SYBASE®

Primary Database Guide

Mirror Replication Agent™

15.1

Linux, Microsoft Windows, and UNIX

DOCUMENT ID: DC00712-01-1510-01

LAST REVISED: September 2008

Contents

About This Book

Mirror Replication Agent™ version 15.1 extends the capabilities of Replication Server® by supporting Sybase® and non-Sybase primary data servers in a Sybase disaster recovery 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)
- Microsoft SQL Server
- 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 Microsoft SQL Server,” describes replication system issues that are specific to Microsoft SQL Server, and details of Mirror Replication Agent for Microsoft SQL Server.

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

Chapter 3, “Mirror Replication Agent for ASE,” describes replication system issues that are specific to Adaptive Server Enterprise and details of Mirror Replication Agent for Adaptive Server Enterprise.

Appendix A, "Upgrading Mirror Replication Agent," describes Mirror Replication Agent upgrades.

Appendix B, "Using the sybfilter driver," describes use of the sybfilter driver.

Related documents

A Sybase replication system consists of several components. You may find it helpful to have the following documentation available.

Mirror Replication Agent

- The Mirror Activator *Administration Guide* introduces replication concepts and Sybase replication technology. This document also describes Mirror Replication Agent features and operations, and how to set up, administer, and troubleshoot the Mirror 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 Activator 15.1 *Release Bulletin* and Mirror Replication Agent 15.1 *Release Bulletin* contain 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 Product Manuals Web site.

Java environment

Mirror Activator 15.1 installs and uses a Java Runtime Environment (JRE) on the machine that acts as the Mirror Replication Agent host.

- The Mirror Activator 15.1 *Release Bulletin* and Mirror Replication Agent 15.1 *Release Bulletin* contain 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.

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 data servers supported by Mirror Replication Agent:

- Adaptive Server Enterprise
- Microsoft SQL Server
- Oracle

Adaptive Server Enterprise

If your replication system includes databases in Sybase 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 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:

```
pdb_setreptable authors, mark
```

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

  Specify the value of *table_name* 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:

```
pdb_setreptable table_name, mark
```

  Here, *table_name* is the variable you should replace.

- In the regular text of this document:

- Names of programs, utilities, procedures, and commands appear like this:

  Use the pdb_setreptable 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_1/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, PDB_XLOG, Pdb_Xlog, and pdb_xlog 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_setreptable "TableName", mark
```

Accessibility features

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

Mirror Replication Agent 15.1 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 Mirror Replication Agent 15.1, see Sybase Accessibility at http://www.sybase.com/detail?id=1028493.

If you need help

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

CHAPTER 1

Mirror Replication Agent for Microsoft SQL Server

The term “Mirror Replication Agent for Microsoft SQL Server” refers to an instance of the Mirror Replication Agent 15.1 software installed and configured for a primary database that resides in a Microsoft SQL Server data server.

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

Note For information on the basic functionality of Mirror Replication Agent 15.1, see the Mirror Activator *Administration Guide* and the Mirror Replication Agent *Reference Manual*.

Microsoft SQL Server-specific considerations

This section describes general issues and considerations that are specific to using Mirror Replication Agent 15.1 with the Microsoft SQL Server data server.

Mirror Replication Agent for Microsoft SQL Server reads a mirrored copy of the Microsoft SQL Server primary database log. To read the database log, Mirror Replication Agent must be installed where it can directly access the log files. Because the machine on which Mirror Replication Agent is installed must have the same hardware and operating system as the machine on which the primary database resides, Mirror Replication Agent for Microsoft SQL Server is available only on the Microsoft Windows platform. In this chapter, the term "Windows" refers to all supported Microsoft Windows platforms. For a complete list of supported platforms, see the Mirror Activator *Release Bulletin*.

The following topics are included in this section:

- Microsoft SQL Server requirements
- DDL Replication
- Mirror Replication Agent connectivity
- Mirror Replication Agent permissions
- The sybfilter driver
- Initialization of the primary data server and Mirror Replication Agent
- Microsoft isql tool
- Character case of database object names
- Format of origin queue ID
- Datatype compatibility
- Replicating ntext datatypes

Microsoft SQL Server requirements

Observe the following requirements for Microsoft SQL Server:

- Mirror Replication Agent supports only Microsoft SQL Server 2005 Service Pack 2 and later, and the database compatibility level must be set to "SQL Server 2005 (90)".
- You cannot simultaneously use Microsoft replication and Mirror Replication Agent on the same Microsoft SQL Server database. Be sure to disable Microsoft replication before using Mirror Replication Agent for Microsoft SQL Server.
- Make sure Microsoft SQL Server is configured to allow a remote dedicated administrative connection (DAC).

- You cannot create a Microsoft SQL Server publication on the primary database where Mirror Replication Agent for Microsoft SQL Server is running.
- The Microsoft SQL Server TCP/IP protocol must be enabled.

DDL Replication

Replication of Data Definition Language (DDL) commands is supported.

Note No translation or adjustment of DDL commands is provided by Mirror Replication Agent. DDL commands should therefore only be replicated to other Microsoft SQL Server databases.

Replication of DDL commands is enabled or disabled in Mirror Replication Agent using the pdb_setrepddl command. For details on this command, see the Mirror Replication Agent *Reference Manual*.

Replication Server uses the ddl_username parameter to execute DDL commands in the standby database as the same user who executed the DDL commands in the primary database.

Setting *ddl_username* and *ddl_password*

To replicate DDL in Microsoft SQL Server, 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 *standby* database user name included in the log transfer language (LTL) for replicating DDL commands to the standby or target database.

Permissions

In addition to the permission to execute all replicated DDL commands at the standby database, the ddl_username should also have the impersonate permission granted for all users whose DDL commands may be replicated to the standby database. This impersonate permission is necessary to switch session context in the standby database when executing a DDL command. This user switches context to apply the DDL command using the same privileges and default schema settings as the user who executed the DDL command at the primary database. To provide this context switch, the ddl_username user must have permission to execute the execute as user Microsoft SQL Server command for any user who might execute DDL commands to be replicated from the primary database.

For example, user1 with a default schema of schema1 executes the following DDL at the primary database:

```
create table tab1 (id int)
```

This results in the creation of a table named schema1.tab1 at the primary database. At the standby database, user2 with a default schema of schema2, cannot immediately execute this DDL because it will generate a table named schema2.tab1. Therefore, user2, whose name is specified by the ddl_username configuration parameter, must first execute the following command at the standby database to impersonate user1:

```
execute as user = 'user1'
```

The DDL can then be executed with the correct schema by user2 at the standby database, generating a table named schema1.tab1.

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

Granting impersonate permission

There are two ways to grant impersonate permission to the ddl_username user:

- You can grant database owner permission to the to the ddl_username user. In doing this, you implicitly grant impersonate permission.
- Alternately, you can grant impersonate permission explicitly with the following Microsoft SQL Server command:

  ```
  GRANT IMPERSONATE ON USER::user1 TO ddl_user
  ```

  Here, *user1* is a user whose DDL is expected to be replicated to the standby database, and *ddl_user* is the ddl_username user.

Note This grant command must be executed in the *standby* database, where the user defined to ddl_username executes the DDL commands.

When you replicate DDL in Microsoft SQL Server, you must use Microsoft SQL Server as the standby database. You *cannot* replicate DDL commands from Microsoft SQL Server to non-Microsoft SQL Server 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 database-scope DDL commands are not replicated:

ALTER_APPLICATION_ROLE
ALTER_ASSEMBLY
ALTER_AUTHORIZATION_DATABASE
ALTER_CERTIFICATE
CREATE_APPLICATION_ROLE
CREATE_ASSEMBLY
CREATE_CERTIFICATE
CREATE_EVENT_NOTIFICATION
DROP_EVENT_NOTIFICATION

The following server-scope DDL commands are not replicated:

ALTER_AUTHORIZATION_SERVER
ALTER_DATABASE
ALTER_LOGIN
CREATE_DATABASE
CREATE_ENDPOINT
CREATE_LOGIN
DENY_SERVER
DROP_DATABASE
DROP_ENDPOINT
DROP_LOGIN
GRANT_SERVER
REVOKE_SERVER

Note It is not safe to replicate these DDL commands because they contain password information.

Any object owned by users defined in the list of non-replicated users is not replicated. You can modify this list using the pdb_ownerfilter command. In addition, Sybase has provided a default list of owners whose objects will not be replicated. 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.

Mirror Replication Agent connectivity

Mirror Replication Agent for Microsoft SQL Server uses the Java Database Connectivity (JDBC) protocol for communications with all replication system components.

Mirror Replication Agent connects to Microsoft SQL Server using the Microsoft SQL Server JDBC driver. You must download and install it on the Mirror Replication Agent host machine, and the directory where the JDBC driver is installed must be in the CLASSPATH environment variable.

For more information about Mirror Replication Agent connectivity, see the Mirror Activator *Administration Guide*.

Mirror Replication Agent permissions

Mirror Replication Agent for Microsoft SQL Server must create database objects to assist with replication tasks in the primary database.

The user ID that the Mirror Replication Agent instance uses to log in to the Microsoft SQL Server must have access to the primary database with the following permissions granted:

- create table – required to create tables in the primary database.
- create trigger – required to create DDL triggers in the primary database.
- create procedure – required to create procedures in the primary database.
- db_owner role – required to allow Mirror Replication Agent to execute sp_repltrans and sp_repldone in the primary database. This role is also required for primary database initialization.
- sysadmin role – required for Microsoft SQL Server data server initialization and deinitialization (using pdb_init, ra_init, and ra_deinit, respectively).

The sybfilter driver

Mirror Replication Agent must be able to read the Microsoft SQL Server log files. However, the Microsoft SQL Server process opens these log files with exclusive read permission, and the file cannot be read by any other processes, including Mirror Replication Agent. Before Mirror Replication Agent can replicate data, you must use the sybfilter driver to make the log files readable.

For the sybfilter driver to work properly, the Microsoft Filter Manager Library must be version 5.1.2600.2978 or later. To determine the version of the library, right-click *c:\windows\system32\fltlib.dll* in Windows Explorer, select Properties, and click the Version or Details tab in the Properties dialog. If the version is earlier than 5.1.2600.2978, go to the Microsoft Web site at http://windowsupdate.microsoft.com, and update your Windows system.

For details on installing and using the sybfilter driver, see Appendix B, "Using the sybfilter driver."

Initialization of the primary data server and Mirror Replication Agent

For Microsoft SQL Server initialization, Mirror Replication Agent for Microsoft SQL Server installs objects at both the data server and database level. The data server-level modifications are only required once. However, to make the server-level modifications, additional permission are required, the pds_dac_port_number parameter is used, and the primary database must be in standalone mode. Subsequent executions of pdb_init do not modify the server again and do not require the additional permission or configurations.

Use the following procedures for initialization and cleanup tasks.

First-time initialization

You must initialize the primary Microsoft SQL Server data server so that Mirror Replication Agent can open the supplemental log of a table or procedure that is marked for replication. You need to do this only once for a given primary data server.

❖ To initialize the primary data server and Mirror Replication Agent for the first time

1 Make sure Microsoft SQL Server is configured to allow a remote DAC by logging in to the Microsoft SQL Server as sysadmin and executing the following SQL commands:

```
sp_configure 'remote admin connections',1
go
RECONFIGURE
go
```

Alternately, you can use the Microsoft SQL Server Surface Area Configuration tool to enable a remote DAC:

   a From the Windows Start menu, choose Microsoft SQL Server | Surface Area Configuration | Configuration Tools | SQL Server Surface Area Configuration | Surface Area Configuration for Features.

   b In the Surface Area Configuration for Features window, choose DAC under MSSQLSERVER/Database Engine, and make sure the Enable remote DAC check box is selected.

2. Determine the primary Microsoft SQL Server DAC port number.

   a Open the *ERRORLOG* file in a text editor. This file is located in the log directory of your Microsoft SQL Server. For example:

      C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

   b Search for the string "Dedicated admin" to find an entry similar to the following:

      ```
      2007-11-09 13:40:02.40 Server Dedicated admin
      connection support was established for listening
      locally on port 1348.
      ```

   c Record the port number specified in this entry.

3. Log in to your Mirror Replication Agent, and set the pds_dac_port_number configuration parameter:

   ```
   ra_config pds_dac_port_number, port
   ```

   Here, *port* is the DAC port number you recorded.

4. Also configure the following Mirror Replication Agent connectivity parameters for the Microsoft SQL Server primary database:

   pds_server_name
   pds_database_name
   pds_username
   pds_password
   pds_port_number

   For information about these configuration parameters, see the Mirror Replication Agent *Installation Guide* and *Reference Manual*.

5. Stop the Microsoft SQL Server service.

   a From the Windows Control Panel, go to Administrative Tools | Services, and find the service named SQL Server (*SERVER*). Here *SERVER* is the name of your Microsoft SQL Server data server.

b. Stop this service.

6. Open a command window, and restart Microsoft SQL Server in single-user mode:

   ```
   "C:\Program Files\Microsoft SQL
   Server\MSSQL.1\MSSQL\Binn\sqlservr.exe” -m -s
   instanceName
   ```

   Here, *instanceName* is the name of the Microsoft SQL Server instance.

7. Make sure that there are no other connections to the primary database, and verify that Mirror Replication Agent can connect to the primary database:

   ```
   test_connection PDS
   ```

8. Initialize the Microsoft SQL Server data server and Mirror Replication Agent:

   ```
   pdb_init
   ```

   In the primary database, Mirror Replication Agent creates all the tables, procedures, and triggers described in “Mirror Replication Agent objects in the Microsoft SQL Server primary database” on page 18. The sp_SybSetLogforReplTable, sp_SybSetLogforReplProc, sp_SybSetLogforLOBCol procedures are created in the mssqlsystemresource database with execute permission granted to Public.

9. Stop the Microsoft SQL Server in single-user mode in either of the following ways:

   - Use the sqlcmd utility to log in to the server:

     ```
     "C:\Program Files\Microsoft SQL
     Server\90\Tools\Binn\SQLCMD.EXE" -U username -P
     password -S serverName
     ```

     Here, *username*, *password*, and *serverName* are your user ID, your password, and the name of the Microsoft SQL Server.

     After you have logged in to the Microsoft SQL Server, use the shutdown command.

   - In the Microsoft SQL Server server Start window, type Ctrl+C.

10. Restart Microsoft SQL Server in multi-user mode (normal start).

    a. From Windows Control Panel, go to Administrative Tools | Services, and find the service named SQL Server (*SERVER*). Here *SERVER* is the name of your Microsoft SQL Server data server.

    b. Start this service.

If you want to start other services, such as the Microsoft SQL Server Agent service, the Microsoft SQL Server Analysis Services, or the Microsoft SQL Server Reporting Services, you can start these services now.

Subsequent initialization

If you have initialized Mirror Replication Agent for the first time, have subsequently de-initialized Mirror Replication Agent using ra_deinit, and want to re-initialize this Mirror Replication Agent instance or another Mirror Replication Agent instance for a different database in the same primary data server, use the following procedure.

❖ **To subsequently initialize Mirror Replication Agent instances**

1. Determine the primary Microsoft SQL Server DAC port number, and make sure Microsoft SQL Server is configured to allow a remote DAC. You can use the Microsoft SQL Server Surface Area Configuration tool to enable a remote DAC:

   a. From the Windows Start menu, choose Microsoft SQL Server | Surface Area Configuration | Configuration Tools | SQL Server Surface Area Configuration | Surface Area Configuration for Features.

   b. In the Surface Area Configuration for Features window, choose DAC under MSSQLSERVER/Database Engine, and make sure the Enable Remote DAC check box is selected.

2. Log in to your Mirror Replication Agent, and set the pds_dac_port_number configuration parameter.

3. Configure the following Mirror Replication Agent connectivity parameters for the Microsoft SQL Server primary database:

   pds_server_name
   pds_database_name
   pds_username
   pds_password

   For information about these configuration parameters, see the Mirror Replication Agent *Installation Guide* and *Reference Manual*.

4. Verify that Mirror Replication Agent can connect to the primary database:

   ```
   test_connection PDS
   ```

5 Initialize the Microsoft SQL Server data server and Mirror Replication Agent:

```
ra_init
```

Final cleanup

If you have removed all Mirror Replication Agent objects from all the databases on a given primary data server by issuing ra_deinit in each database in which you had issued ra_init, and you want to remove all the remnants of Mirror Replication Agent, use the following procedure to completely clean the primary data server.

❖ To clean up all Mirror Replication Agent remnants from the primary data server

1 Stop the Microsoft SQL Server service.

   a From the Windows Control Panel, go to Administrative Tools | Services, and find the service named SQL Server (*SERVER*). Here *SERVER* is the name of your Microsoft SQL Server data server.

   b Stop this service.

2 Open a command window, and restart Microsoft SQL Server in single-user mode:

```
"C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Binn\sqlservr.exe” -m -s
instanceName
```

Here, *instanceName* is the name of the Microsoft SQL Server instance.

3 Make sure the Microsoft SQL Server SQL Browser service is running, and connect to the data server using the sqlcmd utility with -A option or using the Management Studio. Specify the server name as `Admin:`*servername*. Here, *servername* is the name of your data server.

4 Remove the pds_username user if it has been created for Mirror Replication Agent:

```
drop user pds_username
```

5 Remove the special marking procedures from the mssqlsystemresource database:

```
drop procedure sp_SybSetLogforLOBCol;

drop procedure sp_SybSetLogforReplTable;

drop procedure sp_SybSetLogforReplProc;
```

6. Stop Microsoft SQL Server in single-user mode by shutting down the Windows service or by issuing the shutdown command with the sqlcmd utility.
7. To undo the affects of the sybfilter driver on each of the log devices, remove the log path entry by editing the configuration file or by using the sybfilter manager console.

   For information on using the sybfilter manager console, see Appendix B, “Using the sybfilter driver.”
8. Restart Microsoft SQL Server in multi-user mode (normal start).
   a. From Windows Control Panel, go to Administrative Tools | Services, and find the service named SQL Server (*SERVER*). Here, *SERVER* is the name of your Microsoft SQL Server data server.
   b. Start this service.

Microsoft isql tool

The database access tool provided with Microsoft SQL Server is Microsoft sqlcmd. You should use Microsoft sqlcmd (or a compatible tool) to access the Microsoft SQL Server database to execute some of the test scripts documented in this chapter.

However, if you are using the now deprecated Microsoft isql tool, you should be aware that the Sybase isql and Microsoft isql tools are not compatible. For example, you cannot use the Sybase isql tool to access the Microsoft SQL Server data server, and you cannot use the Microsoft isql tool to access the Mirror Replication Agent administration port.

Also, if you have both Sybase and Microsoft isql tools loaded on the same computer, you may need to change an environment variable (possibly the PATH variable) to avoid problems when you invoke one of the isql tools.

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 uppercase, then that table name must appear in all uppercase when it is sent to the primary Replication Server by Mirror Replication Agent.

To specify the character case option you want, set the value of the ltl_character_case configuration parameter to one of the following three options:

- 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 Microsoft SQL Server, database object names are stored in the same case as entered (uppercase and/or lowercase). Therefore, you must use the asis option to send database object names to the primary Replication Server in the same case as they are stored in Microsoft SQL Server.

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 Microsoft SQL Server.

Table 1-1: Mirror Replication Agent for Microsoft SQL Server origin queue ID

Character	Bytes	Description
0-3	2	Database generation ID
4-11	4	Virtual file sequence number
12-19	4	Page start offset
20-23	2	Operation number
24-31	4	Available for specifying uniqueness
32-39	4	Oldest active transaction: virtual file sequence number
40-47	4	Oldest active transaction: page start offset
48-51	2	Oldest active transaction: operation number
52-59	4	Latest committed transaction: page start offset
60-63	2	Latest committed transaction: operation number

Datatype compatibility

Mirror Replication Agent processes Microsoft SQL Server transactions and passes transaction information to the primary Replication Server.

The primary Replication Server uses the datatype formats specified in the replication definition to receive the data from Mirror Replication Agent.

Table 1-2 describes the default conversion of Microsoft SQL Server datatypes to Sybase Replication Server datatypes.

Table 1-2: Microsoft SQL Server to Replication Server default datatype mapping

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
bit	Integer with value of 0 or 1	bit	Integer with value of 0 or 1	
bigint	-2^{63} to 2^{63} - 1	bigint	-2^{63} to 2^{63} - 1	
int	-2^{31} to 2^{31} - 1	int	-2^{31} to 2^{31} - 1	
smallint	Integer with value from -2^{15} to 2^{15} - 1	smallint	Integer with value from -2^{15} to 2^{15} - 1	
tinyint	Integer with value from 0 to 255	tinyint	Integer with value from 0 to 255	
decimal	Numeric from -10^{38} to 10^{38} - 1	decimal	Numeric from -10^{38} to 10^{38} - 1	

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
numeric	Synonym for decimal datatype	numeric	Synonym for decimal datatype	
money	Monetary from -2^{63} to $2^{63} - 1$	money	Monetary from -2^{63} to $2^{63} - 1$	
smallmoney	Monetary from -214,748.3648 to 214,748.3647	smallmoney	Monetary from -214,748.3648 to 214,748.3647	
float	Floating precision from -1.79E + 308 to 1.79E + 308	float	Floating precision from -1.79E + 308 to 1.79E + 308	Results in Sybase are machine dependent.
real	Floating precision from -3.40E + 38 to 3.40E + 38	real	Floating precision from -3.40E + 38 to 3.40E + 38	Results in Sybase are machine dependent.
datetime	Date and time from 01/01/1753 to 12/31/9999	datetime	Date and time from 01/01/1753 to 12/31/9999	
smalldatetime	Date and time from 01/01/1900 to 06/06/2079	datetime	Date and time from 01/01/1900 to 06/06/2079	
timestamp	Database-wide unique number	timestamp or varbinary	Database-wide unique number	For replication to Replication Server 15.0 and earlier versions, the Sybase datatype should be varbinary(8). For replication to Replication Server 15.1 or later, the Sybase datatype should be timestamp.
uniqueidentifier	Globally unique identifier	char	Globally unique identifier	No Sybase equivalent. Map to char(38).
char	Fixed length up to 8000 characters	char	32K	
varchar	Variable length up to 8000 characters	varchar	32K	
varchar(max)	Variable length up to $2^{31} - 1$ characters	text	2GB	
text	Variable length up to $2^{31} - 1$ characters	text	2GB	

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
nchar	Fixed length Unicode up to 4000 characters	unichar or char	32K	Actual maximum length is @@ncharsize * number of characters.
nvarchar	Variable length Unicode up to 4000 characters	univarchar or varchar	32K	Actual maximum length is @@ncharsize * number of characters.
nvarchar(max)	Variable length Unicode up to 2^{30} - 1 characters	unitext or image	2GB	For Replication Server 15.0 and later versions, nvarchar(max) maps to unitext. For earlier versions of Replication Server, nvarchar(max) maps to image.
ntext	Variable length Unicode up to 2^{30} - 1 characters	unitext or image	2GB	For Replication Server 15.0 and later versions, ntext maps to unitext. For earlier versions of Replication Server, ntext maps to image.
binary	Fixed length up to 8000 bytes	binary	32K	
varbinary	Variable length up to 8000 bytes	varbinary	32K	
image	Variable length up to 2^{31} - 1 bytes	image	2GB	
sql_variant	Any datatype except text, ntext, timestamp, and sql_variant, up to 8000 bytes	varchar or opaque	32K	For replication to Replication Server 15.0 and earlier versions, the Sybase datatype should be varchar. For replication to Replication Server 15.1 or later, the Sybase datatype should be opaque.

Replication Server 15.0 unsigned datatype mapping

For Replication Server 15.0 and later, unsigned datatypes are supported and can be specified in the replication definitions.

For versions of Replication Server earlier than 15.0, these datatypes cannot be specified. Table 1-3 identifies the replication definition datatypes that should be used.

Table 1-3: Unsigned integer replication definition datatype mapping

RepServer 15.0 unsigned datatypes	Replication definition datatypes
unsigned bigint	NUMERIC (20)
unsigned int	NUMERIC (10)
unsigned smallint	INT
unsigned tinyint	TINYINT

Replicating *ntext* datatypes

Microsoft SQL Server expects double-byte ntext datatype values to be sent to the server in little-endian byte order, which is the native byte order of the Windows platform. By default, the byte order of ntext data is converted to the little-endian byte order during replication to meet this Microsoft SQL Server expectation. However, if your target database does not expect Unicode values to be sent in little-endian byte order, you can force the Unicode byte order sent during replication by setting the lr_ntext_byte_order property to big (for big-endian) or little (for little-endian) to meet the expectation of your standby database.

Note The default behavior of Mirror Replication Agent for Microsoft SQL Server is to force any Unicode data to big-endian order as defined by the ltl_big_endian_unitext configuration property. To allow the lr_ntext_byte_order configuration property to successfully override the Microsoft SQL Server byte order, you must also set ltl_big_endian_unitext configuration property to false whenever the lr_ntext_byte_order property is used.

The ltl_big_endian_unitext parameter specifies whether unitext data should be converted from little-endian to big-endian before sending LTL to Replication Server. Valid values are true and false. When setting this parameter, you must know how the lr_ntext_byte_order parameter is set. If the lr_ntext_byte_order parameter is set to send the correct byte order for the standby database, the ltl_big_endian_unitext parameter must be set to false so that the byte order is not changed.

The ltl_big_endian_unitext and lr_ntext_byte_order configuration properties have important differences. The ltl_big_endian_unitext property is false by default. When the ltl_big_endian_unitext property is true, Mirror Replication Agent for Microsoft SQL Server ensures all Unicode data is sent in big-endian order. When the ltl_big_endian_unitext property is false, Mirror Replication Agent for Microsoft SQL Server allows Unicode data to be sent in whatever byte order is used when the data is stored in the transaction log file. The lr_ntext_byte_order property forces the result of Unicode data read from the transaction log to be in the requested byte order, regardless of how it normally exists in the transaction log file.

Mirror Replication Agent objects in the Microsoft SQL Server primary database

Note This section describes the Mirror Replication Agent objects created for a Microsoft SQL Server database. For more general information, see the Mirror Activator *Administration Guide*.

Mirror Replication Agent creates objects in the Microsoft SQL Server primary database to assist with replication tasks.

The Mirror Replication Agent objects are created by invoking the pdb_init command. When you invoke this command, Mirror Replication Agent generates a SQL script that contains the SQL statements for the objects created or modified in the primary database. This script is stored in the *partinit.sql* file in the *MA-15_1\inst_name\scripts\xlog\installed* directory. The objects must be created before any primary database objects can be marked for replication.

Note The generated scripts are for informational purposes only and *cannot* be run manually to initialize the primary database or Mirror Replication Agent.

Mirror Replication Agent object names

There are two variables in the transaction log component database object names shown in this chapter:

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

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

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

You can use the ra_helpsysinfo command to view the names of Mirror Replication Agent transaction log components in the primary database.

See the Mirror Activator *Administration Guide* for details on setting up log object names.

Table objects

Table 1-4 lists the tables that are considered Mirror Replication Agent objects. Insert and delete permissions are granted to Public only on the DDL shadow table. No permissions are granted on the other tables.

Table 1-4: Mirror Replication Agent table objects

Table	**Database name**
DDL shadow table	*prefix*ddl_trig_*xxx*
Object marking table	*prefix*markObject_*xxx*
Object verifying table	*prefix*checkObject_*xxx*

Procedure objects

Table 1-5 lists the procedure objects that are considered Mirror Replication Agent objects. The sp_SybSetLogforReplTable, sp_SybSetLogforReplProc, and sp_SybSetLogforLOBCol procedures are created in the Microsoft SQL Server mssqlsystemresource system database. Although execute permission on these procedures is granted to Public, only the Mirror Replication Agent pds_username user is able to successfully execute the procedures because only the pds_username user is granted select permission on the sys.sysschobjs table. No permissions are granted on the other procedures when they are created.

Note The stored procedures listed in Table 1-5 have no effect when executed outside the context of replication.

Table 1-5: Mirror Replication Agent procedure objects

Procedure/Table	**Database name**
Marks/unmarks an object	*prefix*mark_*xxx*
Verifies an object	*prefix*check_*xxx*
Retrieves the ID of the last committed transaction	*prefix*lct_sql_*xxx*
Marks/unmarks a table	sp_SybSetLogforReplTable
Marks/unmarks a procedure	sp_SybSetLogforReplProc
Marks/unmarks LOB column	sp_SybSetLogforLOBCol

Marker objects

Table 1-6 lists the marker procedures and marker shadow tables that are considered Mirror Replication Agent objects. No permissions are granted when these procedures and tables are created.

Table 1-6: Mirror Replication Agent marker objects

Procedure/Table	Database name
Transaction log marker procedure	rs_marker_*xxx*
Dump marker procedure	rs_dump_*xxx*
Transaction log marker shadow table	*prefix*rs_markersh_*xxx*
Dump marker shadow table	*prefix*rs_dumpsh_*xxx*

Trigger objects

Table 1-7 lists Mirror Replication Agent trigger objects.

Table 1-7: Mirror Replication Agent trigger objects

Object	Database name
Captures DDL commands	*prefix*ddl_trig_*xxx*
Captures create_table DDL commands	*prefix*createtable_trig_*xxx*

Administering the transaction log

The only transaction log administration required is backing up the transaction log and truncation.

Backing up and restoring the transaction log

Mirror Replication Agent does not support backing up and restoring the transaction log automatically. Instead, Sybase recommends that you use the database backup utilities provided with your Microsoft SQL Server software to periodically back up the transaction log.

Note Mirror Replication Agent does not support replaying transactions from a restored log.

Truncating the transaction log

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 switched off.

To specify the automatic truncation option you want (including none), use the ra_config command to set the value of the truncation_type configuration parameter.

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.

At any time, you can truncate the Mirror Replication Agent transaction log manually by invoking the pdb_truncate_xlog command at the Mirror Replication Agent administration port.

To truncate the transaction log at a specific time, use a scheduler utility to execute the pdb_truncate_xlog command automatically.

Mirror Replication Agent for Microsoft SQL Server truncates the primary database log in units of transactions. After Mirror Replication Agent for Microsoft SQL Server receives the LTM locator from Replication Server, Mirror Replication Agent for Microsoft SQL Server queries the primary database to obtian the transaction ID of the newest transaction that can be truncated. Mirror Replication Agent for Microsoft SQL Server then marks as reusable the transaction log space before the newest transaction. Microsoft SQL Server can then write log records into the reusable space.

The sp_repltrans and sp_repldone Microsoft SQL Server commands are issued by Mirror Replication Agent to control log truncation within Microsoft SQL Server. These commands require that the Mirror Replication Agent user have the db_owner role permission.

Note Microsoft SQL Server allows only one session to control log truncation using the sp_repltrans and sp_repldone commands. You should not use these commands while Mirror Replication Agent is controlling the log truncation processing.

Using Windows authentication with Microsoft SQL Server

When running Mirror Replication Agent for Microsoft SQL Server on a Windows platform, you have the option of configuring it to connect to Microsoft SQL Server using Windows credentials to authenticate the user.

❖ **To use Windows authentication**

1 In your primary Microsoft SQL Server, add the user who will be starting Mirror Replication Agent, <*rauser*>, as a Windows-authenticated user, including the user domain as appropriate. Be sure to add the <*ra_user*> to the primary database and grant the appropriate permissions. For additional information, refer to the Microsoft SQL Server documentation.

2 On the machine on which the Mirror Replication Agent for Microsoft SQL Server is running, add <*domain*>\<*ra_user*> to the Windows user account. If no domain exists, add only the <*ra_user*> to the Windows user account.

3 On the same machine, copy the *sqljdbc_auth.dll* file from the Microsoft SQL Server JDBC driver location to a directory on the Windows system path. When you installed the Microsoft SQL Server JDBC driver, the *sqljdbc_auth.dll* files were installed in the following location:

```
<install_dir>\sqljdbc_<version>\<language>\auth\
```

Note On a 32-bit processor, use the *sqljdbc_auth.dll* file in the x86 folder. On a 64-bit processor, use the *sqljdbc_auth.dll* file in the x64 folder.

4 On the same machine, login as the <*ra_user*> and start the Mirror Replication Agent for Microsoft SQL Server instance.

5 Log in to Mirror Replication Agent and configure the following parameters using values appropriate for the primary Microsoft SQL Server:

```
ra_config pds_server_name, <server>
ra_config pds_port_number, <port>
ra_config pds_database_name, <database>
ra_config pds_username, <ra_user>
ra_config pds_integrated_security, true
```

6 Continue configuring and using Mirror Replication Agent as described in Mirror Replication Agent documentation.

Mirror Replication Agent for Microsoft SQL Server setup test scripts

Mirror Replication Agent provides a set of test scripts that automate the process of creating a replication test environment that you can use to verify the installation and configuration of the Mirror Replication Agent software and the basic function of the other components in your replication system.

The Mirror Replication Agent test scripts are located in the *scripts* subdirectory under the Mirror Replication Agent base directory, for example, *MA-15_1\scripts*.

The Mirror Replication Agent test scripts perform the following tasks:

1. Create a primary table.
2. Create a replicate table that corresponds to the primary table.
3. Create the primary data server connection in the primary Replication Server.
4. Create the replication definition in the primary Replication Server.
5. Test the replication definition.
6. Create the subscription in the standby Replication Server.
7. Test the subscription.
8. Initialize Mirror Replication Agent and create Mirror Replication Agent objects.
9. Modify the primary table.
10. Clean up and remove the replication test environment created by the other scripts.

Before you begin

Before running the test scripts, make sure that you have:

- Installed a Microsoft SQL Server data server
- Installed a data server to act as a standby data server
- Created the standby database in the standby data server

- Installed primary and standby Replication Servers (PRS and SRS)

Note The PRS and SRS can be the same Replication Server. However, if they are not, you must create routes between them.

- Added the standby database as an SRS-managed database
- Added an entry for Mirror Replication Agent to the *interfaces* or *sql.ini* file (if you intend to use the isql utility to administer Mirror Replication Agent).
- Used the sybfilter driver to make the Microsoft SQL Server transaction log files readable by Mirror Replication Agent
- Installed the Mirror Replication Agent software, created a Mirror Replication Agent instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, lower
  - ra_config rs_source_ds, mrx
  - ra_config rs_source_db, test

Note These recommended configuration parameter values are for this test only. The values you should use in a production environment (or even a different test environment) may be different.

- Configured all the pds, rs, and rssd connection parameters and tested them successfully with the Mirror Replication Agent test_connection command
- Confirmed that all servers are running
- Confirmed that the Mirror Replication Agent instance is in the *Admin* state

Create the primary table

Use your Microsoft SQL Server database access tool (such as Microsoft sqlcmd) to log in to the primary Microsoft SQL Server database and execute the *mssql_create_test_primary_table.sql* script in the *%SYBASE%\MA-15_1\scripts\mssql* directory to create the primary table in the primary database.

Note This script (*mssql_create_test_primary_table.sql*) does not specify a database name. You must either edit the script to include a use command or invoke isql with the -d option.

Create the replicate table

Use isql or another query processor to log in to the standby data server and execute the *mssql_create_test_replicate_table.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory to create the replicate table in the standby database.

Note If you have enabled DDL replication, you do not need to create the replicate table.

Create the Replication Server connection for Mirror Replication Agent

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_primary_connection.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory to create the Replication Server connection to the primary database.

Create the replication definition

Edit the *rs_create_test_db_repdef.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory so that all occurrences of *{pds}.{pdb}* contain the name of the Replication Server connection used by your Mirror Replication Agent. Then, use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_db_repdef.sql* script to create a test replication definition.

Test the replication definition

Use isql or another query processor to log in to the RSSD of the primary Replication Server and execute the *rssd_helprep_test_repdef.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory to test the replication definition.

Create the subscription

The *rs_create_test_db_sub.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory is designed for use with Replication Server 11.5 or later and is provided for this step.

❖ **To create the test subscription**

Note This procedure assumes that no materialization is necessary. See the Mirror Activator *Administration Guide* for more information about database materialization.

---

1 Edit the subscription script file (*rs_create_test_db_sub.sql*) so that the values for *PDS.PDB* in the with primary at clause match the *PDS.PDB* values that you used to create the connection for the primary data server and primary database. These values are initially *{pds}.{pdb}*. Also, change the values for *SDS.SDB* on the with replicate at clause for each command match the *SDS.SDB* values that you used to create the connection to the standby data server and standby database. These values are initially set to *{sds}.{sdb}*.

2 Use isql or another query processor to access the standby Replication Server and execute the appropriate script to create the test subscription.

Test the subscription

Use isql or another query processor to access the RSSD of the standby Replication Server and execute the *rssd_helpsub_test_sub.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory to test the subscription.

Initialize Mirror Replication Agent

If you have not already done so, initialize Mirror Replication Agent and create Mirror Replication Agent objects in the primary database. If this is first-time initialization, be sure to follow the procedure described in “First-time initialization” on page 7.

❖ **To initialize Mirror Replication Agent**

1. Use isql or another query processor to log in to the Mirror Replication Agent administration port.
2. Use the following commands to initialize Mirror Replication Agent and create Mirror Replication Agent objects in the primary database:

```
pdb_init
ra_init
```

For general information, see the Mirror Activator *Administration Guide*.

Mark a primary table for replication

Mark the test primary table (mrx_test) that was created in the Microsoft SQL Server database by the *mssql_create_test_primary_table.sql* script in the *%SYBASE%\MA-15_1\scripts\mssql* directory.

❖ **To mark the test primary table for replication**

1. Use isql or another query processor to log in to the Mirror Replication Agent administration port.
2. Use the following command to mark the mrx_test table for replication:

```
pdb_setreptable mrx_test, mark
```

Note Make sure that replication is enabled for the mrx_test table.

See the Mirror Activator *Administration Guide* for more information about marking objects in the primary database.

Start replication

If you have not done so, put the Mirror Replication Agent instance in the *Replicating* state now.

❖ **To start test replication**

1. Use isql or another query processor to log in to the Mirror Replication Agent administration port.
2. Use the following command to put the Mirror Replication Agent instance in the *Replicating* state:

```
resume
```

See the Mirror Activator *Administration Guide* for more information about starting replication.

Execute the test transaction script in Microsoft SQL Server

Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the Microsoft SQL Server data server and execute the *mssql_primary_test_transactions.sql* script in the *%SYBASE%\MA-15_1\scripts\mssql* directory to generate transactions in the primary table in the primary database.

Note This script (*mssql_primary_test_transactions.sql*) does not specify a database name. You must either edit the script to include a use command or invoke isql with the -d option.

Check results of replication

Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the Microsoft SQL Server data server and execute the *mssql_select_test_primary.sql* script in the *%SYBASE%\MA-15_1\scripts\mssql* directory to view the changes in the test primary table in the primary database.

Note This script (*mssql_select_test_primary.sql*) does not specify a database name. You must either edit the script to include a use command or invoke isql with the -d option.

Use isql or another query processor to log in to the standby database and execute the *mssql_select_test_replicate.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase* directory to verify that the transactions generated in the primary database were replicated to the test replicate table in the standby database.

Clean up the test environment

Use the following procedure to clean up and remove the replication test environment that you created in the previous sections.

❖ **To clean up and remove the replication test environment**

1. Log in to the Mirror Replication Agent administration port using isql (or another query processor).
2. Use the following command to quiesce the Mirror Replication Agent instance:

   ```
   quiesce
   ```

3. Use the following command to remove the Mirror Replication Agent transaction log and unmark the test primary table (mrx_test) in the Microsoft SQL Server database:

   ```
   ra_deinit, force
   ```

4. Log in to the standby Replication Server and execute the following scripts:
   - *%SYBASE%\MA-15_1\scripts\sybase\rs_drop_test_db_sub.sql* (to drop the test subscription)

Edit the *rs_drop_test_db_sub.sql* script file so that the values for *PDS.PDB* in the with primary at clause match the *PDS.PDB* values that you used to create the connection for the primary data server and primary database. These values are initially *{pds}.{pdb}*. Also, change the values for *SDS.SDB* on the with replicate at clause to match the *SDS.SDB* values that you used to create the connection to the standby data server and standby database. These values are initially set to *{sds}.{sdb}*.

- *%SYBASE%\MA-15_1\scripts\sybase\rs_drop_test_db_repdef.sql* (to drop the test replication definition)
- *%SYBASE%\MA-15_1\scripts\sybase\rs_drop_test_primary_connection.sql* (to drop the test connection to the primary database)

5 Log in to the standby data server and execute the *mssql_drop_test_replicate_table.sql* script in the *%SYBASE%\MA-15_1\scripts\sybase\* directory to drop the test replicate table.

6 Use your Microsoft SQL Server database access tool (such as Microsoft sqlcmd) to log in to the Microsoft SQL Server data server and execute the *mssql_drop_test_primary_table.sql* script in the *%SYBASE%\MA-15_1\scripts\mssql\* directory to drop the test primary table.

Note The *mssql_drop_test_primary_table.sql* script does not specify a database name. You must either edit the script to include a use command or invoke sqlcmd with the -d option.

CHAPTER 2

Mirror Replication Agent for Oracle

The term “Mirror Replication Agent for Oracle” refers to an instance of Mirror Replication Agent 15.1 software installed and configured for a primary database that resides in an Oracle data server.

Note For information on the basic functionality of Mirror Replication Agent 15.1, refer to the Mirror Activator *Administration Guide* and the Mirror Replication Agent *Reference Manual*.

Oracle-specific considerations

The following topics are included in this section:

- Mirror Replication Agent connectivity
- Mirror Replication Agent permissions
- Redo and archive log setup
- Supplemental logging
- Recycle bin setup
- Setting ddl_username and ddl_password
- Character case of database object names
- Format of origin queue ID
- Replication Server and RSSD scripts

- Datatype compatibility
- Oracle datatype restrictions
- Oracle large object (LOB) support
- Oracle user-defined types
- Marking and unmarking sequences
- Enabling and disabling replication for sequences
- Real Application Clusters (RAC)
- Automatic Storage Management

Mirror Replication Agent connectivity

Connectivity between 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 this driver is installed in 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. See the *Oracle Database Net Services Administrator's Guide* for more information.

Mirror Replication Agent permissions

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

- create session – required to connect to Oracle.
- select_catalog_role – required to select from the DBA_* views.
- alter system – required to perform redo log archive operations.
- execute on DBMS_FLASHBACK – required to execute DBMS_FLASHBACK.get_system_change_number.
- alter any procedure – required to manage 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 support replication.
- 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 table replication.
- select on SYS.CON$ – required to support table replication.
- select on SYS.CDEF$ – required to support replication.
- select on SYS.IND$ – required to support index identification.
- select on SYS.USER$ – required to support replication.
- select on SYS.SEQ$ – required to support sequence replication.

In addition, the user who starts Mirror Replication Agent for Oracle instance must have read access to mirror copies of the Oracle redo log files and the archive directory that contains the archive log files to be accessed for replication. If Mirror Replication Agent is configured to remove old archive files, the user must have update authority to the directory and the archive log files. If the redo logs or archived redo logs are stored within Automatic Storage Management (ASM), the user who starts Mirror Replication Agent for Oracle must have read access to the ASM disk devices that contain the redo log data.

Redo and archive log setup

Note The Mirror Replication Agent for Oracle *must* be installed on a machine where it can directly access mirrored copies of 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, Mirror Replication Agent can be configured to do so, but it requires that you turn auto-archiving off and requires 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, Mirror Replication Agent must be configured to use a directory path where archive log files are located. By default, an Oracle instance creates multiple directories under the flash recovery area specified by the DB_RECOVERY_FILE_DEST parameter of the Oracle ALTER SYSTEM command, each directory corresponding to and named after a separate day. However, Mirror Replication Agent requires mirrored archived redo log files to reside in a single directory. Consequently, you must configure Oracle to archive to a single directory to be read by Mirror Replication Agent.

Note To prevent conflicts with other archive file processes, you may want to configure Oracle to duplex the archive log files into an additional destination directory that is used only for replication.

For information on specifying archive log destinations for your Oracle environment, see the Oracle ALTER SYSTEM command and LOG_ARCHIVE_DEST_n parameter.

Note This section discusses access to Oracle archived redo logs that are stored as file-system files. If the archived redo logs are stored using Oracle ASM, see "Automatic Storage Management" on page 65 for details.

Mirror Replication Agent for Oracle requires the following settings in your Oracle database:

- Redo log archiving must be enabled:

  ```
  alter database ARCHIVELOG;
  ```

  Note If you are using Oracle Real Application Clusters (RAC), you must enable redo log archiving for each instance in the cluster.

  Verify that log archiving is enabled:

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

  If you are using Oracle RAC, use the following SQL to verify that log archiving has been enabled:

```
select instance, name, log_mode from gv$database;
```

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

Accessing archive log files from a remote site

Because 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 the Oracle ALTER SYSTEM command and LOG_ARCHIVE_DEST_n parameter 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) Mirror Replication Agent does not archive, 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 Mirror Replication Agent as the data in the mirrored redo log files is replicated.

Mirror Replication Agent for Oracle requires the following settings in your Oracle database depending on the Oracle version.

For Oracle 10g

❖ **To disable automatic archiving**

1 Make sure you have SYSDBA administrator privileges, and close the database.

2 Enter the following:

```
alter database ARCHIVELOG MANUAL;
```

3. 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 restart the Oracle server. For more information about the log_archive_start parameter or the alter system command, see the Oracle *Database Reference Manual*.

4. Automatic archiving must be disabled in the active server and when you restart the Oracle server. To stop automatic archiving in the active server:

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

5. To disable automatic archiving when you restart 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 Mirror Replication Agent to be overwritten, then the data in the lost redo log file will not be replicated. You can recover from this situation by reconfiguring 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 Mirror Replication Agent, and reset pdb_include_archives to false.

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, execute the following Oracle commands:

```
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.

Recycle bin setup

The Oracle flashback feature added in Oracle 10g is not supported in Mirror Replication Agent for Oracle. Mirror Replication Agent requires that you disable the recycle bin. To disable the recycle bin (which requires sysdba privileges):

```
purge dba_recyclebin;
```

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

Note If you are using Oracle RAC, you must disable the recycle bin for each instance in the cluster.

To view the contents of the recycle bin:

```
select * from dba_recyclebin;
```

To view the current recycle bin configuration:

```
select inst_id, value from gv$parameter
```

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 or target database. This user must have permission to execute all replicated DDL commands at the target database. The ddl_password parameter is the password corresponding to the database user name. In addition, the ddl_username database user must have permission to issue the ALTER SESSION SET CURRENT_SCHEMA command for any primary database user that might issue a DDL command to be replicated.

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

Special usage notes

The value of the ddl_username parameter must not be the same as the value of the maintenance user defined in Replication Server for the standby connection. If these names are the same, a Replication Server error results.

The value of the ddl_username parameter is sent in the LTL for all replicated DDL statements. When DDL is replicated, Replication Server connects to the standby database using the user ID and password specified by the ddl_username and ddl_password parameters. Replication Server then issues the following command:

```
ALTER SESSION SET CURRENT_SCHEMA=user
```

Here, *user* is the user ID that generated the DDL operation at the primary database. The actual DDL command is then executed against the standby database. If the user ID specified in ddl_username does not have permission to issue the ALTER SESSION SET CURRENT_SCHEMA or to execute the DDL command against the user schema, the command fails.

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. You can modify this list using the pdb_ownerfilter command. In addition, Sybase has provided a default list of owners whose objects will not be replicated. However, you cannot remove the SYS owner. You can use the pdb_ownerfilter command to return, add, or remove the list of owners whose objects will not be replicated. See the Mirror Replication Agent *Reference Manual* for more information.

Note The truncate table command is replicated as rs_truncate.

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 Mirror Replication Agent.

To control the way Mirror Replication Agent 15.1 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 rendered in LTL as `TABA`
- `create table Tabb` is rendered in LTL as `TABB`
- `create table 'TaBc'` is rendered in LTL 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 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
4-15	6	System change number
16-19	2	Redo log thread
20-23	2	System change number subindex
24-43	10	Redo log record block address
44-55	6	System change number of the oldest active transaction *begin*
56-63	4	Locator ID

Replication Server and RSSD scripts

Replication Server requires changes to the RSSD to support Oracle datatypes. To implement these changes, execute the following scripts against the RSSD database:

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

$SYBASE/REP-15_1/scripts/hds_oracle_udds.sql
$SYBASE/REP-15_1/scripts/hds_oracle_funcstrings.sql
$SYBASE/REP-15_1/scripts/hds_clt_ase_to_oracle.sql
$SYBASE/REP-15_1/scripts/hds_clt_oracle_to_ase.sql

The Mirror Replication Agent installation has supplemental script changes to support additional Replication Server user-defined datatypes for new Oracle datatypes and replication of DDL commands.

The following revised Replication Server scripts are shipped with Mirror Replication Agent 15.1 and must be applied in addition to the aforementioned scripts when the installed Replication Server is version 15.0.1 or earlier:

$SYBASE/MA-15_1/scripts/oracle/hds_oracle_new_udds.sql
$SYBASE/MA-15_1/scripts/oracle/
hds_oracle_new_setup_for_replicate.sql
$SYBASE/MA-15_1/scripts/oracle/oracle_create_error_class_1_rs.sql
$SYBASE/MA-15_1/scripts/oracle/
oracle_create_error_class_2_rssd.sql
$SYBASE/MA-15_1/scripts/oracle/oracle_create_error_class_3_rs.sql

❖ **To apply the script changes for user-defined datatypes**

1. Execute the following existing Replication Server script against your Replication Server System Database (RSSD) database:

   $SYBASE/REP-15_1/scripts/hds_oracle_udds.sql

2. If your Replication Server is version 15.0.1 or earlier, also apply the script from the Mirror Replication Agent 15.1 installation to your RSSD:

   $SYBASE/MA-15_1/scripts/oracle/hds_oracle_new_udds.sql

3. If your Replication Server is version 15.0.1 or earlier, apply the following script to support replication of DDL to an Oracle standby database:

   $SYBASE/MA-15_1/scripts/oracle/hds_oracle_new_setup_replicate.sql

   This script is used to define Replication Server objects that must be created in the standby database. Use this script *instead of* the *hds_oracle_setup_replicate.sql* script provided in the Replication Server installation directory. This revised script contains additional changes to support Oracle-to-Oracle DDL replication.

4. At your RSSD database, apply the script that defines the custom function strings used by Replication Server to communicate with an Oracle standby database:

   $SYBASE/REP-15_1/scripts/hds_oracle_funcstrings.sql

5. At your RSSD database, apply the scripts that define the class-level translations for the Oracle datatypes:

   $SYBASE/REP-15_1/scripts/hds_clt_ase_to_oracle.sql

   $SYBASE/REP-15_1/scripts/hds_clt_oracle_to_ase.sql

6. To correctly define the Oracle error class for Replication Server 15.0.1 or an earlier version, use three scripts:

   - Apply the following script at Replication Server:

     $SYBASE/MA-15_1/scripts/oracle/oracle_create_error_class_1_rs.sql

   - Apply the following against your RSSD:

     $SYBASE/MA-15_1/scripts/oracle/oracle_create_error_class_2_rssd.sql

   - Apply the following script at Replication Server:

     $SYBASE/MA-15_1/scripts/oracle/oracle_create_error_class_3_rs.sql

For more information, see Chapter 4, "Database Server Issues," in the Replication Server *Heterogeneous Replication Guide*.

Datatype compatibility

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

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

Table 2-2: Recommended Oracle 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	rs_oracle_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	

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
DATE	8 bytes, fixed-length	datetime or rs_oracle_datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 9999 AD. If pdb_convert_datetime is true, the Sybase datatype used should be datetime. The value replicated is YYYYMMDD HH:MM:SS.sss. If pdb_convert_datetime is false, the Sybase datatype used should be rs_oracle_datetime. The format replicated is MM/DD/YYYY HH:MI:SS.
TIMESTAMP(n)	21-31 bytes, variable-length	datetime or rs_oracle_timestamp9	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. If pdb_convert_datetime is true, the Sybase datatype used should be datetime. If pdb_convert_datetime is false, the Sybase datatype used should be rs_oracle_timestamp9.
TIMESTAMP(n) WITH [LOCAL] TIME ZONE	Variable-length	rs_oracle_timestamptz		
INTERVAL YEAR(n) TO MONTH	Variable-length	rs_oracle_interval		
INTERVAL DAY(n) TO SECOND(n)	Variable-length	rs_oracle_interval		

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
LONG	2GB, variable-length character data	text		
LONG RAW	2GB, variable-length binary data	image		
BLOB	4GB, variable-length binary large object	image	2GB	
CLOB	4GB, variable-length character large object	text	2GB	
NCHAR	255 bytes, multi-byte characters	unichar or char	32K	
NCLOB	4GB, variable-length multibyte character large object	unitext or text	2GB	For Replication Server 15.0 and later versions, the NCLOB datatype maps to unitext. For earlier versions of Replication Server, the NCLOB datatype maps to image.
NVARCHAR2	2000 bytes, variable-length, multibyte character data	univarchar or varchar	32K	
MLSLABEL	5 bytes, variable-length binary OS label			Not supported.

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
NUMBER (p,s)	21 bytes, variable-length numeric data	float, int, real, number, decimal, or rs_oracle_decimal	float is 4 or 8 bytes. int is 4 bytes. real is 4 bytes. number and decimal are 2 to 17 bytes.	The float datatype can convert to scientific notation if the range is exceeded. Integers (int) are truncated if they exceed the Replication Server range of 2,147,483,647 to -2,147,483,648 or 1x10^{-130} to 9.99x10^{25}. The number and decimal datatypes are truncated if they exceed the range of -10^{38} to 10^{38}-1. Oracle precision ranges from 1 to 38 digits. Default precision is 18 digits. Oracle scale ranges from -84 to 127. Default scale is 0.
RAW	2000 bytes, variable-length binary data	rs_oracle_binary	32K	
ROWID	6 bytes, binary data representing row addresses	rs_oracle_rowid	32K	
UDD object type	Variable length character data	rs_rs_char_raw	32K	See “Oracle user-defined types” on page 54.
VARCHAR2	2000 bytes, variable-length character data	varchar	32K	

Replication Server 15.0 unsigned datatype mapping

For Replication Server 15.0, unsigned datatypes are supported and can be specified in the replication definitions.

For versions of Replication Server earlier than 15.0, these datatypes cannot be specified, and Table 2-3 identifies the replication definition datatypes that should be used.

Table 2-3: Unsigned integer replication definition datatype mapping

Replication Server 15.0 unsigned datatypes	**Replication definition datatypes**
unsigned bigint	NUMERIC (20)
unsigned int	NUMERIC (10)
unsigned smallint	INT
unsigned tinyint	TINYINT

Oracle datatype restrictions

Note See the Mirror Activator *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 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 12.5 or later, these datatype size restrictions are no longer in effect.

The following Oracle datatypes are not supported:

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

See also

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

Oracle *SQL Reference Manual* for a complete list of Oracle-supplied types.

Oracle large object (LOB) support

Oracle LOB data can exist in several formats in Oracle:

- Character datatypes:
  - LONG
  - CLOB
  - NCLOB
- Binary datatypes:
  - 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, Mirror Replication Agent does not support BFILE data.

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 only updates a LOB that has been defined with the disable storage in row clause records only the LOB data's change to its index, and not the table that holds the LOB. So the 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
```

Note For purposes of replication, LOB objects populated with the empty_clob or empty_lob function are replicated as a NULL values. Replication definitions for LOB columns should therefore include the "null" keyword as part of the column definition.

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 15.1 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 user-defined datatypes in Oracle, the datatype specified in the replication definition must be rs_rs_char_raw. This datatype is installed by the *hds_oracle_new_udds.sql* script as described in “Replication Server and RSSD scripts” on page 44.

❖ 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 the rs_rs_char_raw 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 ma_source_db.ma_source_ds
with all tables named 'USE_NAME_T'
(
   PKEY int,
   PNAME rs_rs_char_raw
)
primary key (PKEY)
searchable columns (PKEY)
go
```

Note The ltl_character_case must be upper for this example.

Marking and unmarking sequences

Support for Oracle sequence replication is supported for replication to Oracle only. No support is provided for replicating a sequence value to a non-Oracle standby database.

Mirror Replication Agent supports replication of sequences in the primary database. To replicate a sequence invoked in a primary database, the sequence must be marked for replication, and replication must be enabled for that sequence. (This is analogous to marking and enabling replication for tables.)

Note Marking a sequence for replication is separate from enabling replication for the sequence. If the value of the pdb_dflt_object_repl parameter is true, replication is enabled automatically at the time a sequence is marked. For more information, see "Enabling and disabling replication for sequences" on page 61.

Oracle does not log information every time a sequence is incremented. Sequence replication occurs when Mirror Replication Agent captures the system table updates that occur when the sequence cache is refreshed. Therefore, the sequence value replicated when a sequence is marked for replication is the "next" sequence value to be used when the current cache expires. The result is that not every individual increment of a sequence is replicated, but the standby site will always have a value greater than the primary site's currently available cached values.

If you need to temporarily suspend replication of a marked sequence, you can disable replication for the sequence.

Replication Server changes to support sequence replication

By default, Replication Server is not installed with support for replication of Oracle sequence objects. Changes are required to Replication Server and the standby Oracle database before replication of Oracle sequences is possible.

For Replication Server, you must create a replication definition that defines a stored procedure to assist with sequence replication. To do this, execute the *$SYBASE/MA-15_1/scripts/oracle/oracle_create_rs_sequence_repdef.sql* script against your primary Replication Server after editing the script to replace values *{pds}* and *{pdb}* with the name of your primary Replication Server connection. These values can also be found in the rs_source_ds and rs_source_db Mirror Replication Agent configuration properties.

Note The replication definition assumes that a database replication definition exists. You may need to alter the definition if a database replication definition does not exist. For details, see comments in the *oracle_create_rs_sequence_repdef.sql* script.

In the standby Oracle database, you must create a stored procedure to support sequence replication. To do this, log in to the standby Oracle database as the maintenance user defined in your Replication Server connection to the standby database. Then, execute the *$SYBASE/MA-15_1/scripts/oracle/oracle_create_replicate_sequence_proc.sql* script to create the necessary stored procedure.

Note The maintenance user defined in your Replication Server connection to the standby database must have sufficient privileges to execute functions in the Oracle DBMS_SQL package. Also, this maintenance user must have authority at the standby Oracle database to update any sequence that is replicated.

Marking a sequence for replication

❖ **To mark a sequence for replication**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the pdb_setrepseq command to determine if the sequence is already marked in the primary database:

   ```
   pdb_setrepseq pdb_seq
   ```

   Here, *pdb_seq* is the name of the sequence in the primary database that you want to mark for replication.

   Consider the following:

   - If the pdb_setrepseq command returns information that the specified sequence is marked, you do not need to continue this procedure.
   - If the pdb_setrepseq command returns information that the specified sequence is not marked, continue this procedure to mark the sequence for replication.
3. Use the pdb_setrepseq command to mark the sequence for replication.

   The pdb_setrepseq command allows you to mark the primary sequence to be replicated and specify a different sequence name to use in the standby database.

   - Use the following command to mark the sequence for replication when the sequence name you wish to increment at the standby site has the same name:

     ```
     pdb_setrepseq pdb_seq, mark
     ```

Here, *pdb_seq* is the name of the sequence in the primary database that you want to mark for replication.

Note Replicating a sequence with a different name that is provided is consistent with other marking commands but is not a typical configuration.

- Use the following command to mark the sequence for replication using a different sequence name:

  ```
  pdb_setrepseq pdb_seq, rep_seq, mark
  ```

  Here, *pdb_seq* is the name of the sequence in the primary database that you want to mark for replication, and *rep_seq* is the name of the sequence in the standby database that you want to increment.

Note Replicating sequence values to a sequence with a different name at the standby site assumes that the standby site sequence has the same attributes and starting value as the primary site sequence.

Consider the following:

- If the value of the pdb_dflt_object_repl parameter is true, the sequence marked for replication with the pdb_setrepseq command is ready for replication after you invoke the pdb_setrepseq command successfully.
- If the value of the pdb_dflt_object_repl parameter is true (the default value), you can skip step 4 in this procedure.
- If the value of the pdb_dflt_object_repl parameter is false, you must enable replication for the sequence before replication can take place.

4 Use the pdb_setrepseq command to enable replication for the marked sequence:

```
pdb_setrepseq pdb_seq, enable
```

Here, *pdb_seq* is the name of the marked sequence for which you want to enable replication.

After replication is enabled for the sequence, you can begin replicating invocations of that sequence in the primary database.

Note To replicate a sequence, you must also run the *oracle_create_replicate_sequence_proc.sql* script in the *$SYBASE/MA-15_1/scripts/oracle* directory at the standby site to create a procedure named rs_update_sequence.

Unmarking a sequence

❖ **To unmark a sequence**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the pdb_setrepseq command to confirm that the sequence is marked in the primary database:

   ```
   pdb_setrepseq pdb_seq
   ```

   Here, *pdb_seq* is the name of the sequence that you want to unmark.

   Consider the following:

   - If the pdb_setrepseq command returns information that the specified sequence is marked, continue this procedure to unmark the sequence.
   - If the pdb_setrepseq command does not return information that the specified sequence is marked, you do not need to continue this procedure.
3. Use the pdb_setrepseq command to disable replication of the sequence:

   ```
   pdb_setrepseq pdb_seq, disable
   ```

   Here, *pdb_seq* is the name of the sequence that you want to unmark.
4. Use the pdb_setrepseq command to remove the replication marking from the sequence:

   ```
   pdb_setrepseq pdb_seq, unmark
   ```

   Here, *pdb_seq* is the name of the sequence that you want to unmark.

   If you need to force the unmark, you can use the following command:

   ```
   pdb_setrepseq pdb_seq, unmark, force
   ```

5. Use the pdb_setrepseq command to confirm that the sequence is no longer marked for replication:

   ```
   pdb_setrepseq pdb_seq
   ```

   Here, *pdb_seq* is the name of the sequence in the primary database that you unmarked.

Enabling and disabling replication for sequences

If you need to temporarily suspend replication of a sequence, you can use the pdb_setrepseq command to disable replication for the marked sequence. When you are ready to resume replication of the marked sequence, you can use the pdb_setrepseq command to enable replication.

Note No sequences are marked by default for replication.

To replicate updates of a sequence in the primary database, the sequence must be marked for replication and replication must be enabled for that sequence.

Marking a sequence for replication is separate from enabling replication for the sequence. For more information, see “Marking a sequence for replication” on page 58.

Enabling replication for sequences

❖ **To enable replication for a marked sequence**

1. Log in to the Mirror Replication Agent instance with the administrator login.
2. Use the pdb_setrepseq command to verify that replication is disabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   Here, *pdb_seq* is the name of the marked sequence you want to enable replication for.

If the pdb_setrepseq command returns information that the sequence is marked and has replication disabled, continue this procedure to enable replication for the sequence.

Note A sequence must be marked for replication before replication can be enabled or disabled for the sequence.

---

3. Use the pdb_setrepseq command to enable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, enable
   ```

   Here, *pdb_seq* is the name of the marked sequence for which you want to enable replication.

   After replication is enabled for the sequence, any invocation of that sequence will be replicated.

4. You can use the pdb_setrepseq command again to verify that replication is now enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   Here, *pdb_seq* is the name of the marked sequence for which you want to verify that replication is enabled.

Disabling replication for marked sequence

❖ **To disable replication for a marked sequence**

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

2. Use the pdb_setrepseq command to verify that replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   Here, *pdb_seq* is the name of the marked sequence you want to disable replication for.

   If the pdb_setrepseq command returns information that the sequence is marked and has replication enabled, continue this procedure to disable replication for the sequence.

Note A sequence must be marked for replication before replication can be enabled or disabled for that sequence.

---

3 Use the pdb_setrepseq command to disable replication for the sequence:

```
pdb_setrepseq pdb_seq, disable
```

Here, *pdb_seq* is the name of the marked sequence for which you want to disable replication.

After replication is disabled for the sequence, any invocation of that sequence will not be captured for replication until replication is enabled again.

4 You can use the pdb_setrepseq command again to verify that replication is now disabled for the sequence:

```
pdb_setrepseq pdb_seq
```

Here, *pdb_seq* is the name of the marked sequence for which you want to verify that replication is disabled.

Real Application Clusters (RAC)

Mirror Replication Agent for Oracle 15.1 provides support for Oracle 10g RAC environments. When a Mirror Replication Agent for Oracle instance is initialized, the Oracle database is queried to determine how many nodes are supported by the cluster. Based on this information, Mirror Replication Agent automatically configures itself to process the mirrored redo log information from all nodes.

To process the mirrored redo log data from all nodes in an Oracle RAC cluster, Mirror Replication Agent must execute from a location that has access to the mirrored data. The Mirror Replication Agent must also have read access to the shared storage where the mirrored online and archived redo logs exist.

Mirror Replication Agent can be configured to connect to a single Oracle instance by supplying the required host, port, and Oracle SID values to the pds_host_name, pds_port_number and pds_database_name configuration parameters. In an Oracle RAC environment, Mirror Replication Agent must be able to connect to any node in the cluster in the event that a node fails or otherwise becomes unavailable. To support the configuration of multiple node locations, Mirror Replication Agent supports connectivity to all possible RAC nodes by obtaining needed information from an Oracle *tnsnames.ora* file for one specified entry. As a result, instead of configuring individual host, port and instance names for all nodes, Mirror Replication Agent only requires the location of a *tnsnames.ora* file and the name of the TNS connection to use.

Sybase recommends that you point Mirror Replication Agent to a *tnsnames.ora* entry that contains the address for all nodes in the cluster.

For example, if the following entry exists in a *tnsnames.ora* file for a three-node cluster, Mirror Replication Agent can be instructed to use that entry by providing the *tnsnames.ora* file location to the pds_tns_filename configuration property and specifying RAC10G as the value for the pds_tns_connection configuration property:

```
RAC10G =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = node1-vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = node2-vip)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = node3-vip)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = rac10g.sybase.com)
    )
   )
```

See the Mirror Replication Agent *Reference Manual* for details about the pds_tns_filename and pds_tns_connection parameters.

Note Mirror Replication Agent must have read access to the *tnsnames.ora* file, which you should copy from the machine on which the primary database is running to the machine on which Mirror Replication Agent is running.

pdb_archive_path

The pdb_archive_path configuration parameter identifies the directory path where Mirror Replication Agent expects to find archived Oracle redo log files. In an Oracle RAC environment, each Oracle instance can be configured to point to one or more archive log destinations. To support replication, all instances in the Oracle RAC cluster must provide a copy of their archive log files to a shared location that Mirror Replication Agent for Oracle can use to access all archived redo logs. The pdb_archive_path configuration parameter must be configured to point to a location to which all Oracle instances write archived log data. Mirror Replication Agent must have read access to this directory and all archived redo logs within that directory.

Note Archived redo logs can also be stored within ASM. See "Automatic Storage Management" on page 65 for details on how the pdb_archive_path configuration should be specified for ASM usage.

Mirror Replication Agent can be configured to remove archived logs from the location specified by pdb_archive_path, using the pdb_archive_remove configuration parameter. This allows Mirror Replication Agent to remove archived log files that are no longer needed to support replication. If pdb_archive_remove is set to true, Mirror Replication Agent must have update authority to the archive log directory and delete authority on the individual archive log files.

Oracle instance failover

If the Oracle instance to which Mirror Replication Agent is connected fails for any reason, Mirror Replication Agent attempts to reconnect to any surviving instance, choosing from the list of instances defined in the *tnsnames.ora* file entry Mirror Replication Agent is configured to use. No manual intervention or configuration is required. If none of the instances are available, Mirror Replication Agent reports an error and continues processing as long as mirrored redo log file information is still available.

Automatic Storage Management

Mirror Replication Agent for Oracle supports the use of the ASM feature. ASM provides file system and volume management support for an Oracle database environment. ASM can be used in both Real Application Cluster (RAC) and non-RAC environments.

ASM provides similar benefits as a redundant array of independent disks (RAID) or a logical volume manager (LVM). Similar to those technologies, ASM allows the definition of a single disk group from a collection of individual disks. ASM attempts to balance loads across all of the devices defined in the disk group. ASM also provides striping and mirroring capabilities.

Unlike RAID or LVMs, ASM only supports files created and read by the Oracle database. ASM cannot be used for a general-purpose file system and cannot store binaries or flat files. Also, ASM files cannot be directly accessed by the operating system.

ASM striping and mirroring

ASM provides striping by dividing files into equal-sized extents. Fine-grained striping extents are 128KB in size. Coarse-grained striping extents are 1MB in size. Striping spreads each file extent evenly across all disks in the assigned disk group.

ASM also provides automatic mirroring of ASM files and allows the mirroring level to be specified by group. This mirroring occurs at the extent level. If a disk group is mirrored, each extent has one or more mirrored copies, and mirrored copies are always kept on different disks in the disk group.

There are three ASM mirroring options:

- Two-way mirroring – each extent has one mirrored copy in this option.
- Three-way mirroring – each extent has two mirrored copies in this option.
- Unprotected mirroring – ASM provides no mirroring in this option, which is used when mirroring is provided by the disk subsystem.

ASM features supported by Mirror Replication Agent for Oracle

The following ASM features are supported by Mirror Replication Agent for Oracle:

- Online redo log files managed by ASM can be used for replication.
- Archive log files managed by ASM can be used for replication.
- ASM disk groups can be changed without interfering with replication. When disks are added or dropped from an ASM disk group, Mirror Replication Agent for Oracle recognizes the change and automatically updates its device information for affected log devices.

- Mirror Replication Agent for Oracle tolerates multiple disk failures within the same disk group without affecting replication.

Archive log removal and configuration

Archive logs that are managed by ASM can be removed from ASM when they are no longer needed by Mirror Replication Agent for Oracle. When the pdb_archive_remove configuration parameter is set to true and the archive logs are managed by ASM, the pdb_archive_path configuration parameter must be set to the name of the ASM disk group in which the archive logs are stored. The disk group name must be preceded with a plus sign "+" indicating that the path is an ASM path. For example:

```
pdb_archive_remove=true

pdb_archive_path=+DISK_GROUP1
```

In addition to automatic removal of archive logs from ASM, manual removal is supported with the pdb_truncate_xlog command. The pdb_archive_path must be set to the ASM disk group name and preceded with a plus "+" sign for archive logs to manually removed.

Disk failure recovery

ASM provides automatic recovery for disk failures. When a disk fails, ASM reconfigures all ASM-managed files in the disk group with the failed disk and removes the failed disk from the disk group. This is known as a rebalance.

When Mirror Replication Agent for Oracle detects a disk failure, it automatically switches to reading disk group mirrors. If the disk group is configured to have the mirror and mirror copy, Mirror Replication Agent for Oracle can recover from multiple disk failures. If the disk group is configured for no mirroring or if too many disks have failed and the log cannot be read, Mirror Replication Agent for Oracle checks to see if an ASM rebalance is occurring or has occurred. Log device information is updated with new ASM information when the rebalance is complete. The time required for a complete rebalance can vary, depending on how many disk are in the failing disk group.

Log device information can be manually updated by issuing the ra_updatedevices command. If a disk group must be changed by adding or removing a disk, ra_updatedevices can be issued to ensure log devices obtain the new disk group configuration. This command must be issued only after the disk group is changed and ASM has completed its rebalance.

If Mirror Replication Agent for Oracle cannot recover on its own from a disk failure or disk group change, ra_updatedevices can be used to update log device information before resuming replication.

Configuration parameters

The following configuration parameters must be set if your log files are being managed by ASM:

asm_password
asm_tns_connection
asm_tns_filename
asm_username

The ASM user ID for asm_username must have sysdba permission and must be set as follows:

```
asm_username="sys as sysdba"
```

For detailed information about these parameters, see the Mirror Replication Agent *Reference Manual*.

Redo log management with ASM

Because Oracle redo log files managed by ASM use multiple disks within a disk group, a simple path cannot be specified for a log device. The following Mirror Replication Agent commands accommodate this ASM functionality.

- pdb_asmdiskmap – This command is available only for Oracle and creates a file with default disk mirror paths for all disks in the ASM disk groups that are used to store online redo logs and archive redo logs. The disks required for online logs are automatically selected by querying ASM.
- ra_helpdevice – This command returns one row output for each disk in the group where the device is stored. For the device ID, the command returns the value of the Oracle log device group ID. The disk mirror path returned is the path obtained from the ASM disk map file.
- ra_updatedevices – When you change a disk group by adding or dropping disks, you must also invoke use this command to be sure the log device repository is updated with correct ASM storage information.

For a full description of these commands and their usage, see the Mirror Replication Agent *Reference Manual*.

Mirror Replication Agent objects in the Oracle primary database

Note This section describes 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. 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_1\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 or Mirror Replication Agent. 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

There are two variables in Mirror Replication Agent database object names shown in this chapter:

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

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

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

You can use the ra_helpsysinfo command to view the names of Mirror Replication Agent transaction log components in the primary database.

See the Mirror Activator *Administration Guide* for details on setting up object names.

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

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

  ```
  ra_helpsysinfo
  ```

  The ra_helpsysinfo command returns a list of all Mirror Replication Agent objects.

Table objects

Table 2-4 lists the tables that are considered Mirror Replication Agent objects.

Table 2-4: Mirror Replication Agent tables

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

Marker objects

Table 2-5 lists Mirror Replication Agent objects related to Replication Server markers. No permissions are granted when these objects are created.

Table 2-5: Mirror Replication Agent marker objects

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 2-6 lists the Oracle sequences that are considered Mirror Replication Agent objects.

Table 2-6: Mirror Replication Agent sequences

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

Marked procedures

Table 2-7 lists 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 2-7: Mirror Replication Agent objects for each marked procedure

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

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, 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, Mirror Replication Agent deletes from the pdb_archive_path location the archive redo logs that have already been processed. The Mirror Replication Agent is not responsible for archiving online transaction logs.

Note Sybase recommends that you configure Mirror Replication Agent to remove archive log files only 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 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

You can truncate the Mirror Replication Agent transaction log manually, at any time, by invoking the pdb_truncate_xlog command at the Mirror Replication Agent administration port.

Mirror Replication Agent for Oracle setup test scripts

Mirror Replication Agent provides a set of test scripts that automate the process of creating a replication test environment that you can use to verify the installation and configuration of the Mirror Replication Agent software and the basic function of the other components in your replication system.

The Mirror Replication Agent test scripts are located in the *scripts* subdirectory under the Mirror Replication Agent base directory, for example, *$SYBASE/MA-15_1/scripts*.

The Mirror Replication Agent test scripts perform the following tasks:

1. Create a primary table.
2. Create a replicate table that corresponds to the primary table.
3. Create the primary data server connection in the primary Replication Server.
4. Create the replication definition in the primary Replication Server.
5. Test the replication definition.
6. Create the subscription in the standby Replication Server.
7. Test the subscription.
8. Create the Mirror Replication Agent transaction log in the primary database.
9. Modify the primary table.
10. Clean up and remove the replication test environment created by the other scripts.

Before you begin

Before running the test scripts, make sure that you have:

- Installed an Oracle data server
- Installed a data server to act as a standby data server
- Created the standby database in the standby data server

- Installed primary and standby Replication Servers (PRS and SRS)

Note The PRS and SRS can be the same Replication Server. You must create routes between them if the PRS and SRS are not the same Replication Server.

- Added the standby database as an SRS-managed database
- Added an entry for Mirror Replication Agent to the *interfaces* or *sql.ini* file (if you intend to use the isql utility to administer Mirror Replication Agent).
- Installed the Mirror Replication Agent software, created a Mirror Replication Agent instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, lower
  - ra_config rs_source_ds, rax
  - ra_config rs_source_db, test

Note These recommended configuration parameter values are for this test only. The values you should use in a production environment (or even a different test environment) may be different.

- Configured all the pds, rs, and rssd connection parameters and tested them successfully with the Mirror Replication Agent test_connection command
- Confirmed that all servers are running
- Confirmed that the Mirror Replication Agent instance is in the *Admin* state

Create the primary table

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_create_test_table.sql* script in the *$SYBASE/MA-15_1/scripts/oracle* directory to create the primary table in the primary database.

Create the replicate table

Use isql or another query processor to log in to the standby data server and execute the *ase_create_test_replicate_table.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory to create the replicate table in the standby database.

Create the Replication Server connection

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_primary_connection.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory to create the Replication Server connection to the primary database.

Create the replication definition

Edit the *rs_create_test_db_repdef.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory so that all occurrences of *{pds}.{pdb}* contain the name of the Replication Server connection used by your Mirror Replication Agent. Then, use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_db_repdef.sql* script to create a test replication definition.

Test the replication definition

Use isql or another query processor to log in to the RSSD of the primary Replication Server and execute the *rssd_helprep_test_repdef.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory to test the replication definition.

Create the subscription

The *rs_create_test_db_sub.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory is designed for use with Replication Server 11.5 or later and is provided for this step.

❖ **To create the test subscription**

Note This procedure assumes that no materialization is necessary. See the Mirror Activator *Administration Guide* for more information about database materialization.

1. Edit the subscription script file (*rs_create_test_db_sub.sql*) so the values for *PDS.PDB* on the with primary at clause match the *PDS.PDB* values that you used to create the connection for the primary data server and primary database. These values are initially *{pds}.{pdb}*. Also change the values for *SDS.SDB* on the with replicate at clause to match the *SDS.SDB* values that you used to create the connection to the standby data server and standby database. These values are initially *{sds}.{sdb}*.
2. Use isql or another query processor to access the standby Replication Server and execute the appropriate script to create the test subscription.

Test the subscription

Use isql or another query processor to access the RSSD of the standby Replication Server and execute the *rssd_helpsub_test_sub.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory to test the subscription.

Initialize Mirror Replication Agent

If you have not already initialized Mirror Replication Agent and created Mirror Replication Agent objects in the primary database, do so now.

❖ **To create the Mirror Replication Agent transaction log**

1. Use isql or another query processor to log in to the Mirror Replication Agent administration port.
2. Use the following commands to initialize Mirror Replication Agent and create Mirror Replication Agent objects in the primary database:

```
pdb_init
ra_init
```

See the Mirror Activator *Administration Guide* for more information.

Mark a primary table for replication

Mark the test primary table (mrx_test) that was created in the Oracle database by the *oracle_create_test_table.sql* script in the *$SYBASE/MA-15_1/scripts/oracle/* directory.

❖ **To mark the test primary table for replication**

1. Use isql or another query processor to log in to the Mirror Replication Agent administration port.
2. Use the following command to mark the mrx_test table for replication:

```
pdb_setreptable mrx_test, mark
```

Note Make sure that replication is enabled for the mrx_test table. See the Mirror Activator *Administration Guide* for more information.

Start replication

If you have not already done so, put the Mirror Replication Agent instance in the *Replicating* state now.

❖ **To start test replication**

1. Use isql or another query processor to log in to the Mirror Replication Agent administration port.
2. Use the following command to put the Mirror Replication Agent instance in the *Replicating* state:

```
resume
```

See the Mirror Activator *Administration Guide* for more information about starting replication.

Execute the test transaction script in Oracle

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_primary_test_transactions.sql* script in the *$SYBASE/MA-15_1/scripts/oracle* directory to generate transactions in the primary table in the primary database.

Check results of replication

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_select_test_primary.sql* script in the *$SYBASE/MA-15_1/scripts/oracle* directory to view the changes in the test primary table in the primary database.

Use isql or another query processor to log in to the standby database and execute the *ase_select_test_replicate.sql* script in the *$SYBASE/MA-15_1/scripts/sybase* directory to verify that the transactions generated in the primary database were replicated to the test replicate table in the standby database.

Clean up the test environment

Use the following procedure to clean up and remove the replication test environment that you created in the previous sections.

❖ To clean up and remove the replication test environment

1. Log in to the Mirror Replication Agent administration port using isql (or another query processor).
2. Use the following command to quiesce the Mirror Replication Agent instance:

   ```
   quiesce
   ```

3. Use the following command to remove the Mirror Replication Agent transaction log and unmark the test primary table (mrx_test) in the Oracle database:

   ```
   ra_deinit, force
   ```

4. Log in to the standby Replication Server and execute the following:

   - *$SYBASE/MA-15_1/scripts/sybase/rs_drop_test_db_sub.sql* (to drop the test subscription).

     Edit the *rs_drop_test_db_sub.sql* script file so that the values for *PDS.PDB* on the with primary at clause match the *PDS.PDB* values that you used to create the connection from the primary data server and primary database. These values are initially *{pds.pdb}*. Also change the values for *SDS.SDB* on the with replicate at clause to match the *SDS.SDB* values that you used to create the connection to the standby data server and standby database. These values are initially *{sds.sdb}*.

- *$SYBASE/MA-15_1/scripts/sybase/rs_drop_test_db_repdef.sql* (to drop the test replication definition)
- *$SYBASE/MA-15_1/scripts/oracle/oracle_drop_rs_primary_connection.sql* (to drop the test connection to the primary database)

5 Log in to the standby data server and execute the *$SYBASE/MA-15_1/scripts/sybase/ase_drop_test_replicate_table.sql* script to drop the test replicate table.

6 Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *$SYBASE/MA-15_1/scripts/oracle/oracle_drop_test_table.sql* script to drop the test primary table.

CHAPTER 3

Mirror Replication Agent for ASE

The term "Mirror Replication Agent for ASE" refers to an instance of Mirror Replication Agent 15.1 software that is configured for a primary database that resides in an Adaptive Server Enterprise server.

Note For information on the basic features and operation of Mirror Replication Agent 15.1, refer to the Mirror Activator *Administration Guide* and the Mirror Replication Agent *Reference Manual*.

ASE database-specific considerations

This section describes general issues and considerations that are specific to using Mirror Replication Agent 15.1 with the Adaptive Server Enterprise 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
- Datatype support

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 3-1 illustrates the format of the origin queue ID for the Mirror Replication Agent for ASE.

Table 3-1: 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 Replication Agent to roll back orphan transactions.
60-63	2	Unused.

ASE Cluster Edition

Mirror Replication Agent for ASE 15.1 provides support for ASE Cluster Edition (CE).

In an ASE CE replication environment, the Mirror Replication Agent must be able to connect to any node in the cluster in the event that a node fails or otherwise becomes unavailable. To support the configuration of multiple node locations, Mirror Replication Agent supports connectivity to all possible ASE CE database instances using the Sybase open-client/open-server *interfaces* (UNIX) or *sql.ini* (Windows) file. Instead of configuring individual host, port, and instance names for all nodes, the Mirror Replication Agent requires only the location of the *interfaces* or *sql.ini* file, the ASE CE server name, and the primary database name. The *interfaces* or *sql.ini* file specified should contain an entry that includes the address for each node in the database cluster.

For example, if the following entry exists in the *interfaces* file for a two-node cluster, Mirror Replication Agent can be instructed to use that entry by providing the *interfaces* file location to the pds_interfaces_file configuration property and specifying “clustA” as the value for the pds_server_name configuration property:

```
clustA
    query tcp ether suse9364asece1.sybase.com 4311
    query tcp ether suse9364asece2.sybase.com 4312
```

See the Mirror Replication Agent *Reference Manual* for details about the pds_interfaces_file and pds_server_name parameters.

Note Mirror Replication Agent must have read access to the *interfaces* or *sql.ini* file, which you should copy from the machine on which the primary database is running to the machine on which Mirror Replication Agent is running.

On initialization, the Mirror Replication Agent automatically determines that the primary database is a cluster database so, besides database connectivity, Mirror Replication Agent does not require any other unique configuration to support ASE CE database replication.

ASE instance failover

If the ASE CE instance to which Mirror Replication Agent is connected fails for any reason, Mirror Replication Agent attempts to reconnect to any surviving instance, choosing from the list of instances defined in the *interfaces* or *sql.ini* file entry Mirror Replication Agent is configured to use. No manual intervention or configuration is required. If none of the instances are available, Mirror Replication Agent reports a warning and continues replication processing.

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 3-2 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 3-2 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 3-2. Instead, you must 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 3-2 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 3-2: 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 Mirror Replication Agent thread in the primary database. **Note** You must preserve the secondary truncation point when you disable the Mirror Replication Agent thread.

Task	Description
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 Mirror Replication Agent thread.

Use the following procedure to find the values of the Mirror 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 Mirror 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 Mirror 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 Mirror Replication Agent thread parameters:

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

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

❖ To set up connection parameters for the Replication Server

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

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

```
    ra_status
```

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

```
    suspend
```

2 Specify the Replication Server host name:

```
ra_config rs_hostname, rs_host
```

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

3 Specify the Replication Server port number:

```
ra_config rs_port_number, NNN
```

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

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

```
ra_config rs_username, 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 Mirror 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 Mirror 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 Mirror 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 Mirror 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 Mirror 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 Mirror Replication Agent thread

1. Log in to the primary database with a System Administrator user role.
2. Stop the Mirror 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 Mirror 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 Mirror 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 3-3 provides a checklist of the snapshot materialization tasks for a standby database in Adaptive Server version 12.5.1 or later. This checklist assumes that the standby Adaptive Server is already installed and configured identically to the primary Adaptive Server.

Table 3-3: 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, 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 3-4 provides a checklist of the snapshot re-materialization tasks for a primary (target) database in Adaptive Server version 12.5.1 or later. This checklist 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 3-4: 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 pre-evaluated 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 for encrypted columns must be provided, even if database-level replication or warm standby is used.

APPENDIX A

Upgrading Mirror Replication Agent

Upgrading Mirror Replication Agent for Oracle

Using any of the upgrade procedures described in this section, the new Mirror Replication Agent for Oracle 15.1 instances will have the same configuration as previously existing instances, including instance names, administrative user IDs and passwords, and administrative port numbers.

Note Mirror Replication Agent for Oracle 15.1 does not support downgrading Mirror Replication Agent for Oracle 15.1 to an earlier log-based version (12.6 or 15.0).

This section includes the following topics:

- Upgrading a log-based Mirror Replication Agent (version 12.6 or 15.0)
- Migrating Mirror Replication Agent 15.1 when upgrading Oracle 9i to 10g

Upgrading a log-based Mirror Replication Agent (version 12.6 or 15.0)

This section describes how to upgrade Mirror Replication Agent for Oracle 12.6 or 15.0 to 15.1.

Note Mirror Replication Agent 15.1 must be installed on the same host on which the primary Oracle server is running.

❖ **To upgrade a log-based Replication version 12.6 or 15.0 to 15.1**

1. For *each* existing Mirror Replication Agent for Oracle instance, Sybase recommends that you first back up the Mirror Replication Agent System Database (RASD), as described in the Mirror Activator 15.1 *Administration Guide*. Then, back up the complete existing Mirror Replication Agent instance directory.
2. Install the Mirror Replication Agent 15.1 software as described in "Installing the Mirror Replication Agent software" in the Mirror Replication Agent 15.1 *Installation Guide*. Sybase recommends that you install Mirror Replication Agent 15.1 into the same *SYBASE* directory as the previous version of Mirror Replication Agent.
3. Download and install the Oracle JDBC driver, and set the CLASSPATH environment variable, as described in the Mirror Replication Agent 15.1 *Installation Guide*. If the CLASSPATH contains another Oracle JDBC driver, be sure to remove it. Only the Oracle JDBC driver required by Mirror Replication Agent 15.1 should be in the CLASSPATH.
4. Create the 15.1 version of all valid existing Mirror Replication Agent instances.

   Note This step creates new Mirror Replication Agent 15.1 instances for all valid existing instances of the previous version of Mirror Replication Agent, regardless of whether the existing instances are for ASE, Oracle, or Microsoft SQL Server. To complete the upgrade for Microsoft SQL Server instances, see the appropriate section in this appendix. If you do not want to run a newly-created instance on this host, simply delete the new instance directory.

   ---

   a. On UNIX, set the SYBASE environment variables by changing directory to the *SYBASE* directory in which Mirror Replication Agent 15.1 is installed and source the *SYBASE* script:

- For C Shell: `source SYBASE.csh`
- For Bourne or Korn shell: `. SYBASE.sh`

b Change directory to the Mirror Replication Agent 15.1 *bin* directory:

- On UNIX:

```
cd $SYBASE/MA-15_1/bin
```

- On Windows:

```
cd %SYBASE%\MA-15_1\bin
```

c Use the ma_admin utility to create new versions of all valid existing instances:

```
ma_admin -u src_directory
```

Here, *src_directory* is the full path name of the previous Mirror Replication Agent version's installation directory. This is the source directory. For example:

```
ma_admin -u d:\sybase\MA-15_0
```

For information about the instances that failed to upgrade, refer to the administration logs (*…/MA-15_1/admin_logs*). After you correct the problem, re-run this command. This command will not upgrade again those Mirror Replication Agent instances that have already been successfully upgraded.

5 If necessary, set the RA_JAVA_DFLT_CHARSET environment variable in the Mirror Replication Agent 15.1 *RUN_instance* script to the name of the Java character set that is equivalent to the one being used at the primary database. For detailed information, see the Mirror Activator 15.1 *Administration Guide*.

6 In the primary Oracle database, grant the previously existing pds_username user any additional required privileges. See "Mirror Replication Agent permissions" on page 6.

7 To prevent any loss of replicated data, deny users (other than the previously existing Mirror Replication Agent pds_username user) any further access to the primary Oracle instance.

8 Log in to the previously existing Mirror Replication Agent instance, verify that it is in *Replicating* state, and allow replication to finish. To verify that replication has completed:

a Periodically issue the ra_statistics command, watching until all of the following statistics are zero (0):

Operation queue size
Operation data hash size
Input queue size
Output queue size

- b When all of these values are zero, note the Last QID Sent from the last set of statistics.
- c Issue the ra_locator update command so that Mirror Replication Agent retrieves the truncation point from Replication Server.
- d Wait, then issue the ra_locator command again and compare the displayed locator with that of the Last QID Sent. If they are different, wait and repeat this step.
- e Quiesce the Mirror Replication Agent instance by issuing the quiesce command.
- f Shut down the Replication instance by issuing the shutdown command.

9. Start and log in to the Mirror Replication Agent 15.1 instance and migrate the Mirror Replication Agent metadata by issuing the ra_migrate command.
10. Allow all users to access the primary Oracle.
11. Log in to the RSSD, and set the Replication Server locator to zero:

    ```
    rs_zeroltm source_ds, source_db
    ```

    Here, *source_ds* matches the previous Mirror Replication Agent instance values for rs_source_ds, and *source_db* matches the previous Mirror Replication Agent instance values for rs_source_db.

    Note This step is required because the format of the QID has changed in version 15.1, requiring the previous value held by Replication Server be replaced. The rs_source_ds and rs_source_db values were migrated from the previous version of Mirror Replication Agent and do not need to be changed.

12. In the Mirror Replication Agent 15.1 instance, do the following:

- If you want to have automatic archiving turned off, set pdb_include_archives to false. If you want to have automatic archiving turned on, set pdb_include_archives to true, set pdb_archive_path to the directory containing the archive logs, and set pdb_archive_remove to false. For more information, see “Redo and archive log setup” on page 35. Do this prior to resuming replication.
- Resume replication by issuing the resume command.

Migrating Mirror Replication Agent 15.1 when upgrading Oracle 9i to 10g

Mirror Replication Agent for Oracle migration to support upgrading Oracle 9i to Oracle 10g is a process similar to upgrading Mirror Replication Agent for Oracle 12.6 or 15.0 to Mirror Replication Agent 15.1.

Note Verify that the Mirror Replication Agent is quiesced prior to upgrading Oracle 9i to Oracle 10g, that is, the replication environment must have completed processing of all transactions prior to upgrading Oracle because the Mirror Replication Agent moves the truncation point to the end of the log during Mirror Replication Agent 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 upgrading Oracle, you must restart the Mirror Replication Agent and issue the ra_migrate command.
3. As with the log-based Mirror Replication Agent upgrade, you may have to re-configure the Mirror Replication Agent for Oracle instance to read archive logs depending on the configuration in Oracle. This may change following the Oracle upgrade.

If you are upgrading from log-based Mirror Replication Agent and upgrading Oracle 9i to Oracle 10g at the same time, you need to migrate Mirror Replication Agent 15.1 only once.

Upgrading Mirror Replication Agent for ASE

This section describes the upgrading process for Mirror Replication Agent for ASE.

Upgrading from Mirror Replication Agent for ASE 15.0

This section describes the steps for upgrading from Mirror Replication Agent for ASE version 15.0 to Mirror Replication Agent for ASE version 15.1.

❖ To upgrade from Mirror Replication Agent for ASE 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 15.0 instances.
3. Use the ma_admin utility script from the Mirror Replication Agent 15.1 installation *bin* directory to upgrade all the verifiable Mirror Replication Agent version 15.0 instances. Execute the ma_admin utility script with the *-u* option:

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

   where *src_directory* is the full path name to the Mirror Replication Agent version 15.0 installation directory. For example:

   - For UNIX:

   ```
   /sybase15/MA-15_1/bin/ma_admin.sh -u /sybase/MRA-15_0
   ```

   - For Windows:

   ```
   d:\sybase15\MA-15_1\bin\ma_admin -u d:\sybase\MRA-15_0
   ```

   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_1/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 15.1.

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 15.1 migration, refer to the *Release Bulletin* for Mirror Replication Agent 15.1.

---

6 You can now resume replication.

APPENDIX B

Using the sybfilter driver

Overview

Mirror Replication Agent must be able to read the Microsoft SQL Server log files directly. However, the Microsoft SQL Server process opens these log files with exclusive read permission, and the files cannot be read by any other processes, including Mirror Replication Agent. Before Mirror Replication Agent can replicate data, you must use the sybfilter driver to make the log files readable.

Requirements

For the sybfilter driver to work properly, the Microsoft Filter Manager Library must be version 5.1.2600.2978 or later. To determine the version of the library, right-click *c:\windows\system32\fltlib.dll* in Windows Explorer, select Properties, and click the Version tab in the Properties dialog. If the version is earlier than 5.1.2600.2978, go to the Microsoft Web site at http://windowsupdate.microsoft.com, and update your Windows system.

Installation and setup

Perform the following steps to install and set up the sybfilter driver.

Note On Windows Vista, you must be logged in as an Administrator to install, set up, and run the sybfilter driver.

❖ **To install and set up the sybfilter driver**

1. In Windows Explorer, navigate to the sybfilter driver installation directory. On Windows, this directory is located at *%SYBASE%\MA-15_1\system\<platform>*.

   Here, *<platform>* is winx86, winx64, or winvistax64.

2. Right-click the *sybfilter.inf* file to install the sybfilter driver.

   Note There can be only one installation of the sybfilter driver on a Windows machine. Once the driver is installed, it works for all Mirror Replication Agent for Microsoft SQL Server instances running on the same machine.

   ---

3. Under any directory, create a configuration file to store all log file paths for primary databases. The configuration file must have a *.cfg* suffix. For example, under the directory *%SYBASE%\MA-15_1\system\<platform>*, create a file named *LogPath.cfg*.

4. Add a system environment variable named *RACFGFilePath*, and set its value to the path of the configuration file.

   a. From the Control Panel, open System | Advanced | Environment Variables.

   b. Click New to add a new system variable.

   c. Name the variable *RACFGFilePath*, and set its value to the location of the your configuration file.

5. In Windows Explorer, navigate to *%SYBASE%\MA-15_1\bin*, and double-click the *sybfiltermgr.exe* file to start the sybfilter driver management console.

6. To start the sybfilter driver, enter `start` at the management console.

7 Add the log file path to the sybfilter driver with the user manager or by modifying the configuration file:

Note If the log file is a raw partition drive, there is no need add the raw partition path to the configuration file. The sybfilter driver will automatically—without checking the configuration file—change the read permission for the raw partition to nonexclusive so that the log file can be read.

- *User manager* – Use the add command in the management console:

  add *serverName dbName logFilePath*

  For example, to add the log file named *pdb2_log.ldf* at *D:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\* to the *dbName* database on the *serverName* data server, use the following:

  ```
  add myseverName dbName D:\Program
  Files\Microsoft SQL
  Server\MSSQL.1\MSSQL\Data\pdb2_log.ldf
  ```

  Note If you add the log file path with the user manager, the user manager refreshes all log paths to the sybfilter driver automatically after adding the log path into the configuration file.

- *Configuration file* – To add the log file path directly to the configuration file, open and manually edit the configuration file. The following is an example of log file path entries:

  ```
  [myserver, pdb1]
  log_file_path=D:\Program Files\Microsoft SQL
  Server\MSSQL.1\MSSQL\Data\pdb11_log.ldf
  log_file_path=D:\Program Files\Microsoft SQL
  Server\MSSQL.1\MSSQL\Data\pdb12_log.ldf
  [myserver, pdb2]
  log_file_path=D:\Program Files\Microsoft SQL
  Server\MSSQL.1\MSSQL\Data\pdb2_log.ldf
  ```

  Note Once you have added the log file paths to the configuration file, you must use the refresh command in the management console.

8 If you added a log file for your primary database before adding the log file path to the sybfilter driver, you must restart Microsoft SQL Server to make the log file readable.

9 At the management console, enter `check` to verify that log files are readable.

If some log files are unreadable, make sure the files have been created and that Microsoft SQL Server has been restarted, if necessary.

Troubleshooting

Consider the following issues when troubleshooting the sybfilter driver.

System environment variable is not set

Problem: The management console reports an error like the following:

```
ERROR: System environment variable RACFGFilePath has
not been set. Please set its value before starting this
manager. Fatal error occurs. Please press any key to
quit.
```

Workaround: Set the RACFGFilePath environment variable.

Configuration file does not exist

Problem: In response to the list command, the management console reports the following error:

```
ERROR: Cannot open config file.
```

Workaround: Create a configuration file.

Configuration file is not writeable

Problem: In response to the add command, the management console reports the following error:

```
ERROR: Cannot open config file.
```

Workaround: Add write permission for the configuration file.

Sybfilter command reference

The following commands are available in the sybfilter management console. For a list and description of commands, enter the help command at the sybfilter management console.

add	Add a log file path to the sybfilter driver and configuration file. *Syntax*: add *serverName dbName logFilePath* • *serverName* – The name of the Microsoft SQL Server • *dbName* – The name of the database to be replicated • *logFilePath* – the path of the database log
check	Check whether the sybfilter driver is running or not. Check for differences between path names in the configuration file and the sybfilter driver. Check whether or not configuration files for sybfilter are readable, and list any files that are not readable.
exit	Exit from the sybfilter management console.
help	Print help information for all sybfilter commands.
list	List all configured database names and the corresponding log file paths in the configuration file.
refresh	Refresh the content in the sybfilter configuration file.
remove	Remove a log file path from the sybfilter driver and configuration file. *Syntax*: remove *logFilePath* • *logFilePath* – the path of the database log
start	Start the sybfilter driver.
stop	Stop the sybfilter driver.

Glossary

This glossary describes 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 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, 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 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 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 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 Replication Agent that interacts with the other Replication Agent internal components to control and coordinate 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 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 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**.
route	A one-way message stream from a primary Replication Server to a replicate Replication Server. Routes carry data-changing commands (including those for RSSDs) and replicated functions (database procedures) between separate Replication Servers. See also **Replication Server**.
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 *standby 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. 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