bytes)
28-39	6	Row ID of the begin record for the oldest open transaction
40-55	8	Date and time of the begin record for the oldest open transaction
56-59	2	An extension used by the RepAgent to roll back orphan transactions
60-63	2	Unused

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 the Mirror Activator *Administration Guide,* Chapter 2, “Setting up and Configuring Mirror Replication Agent.” for complete setup and configuration tasks.

Table 1-2 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 1-2 assumes the following:

- 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 do not need to 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 the Mirror Activator *Administration Guide*, Chapter 2, “Setting Up and Configuring Mirror Replication Agent,” *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 1-2. Instead, you must instead use the setup and configuration tasks described in “Setting up a new Mirror Activator system,” in Chapter 2 of the Mirror Activator *Administration Guide*.

When converting an existing warm standby application to a Mirror Activator system, you must perform all of the tasks in Table 1-2 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 1-2: Tasks for converting a warm standby application to a Replication Server—Heterogeneous Replication Options 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 Replication Agent thread in the primary database. **Note** You must preserve the secondary truncation point when you disable the Replication Agent thread.
4	Initialize the primary database using the Mirror Replication Agent pdb_init command. **Note** After the primary database 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 a new Mirror Activator system,” in Chapter 2 of the Mirror Activator *Administration Guide* – ***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 Replication Agent 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 Replication Agent thread

1. Log in to the primary database with a System Administrator or Database Owner user role.
2. View the current values of the Replication Agent 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 Replication Agent thread parameters:

- rs username – Replication Server user login for the Replication Agent 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 Replication Agent thread parameters. Consult the System Administrator or System Security Officer for the Replication Server to obtain the password that the 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, ma_rs_user
```

where *ma_rs_user* is the value of the Mirror Replication Agent thread rs_username parameter.

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

```
ra_config rs_password, ma_rs_pwd
```

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

6 Specify the Replication Server character set:

```
rs_charset
```

where rs_charset matches the character set used by Replication Server and is found in the configuration (*.cfg*) file.

7 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 Replication Agent thread connect dataserver parameter.

8 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 Replication Agent 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. For more information, see the Mirror Activator *Administration Guide*, Chapter 3, “Administering Mirror Replication Agent.”

Stop and disable the Mirror Replication Agent 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 Replication Agent 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 Replication Agent 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 Replication Agent 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 database with a System Administrator user role.
2. Stop the Replication Agent thread in the primary database:

```
use pdb
sp_stop_rep_agent pdb
```

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

3. Disable the Replication Agent 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 ASE Replication Agent 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 database, 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

Quiesce the primary database to suspend update activities until the Mirror Replication Agent is initialized.

Note A System Administrator user role in the primary Adaptive Server must exist to perform this procedure.

❖ To quiesce the primary database

1 Log in to the primary database 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

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 paths 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 paths 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.

For more information about the ra_devicepath and ra_helpdevice commands, see the Mirror Replication Agent *Reference Manual*.

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.

Do *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 A System Administrator user role in the primary Adaptive Server must exist to perform this procedure.

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

1. Log in to the primary database 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 the Mirror Activator Administration Guide, Chapter 4, “Troubleshooting Mirror Replication Agent,” for more information.

Materializing databases in ASE 12.5.1 or later

This section describes two materialization procedures for Replication Server—Heterogeneous Replication Options databases in Adaptive Server version 12.5.1 or later:

- Materializing the standby database (required when the Replication Server—Heterogeneous Replication Options system is set up)
- Re-materializing the primary database for failback

Both of these procedures use the snapshot materialization technique.

The materialization procedures in this section take advantage of the Adaptive Server mount command. 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. This feature eliminates 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 the procedures in this section.

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. For detailed information, see the Mirror Activator *Administration Guide*, Chapter 2, "Setting Up and Configuring Mirror Replication Agent."

Note *Before* you materialize a standby database, create the Replication Server database objects in the primary database, and then initialize the primary database using the Mirror Replication Agent pdb_init command. For more information, see the Mirror Activator *Administration Guide*, Chapter 2, "Setting Up and Configuring Mirror Replication Agent."

Table 1-3 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 1-3 assumes that the standby Adaptive Server is already installed and configured identically to the primary Adaptive Server.

Table 1-3: 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 Replication Server—Heterogeneous Replication Options system setup, you also need to: • 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.

The most efficient way to complete this task is to use an external copy utility (such as bcp) to copy the contents of the following tables from the primary Adaptive Server master database to the standby Adaptive Server master database:

- sysloginroles
- syslogins
- sysroles
- syssrvroles

See the Adaptive Server *Utility Guide* for more information about using the bcp utility.

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 Use the disk replication system facilities to do the following:

- 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 Replication Server—Heterogeneous Replication Options 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

  For more information see the Mirror Activator *Administration Guide*, Chapter 2, "Setting Up and Configuring 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:

```
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, to bring the standby database online:

```
online database sdb
```

where *sdb* is the name of the standby database.

Note 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 Replication Server—Heterogeneous Replication Options 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 1-4 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 1-4 assumes the following:

- 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 1-4: 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. For more information, see the Mirror Activator *Administration Guide*.

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.

2. Use the disk replication system facilities to do the following:

   - 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.

Datatype support

All ASE datatypes are supported by Mirror Replication Agent. The following is additional information for two of the datatypes: encrypted columns and computed columns:

- Encrypted columns – the ability to encrypt most columns as they are stored on disk. (Keys and some indexing may not be encrypted.) The logged data is ciphertext (encrypted), and transmitted to Replication Server as ciphertext. A new option to insert data already encrypted has been added, enabling the data to be transmitted in an image form to the replicate database.
- Computed columns – defined by an expression, whether from regular columns in the same row, functions, arithmetic operators, or path names. Computed columns are different from function based indexes in the following ways:
  - A computed column provides a shorthand for an expression and indexability.
  - Computed columns can be either deterministic or non-deterministic, while function based indexes must be deterministic. "Deterministic" means that if the input values in an expression are the same, the return values must also be the same.
  - Computed columns can be materialized or not materialized. Columns that are materialized are preevaluated and stored in the table when base columns are inserted or updated. Any subsequent access to a materialized column does not require reevaluation; its pre-evaluated result is accessed

- Computed columns that are not materialized are called virtual columns, with the value evaluated each time the column is accessed. A non-deterministic expression for a virtual column may result in different values for each access

Materialized computed columns need to be replicated as there is no method to determine if the expression is deterministic or non-deterministic.

Note A table level replication definition (repdef) for encrypted columns must be provided, even if database level replication or warm standby is used.

CHAPTER 2 Mirror Replication Agent for Oracle

Mirror Replication Agent for Oracle refers to an instance of the Mirror Replication Agent version 15.0 software installed and configured for a primary database that resides in an Oracle data server.

This chapter describes the characteristics of the Mirror Replication Agent that are unique to the Mirror Replication Agent for Oracle implementation.

Note For information on the basic functionality of Mirror Replication Agent version 15.0, refer to the Mirror Activator *Administration Guide* and *Reference Manual*.

Oracle-specific issues

This section describes general issues and considerations that are specific to using Mirror Replication Agent version 15.0 with the Oracle data server:

- Mirror Replication Agent connectivity
- Mirror Replication Agent permissions
- Redo and archive log setup
- Supplemental logging
- Flashback enhancements
- Setting ddl_username and ddl_password

- Character case of database object names
- Format of origin queue ID
- Datatype compatibility
- Oracle datatype restrictions
- Oracle large object (LOB) support
- Oracle user-defined types

Mirror Replication Agent connectivity

Connectivity between the Mirror Replication Agent for Oracle and the Oracle data server is through the Oracle JDBC thin driver.

The Oracle JDBC driver must be installed on the Mirror Replication Agent host machine, and the directory in which this driver is installed must be in the CLASSPATH environment variable.

The TNS Listener Service must be installed and running on the primary database so the Mirror Replication Agent instance can connect to it. For more information, see the *Oracle Database Net Services Administrator's Guide*.

Mirror Replication Agent permissions

Mirror Replication Agent uses the pds_username configuration parameter to connect to Oracle and must have the following Oracle permissions:

- create session – required to connect to Oracle.
- select_catalog_role – required to select from the DBA_* views.
- alter system – required to perform *redo* log archive operations.
- execute on DBMS_FLASHBACK – required to execute DBMS_FLASHBACK.get_system_change_number.
- alter any procedure – required to instrument procedures for replication.
- create table – required to create tables in the primary database.
- create procedure – required to create rs_marker and rs_dump proc procedures.

- create public synonym – required to create synonyms for created tables in the primary database.
- create sequence – required to create a sequence number for each procedure/function call.
- drop public synonym – required to drop created synonyms.
- select on SYS.OBJ$ – required to process procedure DDL commands.
- select on SYS.LOB$ – required to support LOB replication.
- select on SYS.COLLECTION$ – required to support table replication.
- select on SYS.COL$ – required to support table replication.
- select on SYS.COLTYPE$ – required to support schema column.
- select on SYS.CON$ – required to support table replication.
- select on SYS.CDEF$ – required to support replication.
- select on SYS.IND$ – required to support replication.
- select on SYS.USER$ – required to support replication.
- select on SYS.SEQ$ – required to support sequence replication.

In addition, the user who starts the Mirror Replication Agent for Oracle instance must have read access to the Oracle *redo log* files and the Oracle archive directory that contains the *archive log* files to be accessed for replication. If the Mirror Replication Agent is configured to remove old archive files, the user must have update authority to the directory and the *archive log* files.

Redo and archive log setup

Note The Mirror Replication Agent for Oracle *must* be installed on a machine where it can directly access the Oracle *redo log* and *archive log* files.

You can access both online and archive logs by default. If you want to access only the online logs, the Mirror Replication Agent can be configured to do so, but it requires that you turn auto-archiving off and that you set Mirror Replication Agent to issue manual archive log commands to Oracle.

Accessing archive logs

When the default is used and archive log files are to be accessed, configure the Mirror Replication Agent to use the directory path where the archive log files are located. To prevent conflicts with other archive file processes, you may want to configure Oracle to produce archive log files into an additional directory used only for replication. Mirror Replication Agent can be configured to remove archive log files when they are no longer needed. Sybase recommends that you only configure the Mirror Replication Agent to remove archive log files if an additional directory is used.

To enable *redo* log archiving:

```
alter database ARCHIVELOG;
```

To verify that log archiving is enabled:

```
select log_mode from v$database;
```

If ARCHIVELOG (ARCHIVELOG or MANUAL in Oracle 10g) is returned, then log archiving is enabled.

Accessing archive log files from a remote site

Since archive log files are always files, not raw devices, you must be careful to ensure the disk replication system and file system at the remote site allows the archive logs to be accessed from the remote site. If the disk replication system or remote file system does not provide a means to have access to archived log files from the primary site, the archive log files must be provided to the remote site outside of disk replication.

As an example, if the file system on the remote site does not recognize files replicated using disk replication, you can write a copy of the archive log files directly from the primary to a remote system's device, using a remote mount. To do this, make a remote site device mountable to the primary system and mount that device at the primary site. Then, configure Oracle to write an additional copy of the archive log files to this remote device (see Oracle command "alter system" and parameter "log_archive_dest" for details on adding additional archive log destinations to your Oracle environment).

In the Mirror Replication Agent, set configuration property pdb_archive_path to the remote device location. You can also set Mirror Replication Agent configuration parameter pdb_remove_archives to true, to allow the Replication Agent to remove these archive log files when they are no longer needed to support replication.

Setting archiving for Mirror Replication Agent

When pdb_include_archives is set to true (the default) the Mirror Replication Agent does not do archiving and Sybase recommends that you configure Oracle to do automatic archiving of *redo* logs.

When the configuration parameter pdb_include_archives is set to false, Mirror Replication Agent for Oracle requires that automatic archiving of Oracle *redo* logs be disabled. Archiving is performed manually by the Mirror Replication Agent as the data in the *redo* log files is replicated.

Mirror Replication Agent for Oracle requires the following settings in your Oracle database depending on the Oracle version when configuration parameter pdb_include_archives is set to false, disabling usage or archived *redo* logs.

For Oracle 10g

❖ To disable automatic archiving

1 Enter the following:

```
alter database ARCHIVELOG MANUAL;
```

2 Verify that log archiving is disabled:

```
select log_mode from v$database;
```

If MANUAL is returned, then automatic log archiving is disabled.

For Oracle 9i

❖ To disable automatic archiving

1 To change the LOG_ARCHIVE_START parameter, you can manually edit the server's start-up parameter file or use the following Oracle command:

```
alter system set log_archive_start=false
scope=spfile;
```

2 Check the setting of the LOG_ARCHIVE_START parameter:

```
select value from v$system_parameter where name =
'log_archive_start';
```

3 If false is returned, the value in the server parameter file has been correctly modified to prevent automatic archiving when you re-start the Oracle server. For more information about the LOG_ARCHIVE_START parameter or the ALTER SYSTEM commands, see the Oracle *Database Reference Guide*.

4 Automatic archiving must be disabled in the active server when you re-start the Oracle server. To stop automatic archiving in the active server:

```
alter system archive log stop;
```

5 To disable automatic archiving when you re-start the Oracle server, change the value of the server's LOG_ARCHIVE_START parameter to false.

Note This note applies only when pdb_include_archives is set to false. For *redo* log file processing after Mirror Replication Agent for Oracle is initialized, automatic archiving must *never* be enabled, even temporarily. If automatic archiving is re-enabled or manual archiving is performed, causing a *redo* log file not yet processed by the Mirror Replication Agent to be overwritten, then the data in the lost *redo* log file will not be replicated. To recover from this situation, reconfigure the Mirror Replication Agent to access archive log files. Set pdb_include_archives to true, set pdb_archive_path to the directory location that contains the archive of the file that has been overwritten, and resume. After catching up, suspend the Mirror Replication Agent, reset pdb_include_archives to false, and "resume" the Mirror Replication Agent.

Forced logging of all database changes

You can enable the forced logging of all database changes to the Oracle *redo* log file. Sybase recommends setting this option to insure that all data that should be replicated is logged. To enable the force logging command, execute the following statement on the primary database:

```
alter database FORCE LOGGING;
```

To verify the current setting of the force logging command, execute the following statement on the primary database:

```
select force_logging from v$database;
```

Supplemental logging

In Oracle release 9.2 and later, minimal supplemental logging and supplemental logging of primary key data and index columns must be enabled. To enable supplemental logging:

```
alter database add SUPPLEMENTAL LOG DATA;
alter database add SUPPLEMENTAL LOG DATA (PRIMARY KEY,
UNIQUE INDEX) COLUMNS;
```

To verify that minimal supplemental logging and supplemental logging of primary key and unique index information is enabled:

```
select SUPPLEMENTAL_LOG_DATA_MIN,
SUPPLEMENTAL_LOG_DATA_PK, SUPPLEMENTAL_LOG_DATA_UI
from v$database;
```

If YES is returned for each column, then supplemental logging of primary key information is enabled.

Flashback enhancements

Oracle's new flashback feature available in Oracle version 10g is not supported in Replication Agent for Oracle. Because flashback is not supported, it requires that you disable the recycle bin:

- To disable the recycle bin which was added in Oracle 10.2 (which requires sysdba privileges):

  ```
  purge dba_recyclebin;
  ALTER SYSTEM SET recyclebin = OFF;
  ```

- To disable the recycle bin in Oracle 10.1, you must set the Oracle hidden property:

  ```
  ALTER SYSTEM SET “_recyclebin”=FALSE SCOPE = BOTH;
  ```

In Oracle 10.2, to view the contents of the recycle bin:

```
select * from dba_recyclebin;
```

In Oracle 10.2, to view the current recycle bin configuration:

```
select value from v$parameter where name = “recyclebin”;
```

Setting *ddl_username* and *ddl_password*

To replicate DDL in Oracle, in addition to setting the value of pdb_setrepddl to enable, you must set the Mirror Replication Agent ddl_username and ddl_password parameters. The ddl_username parameter is the database user name included in LTL for replicating DDL commands to the standby database. This user must have permission to execute all replicated DDL commands at the standby database. The ddl_password parameter is the database user name’s password.

See the Mirror Replication Agent *Reference Manual* for details on setting these parameters.

When you replicate DDL in Oracle, you must use Oracle as the standby database. You *cannot* replicate DDL commands from Oracle to non-Oracle standby databases.

Note To replicate DDL, Replication Server must have a database-level replication definition with replicate DDL set in the definition. For details, see the Replication Server *Reference Manual*.

DDL commands and objects filtered from replication

The following DDL commands are not replicated:

alter database
create database link
drop database link
alter session
create snapshot
create snapshot log
alter snapshot
alter snapshot log
drop snapshot
drop snapshot/log
alter rollback segment
create rollback segment
drop rollback segment
alter system switch log
create control file
create pfile from spfile
create schema authorization
create spfile from pfile
explain
lock table
rename
set constraints
set role
set transaction
analyze
audit
no audit
create tablespace
alter tablespace
drop tablespace

The following objects are not replicated:

- Any objects that are owned by SYS.
- Any object owned by users defined in the list of non-replicated users. In addition, Sybase has provided a default list of owners whose objects will not be replicated. However, you cannot remove the SYS owner. You can use the pdb_ownerfilter command to return, add, or remove the list of owners whose objects will not be replicated. See the Mirror Replication Agent *Reference Manual* for more information.

Character case of database object names

Database object names must be delivered to the primary Replication Server in the same format as they are specified in replication definitions; otherwise, replication will fail. For example, if a replication definition specifies a table name in all lowercase, then that table name must appear in all lowercase when it is sent to the primary Replication Server by the Mirror Replication Agent.

To control the way Mirror Replication Agent 15.0 treats the character case of database object names when it sends LTL to the primary Replication Server, set the ltl_character_case configuration parameter to one of the following values:

- asis – (the default) database object names are passed to Replication Server in the same format as they are actually stored in the primary data server.
- lower – database object names are passed to Replication Server in *all lowercase*, regardless of the way they are actually stored in the primary data server.
- upper – database object names are passed to Replication Server in *all uppercase*, regardless of the way they are actually stored in the primary data server.

In the Oracle data server, database object names are stored in all uppercase by default. However, if you create a case-sensitive name, the case sensitivity is retained in Oracle.

See the following examples using the asis option:

- `create table tabA` is stored as `TABA`
- `create table Tabb` is stored as `TABB`
- `create table ‘TaBc’` is stored as `TaBc`

See the following examples using the upper option:

- `create table tabA` is stored as `TABA`
- `create table Tabb` is stored as `TABB`
- `create table 'TaBc'` is stored as `TABC`

Format of origin queue ID

Each record in the transaction log is identified by an origin queue ID that consists of 64 hexadecimal characters (32 bytes). The format of the origin queue ID is determined by the Mirror Replication Agent instance, and it varies according to the primary database type.

Table 2-1 illustrates the format of the origin queue ID for the Mirror Replication Agent for Oracle.

Table 2-1: Mirror Replication Agent for Oracle origin queue ID

Character	Bytes	Description
0-3	2	Database generation ID
4-19	8	System change number
20-27	4	Log sequence number
28-35	4	Block number
36-39	2	Block offset, relative to the start of the block
40-47	4	Oldest active transaction begin log sequence number
48-55	4	Oldest active transaction begin block number
56-59	2	Oldest active transaction begin block offset
60-63	2	Locator generation ID

Datatype compatibility

Mirror Replication Agent processes Oracle transactions and passes data to the primary Replication Server. In turn, the primary Replication Server uses the datatype formats specified in the replication definition to receive the data from the Mirror Replication Agent for Oracle.

Table 2-2 describes the conversion of Oracle datatypes to Sybase datatypes.

Table 2-2: Oracle to Sybase datatype mapping

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
BINARY_FLOAT	5 bytes, 32-bit single precision floating point number datatype	float	4 or 8 bytes, depending on precision	• Maximum positive finite value is 3.40282E+38F. • Minimum positive finite value is 1.17549E-38F.
BINARY_DOUBLE	9 bytes, 64-bit single precision floating point number datatype	double	8 bytes	• Maximum positive finite value is 1.79769313486231E+308. • Minimum positive finite value is 2.22507485850720-308.
CHAR	255 bytes	char	32K	
DATE	8 bytes, fixed-length, default format: DD-MON-YY	datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD. Default value replicated is YYYYMMDD. If pdb_convert_datetime is true, the value replicated is YYYYMMDD HH:MM:SS.sss.
TIMESTAMP(n)	21-31 bytes, variable-length, default format: DD-MON-YY hh.mm.ss.ffffff AM	datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD.
TIMESTAMP(n) WITH [LOCAL] TIME ZONE	Variable-length, default format: DD-MON-YY hh.mm.ss.ffffff AM {+|-}hh:mm	varchar(100)		
INTERVAL YEAR(n) TO MONTH	Variable-length	varchar(25)		
INTERVAL DAY(n) TO SECOND(n)	Variable-length	varchar()		
LONG	2GB, variable-length character data	text		

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
LONG RAW	2GB, variable-length binary data	image		
BLOB	4GB, variable-length binary large object	image		
CLOB	4GB, variable-length character large object	text		
NCHAR	255 bytes, multi-byte characters	unichar or char	32K	
NCLOB	4GB, variable-length multibyte character large object	text		
NVARCHAR2	2000 bytes, variable-length, multibyte character data	univarchar or varchar	32K	
BFILE	4GB, locator points to large binary file	image		BFILE replication is not supported for Mirror Activator.
MLSLABEL	5 bytes, variable-length binary OS label			Not supported.
NUMBER (p,s)	21 bytes, variable-length numeric data	float, int, real, number, or decimal	float is 4 or 8 bytes. int is 4 bytes. real is 4 bytes. number and decimal are 2 to 17 bytes.	The float datatype can convert to scientific notation if the range is exceeded. Integers (int) are truncated if they exceed the Replication Server range of 2,147,483,647 to -2,147,483,648 or 1x10^{-130} to 9.99x10^{25}. The number and decimal datatypes are truncated if they exceed the range of -10^{38} to 10^{38}-1. Oracle precision ranges from 1 to 38 digits. Default precision is 18 digits. Oracle scale ranges from -84 to 127. Default scale is 0.

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
RAW	2000 bytes, variable-length binary data	binary or varbinary	32K	
ROWID	6 bytes, binary data representing row addresses	char	32K	**Note** Sybase does not support marking of procedures that have a ROWID type argument.
UDD object type	Variable length character data	User-defined Replication Server datatype	32K	See "Oracle user-defined types" on page 37.
VARCHAR2	2000 bytes, variable-length character data	varchar	32K	

Oracle datatype restrictions

Note See the Mirror Replication Agent *Release Bulletin* for the latest information on datatype restrictions.

Replication Server and Mirror Replication Agent impose the following constraints on the Oracle NUMBER datatype:

- In the *integer* representation:
  - The corresponding Sybase int datatype has a smaller absolute maximum value.

    The Oracle NUMBER absolute maximum value is 38 digits of precision, between 9.9 x 10^{125} and 1 x 10^{-130}. The Sybase int value is between 2^{31} - 1 and -2^{31} (2,147,483,647 and -2,147,483,648), inclusive.
  - Oracle NUMBER values greater than the Sybase int maximum are rejected by Replication Server.
- In the *floating point* representation:

- The precision of the floating point representation has the same range limitation as the integer representation.
- If the floating point value is outside the Sybase range of 2^{31} - 1 and -2^{31} (2,147,483,647 and -2,147,483,648), Mirror Replication Agent for Oracle converts the number into exponential format to make it acceptable to Replication Server. No loss of precision or scale occurs.

Replication Server and Mirror Replication Agent impose the following constraints on the Oracle TIMESTAMP WITH [LOCAL] TIME ZONE datatype:

- When a TIMESTAMP WITH TIME ZONE datatype is replicated, the time zone information is used to resolve the timestamp value to the “local” time zone and then the resolved value is replicated. (The time zone information itself is not replicated.)
- For example, if a TIMESTAMP WITH TIME ZONE datatype is recorded in Oracle as “01-JAN-05 09:00:00.000000 AM -8:00” and the “local” time zone is -6:00, the value replicated will be “01-JAN-05 11:00:00.000000”. The timestamp value is adjusted for the difference between the recorded timezone of -8:00 and the local time zone of -6:00, and the adjusted value is replicated.

If you use a version of Replication Server prior to version 12.5, the following size restrictions are imposed on Oracle datatypes:

- Oracle BLOB, CLOB, NCLOB, and BFILE datatypes that contain more than 2GB are truncated to 2GB.
- Oracle CHAR, RAW, ROWID, and VARCHAR2 datatypes that contain more than 255 bytes are truncated to 255 bytes.
- Oracle NCHAR and NVARCHAR2 multibyte character datatypes are replicated as char or varchar single-byte datatypes.

Note With Replication Server version 12.5 or later, these datatype size restrictions are no longer in effect.

The following Oracle datatypes are not supported:

- Oracle REF type
- Oracle VARRAY type
- Oracle NESTED TABLE type
- Oracle-supplied types

See also

See the Replication Server *Reference Manual* for more information on replication definitions and the create replication definition command.

Oracle large object (LOB) support

Oracle LOB data can exist in several formats in Oracle. The LOB datatypes in Oracle are:

- Character:
  - LONG
  - CLOB
  - NCLOB
- Binary:
  - LONG RAW
  - BLOB
  - BFILE

  BFILE points to file contents stored outside of the Oracle database.

For those types stored in the database (all types except BFILE), Oracle records the content of the LOB in the *redo* files. The Mirror Replication Agent reads the LOB data from the *redo* file and submits the data for replication.

Because BFILE type data is stored outside of the database, the BFILE contents are not recorded in the *redo* file. To replicate the content of a BFILE, the Mirror Replication Agent connects to the primary Oracle database and issues a query to select the data from the BFILE. Selecting the BFILE data separate from other data in the *redo* log can provide a temporary out-of-sync condition if the BFILE contents are changed multiple times. As described in the Mirror Activator *Administration Guide*, querying LOB data from the database “outside” the transaction log’s contents allows only the last change to that BFILE to be replicated. Values from earlier transactions might not be sent to the standby site. See “Enabling and disabling replication for LOB columns” in the Mirror Activator *Administration Guide* for additional information.

Note Since the Mirror Replication Agent must query the primary database to obtain BFILE data, replication of BFILE data will fail to be replicated if the Primary database is unavailable.

Special handling for *off row* LOBS

LOB types that are stored within the Oracle database (BLOB, CLOB and NCLOB) can be defined with certain storage characteristics. One of those characteristics, “disable storage in row,” indicates that the data for the LOB should *always* be recorded separately from the rest of the data in the row the LOB belongs to. This *off-row* storage requires special handling for replication of updates to these LOB values.

When an off-row LOB value is updated, the change recorded in the *redo* log is for the index that holds the LOB’s data; the row the LOB belongs to is not changed. As a result, information is missing from the *redo* log to identify which row in the table the LOB belongs to.

For example, when a non-LOB column is updated in a table, all of the column data that identifies the changed values and look-up columns is recorded. The command `updated myTable set col2 = 2 where col1 = 1` records values in the *redo* log for the values of both “col2” and “col1.”

In contrast, a command that updates an LOB that has been defined with the disable storage in row clause records only the LOB data’s change to its index, and not the table that holds the LOB. So the command `updated myTable set ClobColumn = 'more data' where col1 = 1` only records the value changed, and does not include the value of “col1”.

Because the value of the columns in the where clause are not logged in that update, there is insufficient information to build the correct where clause to be used to apply the data at the standby site. To resolve this problem, Mirror Replication Agent for Oracle requires that an update to a LOB column defined with disable storage in row *must* be immediately accompanied by an insert or update to the same row in the table the LOB belongs to.

The Mirror Replication Agent uses the additional column data from the associated operation to correctly build the where clause required to support replication.

For example, the following transaction sequences support replication of updates to LOB column “ClobColumn” when it has been defined with the disable storage in row clause:

```
begin
insert into myTable (col1, col2, ClobColumn, updated)
values (1,1,empty_clob(), sysdate);
update myTable set ClobColumn = 'more data' where col1
= 1;
commit
```

```
begin
update myTable set updated = sysdate() where col1 = 1;
update myTable set ClobColumn = 'more data' where col1
= 1
commit

begin
update myTable set ClobColumn = 'more data' where col1
= 1
update myTable set updated = sysdate() where col1 = 1;
commit
```

The following transaction sequences are *not* supported for LOB columns defined with the disable storage in row clause and result in a failure to supply the LOB data to the standby site:

- Missing accompanying change to the same row:

  ```
  begin
  update myTable set ClobColumn = 'more data' where
  col1 = 1
  commit
  ```

- Accompanying change for the same row is not immediately adjacent to the LOB change:

  ```
  begin
  update myTable set updated = sysdate where col1 = 1;
  update myTable set col2 = 5 where col1 = 5;
  update myTable set ClobColumn = 'more data' where
  col1 = 1
  commit
  ```

This limitation only applies to LOB columns that have been defined with the disable storage in row clause.

You can identify the LOB columns in your database that have this constraint using the following query against your Oracle database:

```
select owner, table_name, column_name from dba_lobs where in_row = 'NO';
```

Oracle user-defined types

User-defined datatypes (UDD) use Oracle built-in datatypes and other user-defined datatypes as building blocks that model the structure and behavior of data in applications.

Mirror Replication Agent for Oracle version 15.0 supports replication of user-defined object types. Object types are abstractions of real-world entities, such as purchase orders, that application programs deal with. An object type is a schema object with three kinds of components:

- A name, which identifies the object type uniquely within that schema.
- Attributes, which are built-in types or other user-defined types. Attributes model the structure of the real-world entity.
- Methods, which are functions or procedures written in PL/SQL and stored in the database, or written in a language such as C or Java and stored externally. Methods implement operations the application can perform on the real-world entity.

Replicating UDDs

To replicate UDDs in Oracle, you must add a datatype definition to Replication Server so the UDD is replicated exactly as it is executed in the primary database. UDDs from Oracle are sent to Replication Server as data for a single varchar column. By default, Replication Server wraps all varchar data in single quotation marks. In order to prevent Replication Server from adding these quotation marks to UDD data, a special datatype must be created in Replication Server *and* that datatype must be used as the datatype for any UDD column defined in a replication definition.

When you create a datatype definition in Replication Server, you must use an unused datatype ID. This is the DTID column of the rs_datatype table. The new datatype is a Replication Server datatype, so it will be available to all connections defined in the Replication Server that owns the Replication Server system database (RSSD); you only have to do this once for each Replication Server instance.

❖ To create a datatype definition in Replication Server

To create the datatype requires Replication Server administrator privileges or granted permission.

1. Log in to the RSSD.
2. Add a row to the rs_datatype table using the following example as a guide:

```
/*  rs_oracle_udd_raw - char with no delimiters */
insert into rs_datatype values(
0,                     /* prsid */
0x0000000001000008,   /* classid */
'rs_oracle_udd',       /* name */
0x0000000000010210,  /* dtid */
0,                   /* base_coltype */
255,                 /* length */
0,                   /* status */
1,                   /* length_err_act */
'CHAR',              /* mask */
0,                   /* scale */
0,                   /* default_len */
'',                  /* default_val */
0,                   /*-delim_pre_len-*/
'',                  /* delim_pre */
0,                   /*-delim_post_len-*/
'',                  /* delim_post */
0,                   /* min_boundary_len */
'',                  /* min_boundary */
3,                   /* min_boundary_err_act */
0,                   /* max_boundary_len */
'',                  /* max_boundary_err_act */
0                    /* rowtype */
)
go
```

3 You *must* restart Replication Server after adding a new type.

4 In Replication Server, test the new type using the admin translate command:

```
admin translate, 'The quick brown fox jumped over the lazy dog.',
'char(255)', 'rs_oracle_udd'

go
Delimiter Prefix   Translated                  Value Delimiter Postfix

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

NULL                  The quick brown fox jumped over the lazy dog.  NULL
```

The new type has been defined correctly if the sentence was translated correctly.

Example

The following example demonstrates how to create a replication definition, using a new type defined in Replication Server. The following Oracle table and type definitions are used in the example:

- Oracle UDD object type name: NAME_T
- Oracle table name: USE_NAME_T
- Oracle table columns: PKEY INT, PNAME NAME_T

```
create replication definition use_name_t_repdef
```

```
with primary at ra_source_db.ra_source_ds
with all tables named 'USE_NAME_T'
(
   PKEY int,
   PNAME rs_oracle_udd
)
primary key (PKEY)
searchable columns (PKEY)
go
```

Note The ltl_character_case must be in uppercase for this example.

Mirror Replication Agent objects in the Oracle primary database

Note This section describes the schema and details of the Mirror Replication Agent objects for an Oracle database. For more general information, see the Mirror Activator *Administration Guide*.

Mirror Replication Agent creates objects in the Oracle primary database to assist with replication tasks.

The Mirror Replication Agent objects are created by invoking the pdb_init command with no keyword. When you invoke this command, Mirror Replication Agent generates a SQL script that contains the SQL statements for the objects created or modified in the primary database. This script is stored in the *partinit.sql* file in the *MA-15_0\inst_name\scripts\xlog* directory. The objects must be created before any primary database objects can be marked for replication.

Note The generated scripts are for informational purposes only and *cannot* be run manually to initialize the primary database. This is also true for the procedure marking and unmarking scripts that are generated when you use pdb_setrepproc. Scripts are no longer generated when marking and unmarking tables with pdb_setreptable.

Mirror Replication Agent object names

All objects created in the primary database to support replication are prefixed with a common value, identified by configuration property pdb_xlog_prefix:

- prefix – represents the one- to three-character string value of the pdb_xlog_prefix parameter (the default is ra_).
- xxx – represents an alphanumeric counter, a string of characters that is (or may be) added to a database object name to make that name unique in the database.

The value of the pdb_xlog_prefix parameter is the prefix string used in all Mirror Replication Agent object names.

The value of the pdb_xlog_prefix_chars parameter is a list of the non-alphanumeric characters allowed in the prefix string specified by pdb_xlog_prefix. This list of allowed characters is database-specific. For example, in Oracle, the only non-alphanumeric characters allowed in a database object name are the $, #, and _ characters.

❖ **To find the names of the objects created**

- At the Mirror Activator administration port, invoke the ra_helpsysinfo command with no keywords:

```
ra_helpsysinfo
```

The ra_helpsysinfo command returns a list of the transaction log base objects.

Table objects

Table 2-3 lists the tables that are considered Mirror Replication Agent objects.

Table 2-3: Mirror Replication Agent transaction log base tables

Table	Database name
Procedure-active table	*prefix*PROCACTIVE_[*xxx*]

Procedure objects

Table 2-4 lists the procedure objects that are considered Mirror Replication Agent objects. No permissions are granted when these procedures are created.

Table 2-4: Mirror Replication Agent marker procedures and shadow tables

Procedure/Table	Database name
Transaction log marker procedure	RS_MARKER[*xxx*]
Dump marker procedure	RS_DUMP[*xxx*]
Transaction log marker shadow table	*prefix*SH_RS_MARKER_[*xxx*]
Dump marker shadow table	*prefix*SH_RS_DUMP_[*xxx*]

Sequences

Table 2-5 lists the Oracle sequences that are considered Mirror Replication Agent base objects.

Table 2-5: Mirror Replication Agent sequences

Sequence	Database name
Assign procedure call	*prefix*PCALL_[*xxx*]

Understanding Oracle sequence replication

Similar to the support for table and procedure replication, Oracle Sequences are individually marked for replication using the new Replication Agent pdb_setrepseq command. For a description of the pdb_setrepseq command, refer to the Mirror Replication Agent *Reference Guide*.

Logging of Oracle Sequence information

Individual sequence changes are not logged in the Oracle database *log* file; however, changes to Oracle Sequences do impact (update) the Oracle sys.seq$ table. These changes do not occur with each new sequence value generated. Instead, the sys.seq$ table is updated periodically, based on sequence caching refresh activity or other system changes. The value stored in the sys.seq$ table for a sequence is the “next” value to be assigned “after” the existing cache of values has been exhausted.

For example, a newly created sequence starts with a value of 1, increments by 1, and has a cache value of 20. (These are all default values and can be customized.) The value stored in the sys.seq$ record for this new sequence is 21. This indicates that the “next” value to be used by the sequence, after the existing cache of 20 numbers is used, is 21. The record in sys.seq$ does not change until the sequence value hits 21. At that time, Oracle will cache the next 20 values for the sequence, and the sys.seq$ record will be updated to 41. It is this value (41), recorded in change to the sequences sys.seq$ record, that will be used for replication. The key point is to recognize that not every individual sequence update is recorded in the log and therefore is not available for replication.

Replicating sequence changes

When a sequence is marked for replication, changes to that sequence against sys.seq$ are captured and sent to Replication Server in the form of parameters passed to a procedure. The procedure (rs_update_sequence) must be installed at the standby site as part of system setup, as well as a function replication definition for that procedure. At the standby site, an implementation of rs_update_sequence will increment a same-named sequence until its value is equal to the value at the primary site. Scripts are provided with installation to create the rs_update_sequence stored procedure and function replication definition and are located as follows:

```
$SYBASE/RAX-15_0/scripts/
oracle_create_replicate_sequence_proc.sql
$SYBASE/RAX-15_0/scripts/
oracle_create_rs_sequence_repdef.sql
```

Performance considerations

Compared to the performance of incrementing a sequence at the primary database, particularly where sequence values are cached, the effort to increment the same sequence at the standby site may be less efficient. The stored procedure must dynamically determine the sequence to increment and must loop internally, incrementing the sequence until the primary value has been reached. The loop is required because there is no way to assign a specific value to a sequence.

Because the name of the sequence is passed as a parameter, Oracle cannot pre-compile the procedure for efficiency. With the addition of the looping activity required to properly increment the sequence, the performance of the solution may impact some environments where a large number of highly used sequences is the norm.

Sequence replication alternatives

If the performance of sequence replication is a concern, other alternatives to replication are available that support primary and standby use of the same sequence. These alternatives are currently suggested by Oracle and others interested in providing sequence coordination between multiple sites:

- Assuming that the sequence is being used to generate primary key values, the sequence at each site can be concatenated with something unique to the site. For example, use a sequence number concatenated with the database name, site name, or something similar. This technique allows each site to maintain a unique “range” of sequence of numbers. If each site has a unique range, there is no reason in sending (replicating) changes of one site's range to another site.
- Similar to concatenating, each site can obtain a different range of numbers by having different starting points, or increment values, for the same sequence. For example, the sequence at one site can start at 1 and increment by 2 to generate odd numbers (1, 3, 5), while the other site starts at 2 and generates even numbers (2, 4, 6). Again, by having a unique range, each site would avoid any need for replication.
- A third option is available to standby solutions, where the standby site is for read-only and does not access the sequence value until failover. Rather than continually replicating a sequence's value, the value of the sequence at the standby site can be updated as part of the failover tasks. *After* failover and *before* the standby allows connection to client applications, a script or procedure can query the last-used sequence value (based on the last table to use it for a primary key) and update or redefine the sequence once, based on that calculated value.

Marked procedures

Table 2-6 lists the Mirror Replication Agent objects that are created for each primary procedure that is marked for replication. These objects are created only when a procedure is marked for replication.

Table 2-6: Mirror Replication Agent objects for each marked procedure

Object	Database name
Shadow table	*prefix*SH_*xxx*

Marking a stored procedure

When you mark a stored procedure for replication, Mirror Replication Agent creates a shadow-row procedure for that stored procedure.

Mirror Replication Agent also modifies the marked stored procedure as follows:

- Inserts a new first step to execute the associated shadow-row procedure
- Inserts a new last step to again execute the shadow-row procedure with different parameters.

When you unmark an object that has been marked for replication, the transaction log objects that were created to facilitate the replication for that object are removed from the primary database.

When you mark a stored procedure for replication, Mirror Replication Agent creates the transaction log objects that capture the stored procedure invocation in the transaction log.

Note DDL replication must be disabled during the marking of stored procedures. Because marking of a stored procedure modifies that stored procedure, you must first disable DDL replication to prevent the marking modifications from replicating to the standby site.

Unmarking a stored procedure

When you unmark a stored procedure, Mirror Replication Agent removes the transaction log objects that were created when the stored procedure was marked.

Note DDL replication must be disabled during the unmarking of stored procedures.

Transaction log truncation

Mirror Replication Agent provides features for both automatic and manual log truncation.

Mirror Replication Agent provides two options for automatic transaction log truncation:

- Periodic truncation, based on a time interval you specify
- Automatic truncation whenever Mirror Replication Agent receives a new LTM Locator value from the primary Replication Server

You also have the option to switch off automatic log truncation. By default, automatic log truncation is enabled and is set to truncate the log whenever Mirror Replication Agent receives a new LTM locator value from the primary Replication Server.

When pdb_include_archives is set to true (the default) and pdb_remove_archives is set false, the Mirror Replication Agent does not do any online or archived transaction log truncation.

When pdb_include_archives is set to true (the default) and pdb_remove_archives is set true, the Mirror Replication Agent deletes already-processed archive *redo* logs from the pdb_archive_path location. The Mirror Replication Agent is not responsible for archiving online transaction logs.

Note Sybase recommends that you only configure the Mirror Replication Agent to remove archive log files if an additional archive log directory is used.

When the configuration parameter pdb_include_archives is set to false, Mirror Replication Agent performs online *redo* log truncation (either scheduled or manual) by issuing the alter system command with the archive log sequence keywords. The command uses the log sequence number of the *redo* log file whose contents have been processed by the Mirror Replication Agent and are ready to be archived.

Note The alter system command syntax in Oracle allows *redo* log files to be archived in addition to the single log sequence specified in the command. To avoid the possibility of unintentional archiving, Mirror Replication Agent only issues this command when it is processing the *redo* log file whose status is `CURRENT`.

Automatic transaction log truncation

You can specify the automatic truncation option you want (including none) by using the ra_config command to set the value of the truncation_type configuration parameter.

If you want to truncate the transaction log automatically based on a time interval, use the ra_config command to set the value of the truncation_interval configuration parameter.

Manual transaction log truncation

To truncate the Mirror Replication Agent transaction log manually, at any time, invoke the pdb_truncate_xlog command at the Mirror Replication Agent administration port.

APPENDIX A

Migration for Mirror Replication Agent for ASE

This appendix describes how to upgrade and downgrade to Mirror Replication Agent for ASE version 15.0.

Migrating from ASE 12.5 to ASE 15.0

Mirror Replication Agent for ASE migration to support Sybase ASE 12.5 to Sybase ASE 15.0 migration is a similar process to migrating from Mirror Replication Agent for ASE 12.6 to Mirror Replication Agent for ASE 15.0.

Note You need to verify that the Mirror Replication Agent is quiesced prior to migrating from Sybase ASE 12.5 to ASE 15.0, that is, the replication environment must have completed processing of all transactions prior to migrating. This precaution is needed because the Mirror Replication Agent moves the truncation point to the end of the log during Sybase ASE 12.5 to Sybase ASE 15.0 migration.

❖ To migrate from Sybase ASE 12.5 to Sybase ASE 15.0

1 Follow the steps that Sybase ASE provides in their documentation for upgrading from 12.5 to 15.0.

Note For the latest update refer to the *Release Bulletin* for the Mirror Replication Agent.

2 After migrating Sybase ASE, re-start the Mirror Replication Agent and issue the ra_migrate command.

Note If you are migrating from both Mirror Replication Agent for ASE 12.6 to Mirror Replication Agent for ASE 15.0, and from Sybase ASE 12.5 to 15.0, you need to migrate only once.

Migrating Mirror Replication Agent for ASE 12.6 to 15.0

This section describes the steps for migrating from Mirror Replication Agent for ASE version 12.6 to Mirror Replication Agent for ASE version 15.0.

❖ To migrate from Mirror Replication Agent for ASE 12.6 to 15.0

1 Sybase recommends that you back up the existing Mirror Replication Agent instances directory containing the instance configuration file. In addition, for Mirror Replication Agent for ASE, back up the Mirror Replication Agent System Database (RASD).

2 Shut down all of the Mirror Replication Agent version 12.6 instances.

3 Use the ma_admin utility script from the Mirror Replication Agent 15.0 installation's *bin* directory to upgrade all the verifiable Mirror Replication Agent version 12.6 instances. Execute the ma_admin utility script with the *-u* option:

```
ma_admin -u <src_directory>
```

where the *src_directory* is the full path name to the Mirror Replication Agent version 12.6 installation directory. For example:

- For UNIX:

```
/sybase15/MA-15_0/bin/ma_admin.sh -u /sybase/MRA-12_6
```

- For Windows:

```
d:\sybase15\MA-15_0\bin\ma_admin -u d:\sybase\MRA-12_6
```

This command will upgrade all valid Mirror Replication Agent instances.

For information about those instances that failed to upgrade, refer to the administration logs (*…/MA-15_0/admin_logs*). After you correct the problem, you can re-run this command. (This command will not upgrade again those Mirror Replication Agent instances that have already been successfully upgraded.)

4. Start Mirror Replication Agent version 15.0.
5. Log in to each Mirror Replication Agent instance and run the ra_migrate command first.

Note For additional information and a manual procedure for Sybase ASE 15.0 migration, refer to the *Release Bulletin* for Mirror Replication Agent 15.0.

6. You can now resume replication.

Downgrading from Mirror Replication Agent for ASE version 15.0 to version 12.6

This procedure assumes that you are using a Mirror Replication Agent for ASE version 15.0 instance. If the Mirror Replication Agent for ASE version 12.6 instance no longer exists, create one using the Mirror Replication Agent for ASE version 12.6 mra_admin command.

❖ To downgrade from Mirror Replication Agent for ASE version 15.0 to version 12.6

1. Prevent users (other than the Mirror Replication Agent for ASE version 15.0 user) from accessing the primary database.
2. Verify that the Mirror Replication Agent for ASE version 15.0 instance is in Replicating state and allow replication to finish. To verify that replication is completed:
   - Periodically issue the ra_statistics command, watching until the following statistics are zero (0):
     - Input queue size
     - Output queue size

- When they are both queue sizes are zero, note the Last QID Sent from the last set of statistics.
- Issue the ra_locator update command to allow Mirror Replication Agent to retrieve the truncation point from Replication Server.
- Wait, and then issue the ra_locator command and compare the displayed locator with that of the Last QID Sent. If they are different, wait and repeat this step.

3. Quiesce the Mirror Replication Agent for ASE version 15.0 instance.
4. In the Mirror Replication Agent for ASE version 15.0 instance, de-initialize the Mirror Replication Agent:

   ```
   ra_deinit, force
   ```

5. Shut down the Mirror Replication Agent for ASE version 15.0 instance.
6. Start and log in to the Mirror Replication Agent for ASE version 12.6 instance.
7. Test the primary database connection:

   ```
   test_connection PDS
   ```

8. Initialize the primary database:

   ```
   pdb_init
   ```

   Note Any configuration properties that you have modified after migration may need to be reset.

9. Quiesce the primary database:

   ```
   pdb_quiesce hold
   ```

10. Initialize the Mirror Replication Agent for ASE 12.6:

    ```
    ra_init
    ```

    Note This automatically marks all the objects for replication. Any objects that you do not want to mark for replication must be re-entered.

11. Allow users access to the primary database:

    ```
    pdb_quiesce release
    ```

12 Log in to the RSSD and set the Replication Server's locator to "0":

```
rs_zeroltm source_ds , source_db
```

where the source_ds and source_db match the Mirror Replication Agent for ASE version 12.6 instance values for the *rs_source_ds* and *rs_source_db* parameters.

Note The *rs_source_ds* and *rs_source_db* values were migrated from Mirror Replication Agent for ASE version 15.0 and should not be changed.

13 In the Mirror Replication Agent for ASE version 12.6 instance, resume replication.

14 Log out of the Mirror Replication Agent for ASE version 12.6 instance.

15 Revoke any additional privileges that were granted to the Mirror Replication Agent primary database user for the upgrade in the primary database.

APPENDIX B Migration for Mirror Replication Agent for Oracle

This appendix describes how to upgrade and downgrade to Mirror Replication Agent for Oracle version 15.0.

Migrating from Oracle 9i to Oracle 10g

Mirror Replication Agent for Oracle migration process to support Oracle 9i to Oracle 10g migration is similar to migrating from Mirror Replication Agent for Oracle 12.6 to Mirror Replication Agent for Oracle 15.0.

Note You need to verify that the Mirror Replication Agent is quiesced prior to migrating from Oracle 9i to Oracle 10g. That is, the replication environment must have completed processing of all transactions prior to migrating because the Mirror Replication Agent moves the truncation point to the end of the log during Oracle 9i to Oracle 10g migration.

❖ To migrate from Oracle 9i to Oracle 10g

1. Follow the steps that Oracle provides in their documentation for upgrading from Oracle 9i to Oracle 10g.
2. After migrating Oracle, you must re-start the Mirror Replication Agent and issue the ra_migrate command.

3 As with the 12.6 to 15.0 migration, you may have to re-configure the Mirror Replication Agent for Oracle instance to read archive logs depending on the configuration in Oracle, which may change following Oracle migration.

Note If you are migrating from both Mirror Replication Agent for Oracle 12.6 to Mirror Replication Agent for Oracle 15.0, and from Oracle 9i to Oracle 10g, you need to migrate only once.

Migrating from version 12.6 to 15.0

This section describes the steps for migrating from Mirror Replication Agent version 12.6 to Mirror Replication Agent version 15.0.

❖ **To migrate from Mirror Replication Agent 12.6 to 15.0**

1 Sybase recommends that you back up the existing Mirror Replication Agent instances directory containing the instance configuration file. In addition, for Mirror Replication Agent for Oracle, back up the Mirror Replication Agent System Database (RASD).

2 Shut down all of the Mirror Replication Agent version 12.6 instances.

3 Use the ma_admin utility script from the Mirror Replication Agent 15.0 installation's *bin* directory to upgrade all of the verifiable Mirror Replication Agent version 12.6 instances. Execute the ma_admin utility script with the *-u* option:

```
ma_admin -u <src_directory>
```

where the *src_directory* is the full path name to the Mirror Replication Agent version 12.6 installation directory. For example:

- For UNIX:

```
/sybase15/MA-15_0/bin/ma_admin.sh -u /sybase/MRO-12_6
```

- For Windows:

```
d:\sybase15\MA-15_0\bin\ma_admin -u d:\sybase\MRO-12_6
```

This command will upgrade all valid Mirror Replication Agent instances.

For information about those instances that failed to upgrade, refer to the administration logs (*…/MA-15_0/admin_logs*). After you correct the problem, you can re-run this command. (This command will not upgrade again those Mirror Replication Agent instances that have already been successfully upgraded.)

Be sure that your CLASSPATH is set properly for the Mirror Replication Agent version 15.0 instances. For Mirror Replication Agent for Oracle, use *ojdbc14.jar*.

4 Start Mirror Replication Agent version 15.0.

5 Log in to each Mirror Replication Agent instance and run the ra_migrate command first.

6 If you do not want to have automatic archiving turned on (a version 12.6 default), then pdb_include_archives in version 15.0 needs to be set to false. If automatic archiving is true (a version 15.0 default), then pdb_archive_path needs to be set and the pdb_archive_remove needs to be false (a version 15.0 default) for each Mirror Replication Agent instance. For example:

```
ra_config pdb_archive_path, <log path on the rep
agent setup machine>
```

7 You can now resume replication.

Downgrading from Mirror Replication Agent for Oracle version 15.0 to version 12.6

This procedure assumes that you are using a Mirror Replication Agent for Oracle version 15.0 instance. If the Mirror Replication Agent for Oracle version 12.6 instance no longer exists, create one using the Mirror Replication Agent for Oracle version 12.6 mro_admin command.

❖ To downgrade from Mirror Replication Agent for Oracle version 15.0 to version 12.6

1 Be sure that the appropriate Oracle JDBC driver is in your CLASSPATH.

Note No other Oracle drivers are allowed in the CLASSPATH.

2 Prevent users (other than the Mirror Replication Agent for Oracle version 15.0 user) from accessing the primary database.

3 Verify that the Mirror Replication Agent for Oracle version 15.0 instance is in Replicating state and allow replication to finish. To verify that replication is completed:

- Periodically issue the ra_statistics command, watching until the following statistics are zero (0):
    - Input queue size
    - Output queue size
- When they are both queue sizes are zero, note the Last QID Sent from the last set of statistics.
- Issue the ra_locator update command to allow Mirror Replication Agent to retrieve the truncation point from Replication Server.
- Wait, and then issue the ra_locator command and compare the displayed locator with that of the Last QID Sent. If they are different, wait and repeat this step.

4 Quiesce the Mirror Replication Agent for Oracle version 15.0 instance.

5 In the Mirror Replication Agent for Oracle version 15.0 instance, de-initialize the Mirror Replication Agent:

```
ra_deinit, force
```

6 Shut down the Mirror Replication Agent for Oracle version 15.0 instance.

7 Start and log in to the Mirror Replication Agent for Oracle version 12.6 instance.

8 Test the primary database connection:

```
test_connection PDS
```

9 Initialize the primary database:

```
pdb_init
```

Note Any configuration properties that you have modified after migration may need to be reset.

10 Quiesce the primary database:

```
pdb_quiesce hold
```

11 Initialize the Mirror Replication Agent for Oracle 12.6:

```
ra_init
```

Note This automatically marks all the objects for replication. Any objects that you want to unmark must be re-entered.

---

12 Allow users access to the primary database:

```
pdb_quiesce release
```

13 Log in to the RSSD and set the Replication Server's locator to "0":

```
rs_zeroltm source_ds, source_db
```

where *source_ds* and *source_db* match the Mirror Replication Agent for Oracle version 12.6 instance's values for the *rs_source_ds* and *rs_source_db* parameters.

Note The *rs_source_ds* and *rs_source_db* values were migrated from Mirror Replication Agent for Oracle version 15.0 and should *not* be changed.

---

14 In the Mirror Replication Agent for Oracle version 12.6 instance, resume replication:

15 Log out of the Mirror Replication Agent for Oracle version 12.6 instance.

16 If you created a new *interfaces* or *sql.ini* entry when you upgraded to Mirror Replication Agent for Oracle version 15.0, update the entry so the Mirror Replication Agent for Oracle version 12.6 instance name is again associated with the old Mirror Replication Agent for Oracle version 12.6 instance machine and port number.

17 Revert the Oracle logging properties back to your desired setup in the primary database.

18 Revoke any additional privileges that were granted to the Mirror Replication Agent primary database user for the upgrade in the primary database.

Glossary

This glossary describes Replication Server—Heterogeneous Replication Options 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*, which 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 standalone 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.

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,* 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*, which 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 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 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 Replication Server—Heterogeneous Replication Options 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*, 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**.

RASD An abbreviation for *Replication Agent System Database*. Information in the RASD is used by the primary database to recognize database structure or schema objects in the transaction log.

RCL An abbreviation for *Replication Command Language*, the command language used to manage Replication Server.

RDBMS An abbreviation for *relational database management system*, 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**.

Mirror Replication Agent

An application that reads a primary database transaction log to acquire information about data-changing transactions in the primary database, processes the log information, and then sends it to a Replication Server for distribution to a standby database. See also **primary database** 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 **Mirror Replication Agent**.

RSSD

An abbreviation for *Replication Server System Database*, which 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*, 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 system of local-area networks (LANs) connected together with data communication lines. Contrast with **LAN**.

Index