SYBASE®

Primary Database Guide

Replication Agent™

15.1

Linux, Microsoft Windows, and UNIX

DOCUMENT ID: DC00269-01-1510-01

LAST REVISED: February 2008

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

Contents

About This Book ..... **vii**

CHAPTER 1 **Replication Agent for Microsoft SQL Server** **1**
Microsoft SQL Server-specific considerations 1
Microsoft SQL Server requirements 2
DDL Replication 3
New commands 6
New configuration parameters 6
Replication Agent connectivity 7
Replication Agent permissions 7
The sybfilter driver 8
Initialization of the primary data server and Replication Agent . 8
Microsoft isql tool 13
Character case of database object names 14
Format of origin queue ID 14
Datatype compatibility 15
Replicating ntext datatypes 18
Replication Agent objects in the Microsoft SQL Server primary database 19
Replication Agent object names 20
Table objects 20
Procedure objects 21
Marker objects 21
Trigger objects 22
Administering the transaction log 22
Using Windows authentication with Microsoft SQL Server 24
Running Replication Agent and Microsoft SQL Server on different machines 25
Replication Agent for Microsoft SQL Server setup test scripts 25
Before you begin 26
Create the primary table 27
Create the replicate table 28
Create the Replication Server connection for Replication Agent . 28

Create the replication definition 28
Test the replication definition 28
Create the subscription 28
Test the subscription 29
Initialize Replication Agent 29
Mark a primary table for replication 30
Start replication 30
Execute the test transaction script in Microsoft SQL Server ... 31
Check results of replication 31
Clean up the test environment 31

CHAPTER 2 Replication Agent for Oracle 33
Oracle-specific considerations 33
Replication Agent connectivity 34
Replication Agent permissions 34
Redo and archive log setup 35
Supplemental logging 38
Flashback feature unsupported 39
Setting ddl_username and ddl_password 39
Character case of database object names 41
Format of origin queue ID 42
Replication Server and RSSD scripts 43
Datatype compatibility 45
Oracle datatype restrictions 49
Oracle large object (LOB) support 50
Oracle user-defined types 53
Marking and unmarking sequences 55
Enabling and disabling replication for sequences 60
Running Replication Agent and Oracle on different machines 62
Real Application Clusters (RAC) 62
Automatic Storage Management 64
Replication Agent objects in the Oracle primary database 67
Replication Agent object names 68
Table objects 69
Marker objects 69
Sequences 69
Marked procedures 69
Transaction log truncation 70
Replication Agent for Oracle setup test scripts 71
Before you begin 72
Create the primary table 73
Create the replicate table 73
Create the Replication Server connection 73
Create the replication definition 74

Test the replication definition 74
Create the subscription 74
Test the subscription 75
Initialize Replication Agent 75
Mark a primary table for replication 75
Start replication 76
Execute the test transaction script in Oracle 76
Check results of replication 76
Clean up the test environment 76

CHAPTER 3 Replication Agent for UDB 79
IBM DB2 Universal Database-specific considerations 79
Feature differences in Replication Agent for UDB 80
Features not available in Replication Agent for UDB 81
IBM DB2 Universal Database Requirements 81
IBM DB2 Universal Database Administration Client 83
Replication Agent for UDB connectivity parameters 85
Handling repositioning in the log 86
Replication Agent for UDB behavior 87
Character case of database object names 89
Format of origin queue ID 89
Datatype compatibility 90
Replication Agent objects in the UDB primary database 93
Replication Agent objects 94
Administering the transaction log 97
Replication Agent for UDB setup test scripts 98
Before you begin 99
Create the primary table 100
Create the replicate table 100
Create the Replication Server connection 100
Create the replication definition 100
Test the replication definition 101
Create the subscription 101
Test the subscription 101
Initialize Replication Agent 101
Mark a primary table for replication 102
Start replication 102
Execute the test transaction script in the primary database 103
Check results of replication 103
Clean up the test environment 103

APPENDIX A **Upgrading Replication Agent 105**
Upgrading Replication Agent for Microsoft SQL Server 105

Upgrading a trigger-based Replication Agent (version 12.6 or 15.0) when the primary Microsoft SQL Server is version 2005 106
Upgrading a trigger-based Replication Agent (version 12.5, 12.6, or 15.0) when the primary Microsoft SQL Server is version 7 or 2000 112
Upgrading Replication Agent for Oracle..... 119
Upgrading a log-based Replication Agent (version 12.6 or 15.0). 119
Migrating Replication Agent 15.1 when upgrading Oracle 9i to 10g 123
Upgrading a trigger-based Replication Agent 12.5 when the primary Oracle is version 9i or 10g..... 123
Downgrading a log-based Replication Agent 15.1 to a trigger-based Replication Agent 12.5..... 128
Upgrading Replication Agent for UDB..... 132
Upgrading from Replication Agent for UDB 15.0..... 132
Migrating Replication Agent 15.1 when upgrading UDB 8 to 9 134

APPENDIX B **Using the sybfilter driver 137**
Overview 137
Requirements..... 137
Installation and setup 138
Troubleshooting 140
System environment variable is not set..... 140
Configuration file does not exist 140
Configuration file is not writeable 140
Sybfilter command reference 140

Glossary 143

Index 151

About This Book

Replication Agent™ 15.1 extends the capabilities of Replication Server® by supporting non-Sybase primary data servers in a Sybase replication system.

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

- IBM DB2 Universal Database (on UNIX and Microsoft Windows platforms)
- Microsoft SQL Server
- Oracle

Audience

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

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 Replication Agent support for non-Sybase data servers.

This book is organized as follows:

Chapter 1, “Replication Agent for Microsoft SQL Server,” describes replication system issues that are specific to Microsoft SQL Server, and details of the Replication Agent for Microsoft SQL Server.

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

Chapter 3, “Replication Agent for UDB,” describes replication system issues that are specific to IBM DB2 Universal Database, and details of the Replication Agent for UDB.

Appendix A, “Upgrading Replication Agent,” describes Replication Agent upgrades.

Appendix B, “Using the sybfilter driver,” describes use of the sybfilter driver.

Related documents

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

Replication Agent

- The Replication Agent *Administration Guide* introduces replication concepts and Sybase replication technology. This document also describes Replication Agent features and operations, and how to set up, administer, and troubleshoot the Replication Agent software.
- The Replication Agent *Reference Manual* describes all Replication Agent commands and configuration parameters in detail, including syntax, examples, and usage notes.
- The Replication Agent *Installation Guide* describes how to install the 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 Replication Agent setup.
- The Replication Agent 15.1 *Release Bulletin* contains last-minute information that was too late to be included in the books.

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

Java environment

Replication Agent 15.1 requires a Java Runtime Environment (JRE) on the machine that acts as the Replication Agent host.

- The Replication Agent 15.1 *Release Bulletin* contains the most up-to-date information about Java and JRE requirements.
- Java documentation available from your operating system vendor describes how to set up and manage your Java environment.

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 non-Sybase data server(s) supported by Replication Agent:

- IBM DB2 Universal Database
- 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/RAX-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, 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 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.

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 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

Replication Agent for Microsoft SQL Server

The term “Replication Agent for Microsoft SQL Server” refers to an instance of the 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 the Replication Agent that are unique to the Replication Agent for Microsoft SQL Server implementation.

Topic	Page
Microsoft SQL Server-specific considerations	1
Replication Agent objects in the Microsoft SQL Server primary database	19
Using Windows authentication with Microsoft SQL Server	24
Running Replication Agent and Microsoft SQL Server on different machines	25
Replication Agent for Microsoft SQL Server setup test scripts	25

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

Microsoft SQL Server-specific considerations

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

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

The following topics are included in this section:

- Microsoft SQL Server requirements
- DDL Replication
- New commands
- New configuration parameters
- Replication Agent connectivity
- Replication Agent permissions
- The sybfilter driver
- Initialization of the primary data server and 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:

- 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 Replication Agent on the same Microsoft SQL Server database. Be sure to disable Microsoft replication before using Replication Agent for Microsoft SQL Server.
- You cannot create a Microsoft SQL Server publication on the primary database where 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 Replication Agent. DDL commands should therefore only be replicated to other Microsoft SQL Server databases.

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

Note If DDL replication is enabled and the pdb_automark_tables parameter is set to true, the pds_username user who has the permission to create a table must also have the "ReplicationAdmin" role.

Replication Server uses the ddl_username parameter to execute DDL commands in the replicate 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 Replication Agent ddl_username and ddl_password parameters. The ddl_username parameter is the *replicate* database user name included in LTL for replicating DDL commands to the replicate or target database.

Permissions

In addition to the permission to execute all replicated DDL commands at the replicate database, the ddl_username should also have the impersonate permission granted for all users whose DDL commands may be replicated to the replicate database. This impersonate permission is necessary to switch session context in the replicate 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 replicate 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 replicate database to impersonate user1:

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

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

See the 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 replicate database, and *ddl_user* is the ddl_username user.

Note This grant command must be executed in the *replicate* 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 replicate database. You *cannot* replicate DDL commands from Microsoft SQL Server to non-Microsoft SQL Server replicate 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

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 Replication Agent *Reference Manual* for more information.

New commands

The following commands have been added for Replication Agent for Microsoft SQL Server:

- pdb_ownerfilter
- pdb_setrepddl
- pdb_skip_op
- ra_devicepath
- ra_helparticle
- ra_helpdevice
- ra_helpfield
- ra_helplocator
- ra_helpuser
- ra_truncatearticles
- ra_truncateusers
- ra_updatedevices
- rasd_backup
- rasd_restore
- rs_create_repdef
- rs_drop_repdef

For detailed information about these commands, see the Replication Agent *Reference Guide*.

New configuration parameters

The following parameters have been added for Replication Agent for Microsoft SQL Server:

- ddl_password
- ddl_username
- pdb_auto_create_repdefs
- pdb_automark_tables
- pds_dac_port_number
- rasd_backup_dir
- rasd_database
- rasd_mirror_tran_log
- rasd_trace_log_dir
- rasd_tran_log
- rasd_tran_log_mirror

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

Replication Agent connectivity

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

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

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

Replication Agent permissions

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

The user ID that the 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
- create role – Required to create the "ReplicationAdmin" role. Only the user with the "ReplicationAdmin" role can mark a table or procedure.
- db_owner role – Required to allow Replication Agent to execute sp_repltrans and sp_repldone in the primary database. This role is also required for primary database initialization.
- grant – Required to grant select permission on sys.sysschobs to the "ReplicationAdmin" role.
- sysadmin role – Required for Microsoft SQL Server data server initialization and deinitialization (using pdb_xlog init and pdb_xlog remove, respectively).

The sybfilter driver

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 Replication Agent. Before 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 Replication Agent

For Microsoft SQL Server initialization, 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 stand-alone mode. Subsequent executions of pdb_xlog 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 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 Replication Agent for the first time**

1 Stop the Microsoft SQL Server Analysis Service. From the Microsoft Windows Control Panel, choose Administrative Tools | Services, and find the service named SQL Server Analysis Service(*SERVER*), where *SERVER* is the name of your Microsoft SQL Server data server. Stop this service.

2 Make sure Microsoft SQL Server is configured to allow a remote dedicated administrative connection (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.

3 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.

4 Log in to your 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.

5 Also configure the following 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 Replication Agent *Installation Guide* and *Reference Manual*.

6 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.

7 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.

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

```
test_connection PDS
```

9 Initialize the Microsoft SQL Server data server and Replication Agent:

```
pdb_xlog init
```

In the primary database, Replication Agent creates all the tables, procedures, and triggers described in "Replication Agent objects in the Microsoft SQL Server primary database" on page 19. The sp_SybSetLogforReplTable and sp_SybSetLogforReplProc procedures are created in the mssqlsystemresource database with execute permission granted to Public. The "ReplicationAdmin" role is required to successfully execute these procedures.

10 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.

11 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 Microsoft SQL Server services, such as Microsoft SQL Server Agent service or the Microsoft SQL Server Analysis Service, you can start these services now.

Subsequent initialization

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

❖ **To subsequently initialize 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 Replication Agent, and set the pds_dac_port_number configuration parameter.

3 Configure the following 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 Replication Agent *Installation Guide* and *Reference Manual*.

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

```
test_connection PDS
```

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

```
pdb_xlog init
```

Final cleanup

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

❖ **To clean up all 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 Replication Agent:

```
drop user pds_username
```

5 Drop the "ReplicationAdmin" role:

```
drop role ReplicationAdmin
```

6 Remove the special marking procedures from the mssqlsystemresource database:

```
drop procedure marking_proc_name;

drop procedure sp_SybSetLogforReplTable;

drop procedure sp_SybSetLogforReplProc;
```

7 Stop Microsoft SQL Server in single-user mode by shutting down the Windows service or by issuing the shutdown command with the sqlcmd utility.

8 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.”

9 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 isql. You must use Microsoft isql (or a compatible tool) to access the Microsoft SQL Server database to execute some of the test scripts documented in this chapter.

Although the name of the Microsoft isql tool is the same as the Sybase tool called isql, the Sybase and Microsoft 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 Replication Agent administration port.

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 the 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 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 Replication Agent for Microsoft SQL Server.

Table 1-1: 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

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 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	Database-wide unique number	To retain the actual value assigned in Microsoft SQL Server, replicate to the varbinary(8) datatype.
uniqueidentifier	Globally unique identifier	varbinary	Globally unique identifier	No Sybase equivalent. Map to binary(38) or varbinary(38) datatype.
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	
nchar	Fixed length Unicode up to 4000 characters	unichar or char	32K	Actual maximum length is @@ncharsize * number of characters.

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
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	varbinary	32K	For replication to Replication Server 15.0 and earlier versions, the Sybase data type should be varbinary. For newer versions of Replication Server, use an opaque data type if the Replication Server provides it.

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. 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 stores double-byte ntext datatype values in little-endian byte order. By default, the byte order of ntext data is converted during replication to big-endian so that the data may be transmitted over networks using big-endian, which is the common network byte order.

If the target database is also Microsoft SQL Server, Microsoft SQL Server does not automatically convert the replicated data from the sent big-endian order to the little-endian order desired by Microsoft SQL Server. To support replicating ntext data to a Microsoft SQL Server (or other replicate server that does not provide the necessary conversion), you may force the byte order to be sent using the lr_ntext_byte_order property by specifying a value of big (for big-endian) or little (for little-endian) as desired to meet the expectations of your replicate database.

The lr_ntext_byte_order parameter is available for Microsoft SQL Server, and Oracle and is especially important for replication between two different database types and between databases that reside on different platforms. For example, for replication between two Microsoft SQL Server databases, both the primary and replicate database store data in little-endian byte order because Microsoft SQL Server only runs on Windows. Therefore, the lr_ntext_byte_order parameter should be set to little. However, if the replicate database is not a Microsoft SQL Server, you should determine its byte order and set the lr_ntext_byte_order parameter accordingly.

Note The default behavior of 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 replicate 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 true by default. When the ltl_big_endian_unitext property is true, 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, 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.

Replication Agent objects in the Microsoft SQL Server primary database

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

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

The Replication Agent objects are created by invoking the pdb_xlog command with the init keyword. When you invoke this command, 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 *RAX-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 Replication Agent.

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 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 pdb_xlog command to view the names of Replication Agent transaction log components in the primary database.

See the Replication Agent *Administration Guide* for details on setting up log object names.

Table objects

Table 1-4 lists the tables that are considered 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: 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 Replication Agent objects. The sp_SybSetLogforReplTable and sp_SybSetLogforReplProc procedures are created in the Microsoft SQL Server mssqlsystemresource system database. Although execute permission on these procedures is granted to Public, only the 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: 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

Marker objects

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

Table 1-6: 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*

Trigger objects

Table 1-7 lists Replication Agent trigger objects.

Table 1-7: 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

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

Truncating the transaction log

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

Replication Agent provides two options for automatic transaction log truncation:

- Periodic truncation, based on a time interval you specify
- Automatic truncation whenever 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 Replication Agent transaction log manually by invoking the pdb_truncate_xlog command at the 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.

Replication Agent for Microsoft SQL Server truncates the primary database log in units of transactions. After Replication Agent for Microsoft SQL Server receives the LTM locator from Replication Server, Replication Agent for Microsoft SQL Server queries the primary database to obtian the transaction ID of the newest transaction that can be truncated. 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 Replication Agent to control log truncation within Microsoft SQL Server. These commands require that the 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 Replication Agent is controlling the log truncation processing.

Using Windows authentication with Microsoft SQL Server

When running 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 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 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 Replication Agent for Microsoft SQL Server instance.

5 Log in to 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 Replication Agent as described in Replication Agent documentation.

Running Replication Agent and Microsoft SQL Server on different machines

Do the following to run Replication Agent and the primary Microsoft SQL Server data server on different machines.

❖ **To set up Replication Agent and Microsoft SQL Server to run on different machines**

1 Install the sybfilter driver on the same machine as the primary Microsoft SQL Server, and use this driver to make the transaction logs readable for Replication Agent.

2 On the machine on which the primary Microsoft SQL Server is running, share the drive or drives containing the transaction log files so that the drives can be mounted on the machine on which Replication Agent is to be installed.

3 Install the Replication Agent on a machine of the same type of hardware and operating system as the machine on which the primary Microsoft SQL Server data server is running.

4 Install the Microsoft SQL Server JDBC driver on the same machine as Replication Agent.

5 On the Replication Agent machine, map network drives that contain the primary Microsoft SQL Server database transaction log files. Use the ra_devicepath command to point Replication Agent to the mapped database log files.

Replication Agent for Microsoft SQL Server setup test scripts

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 Replication Agent software and the basic function of the other components in your replication system.

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

The 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 replicate Replication Server.

7 Test the subscription.

8 Initialize Replication Agent and create 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 replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database
- Used the sybfilter driver to make the Microsoft SQL Server transaction log files readable by Replication Agent.
- Installed the Replication Agent software, created a 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 Replication Agent test_connection command
- Confirmed that all servers are running
- Confirmed that the Replication Agent instance is in the *Admin* state

A word about the isql tool

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

Although the name of the Microsoft isql tool is the same as the Sybase tool called isql, the Sybase and Microsoft 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 Replication Agent administration port.

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

Create the primary table

Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the primary Microsoft SQL Server database and execute the *mssql_create_test_primary_table.sql* script in the *%SYBASE%\RAX-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 replicate data server and execute the *ase_create_test_replicate_table.sql* script in the *%SYBASE%\RAX-15_1\scripts\sybase* directory to create the replicate table in the replicate database.

Create the Replication Server connection for 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%\RAX-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%\RAX-15_1\scripts\sybase* directory so that all occurrences of *{pds}.{pdb}* contain the name of the Replication Server connection used by your 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%\RAX-15_1\scripts\sybase* directory to test the replication definition.

Create the subscription

The *rs_create_test_db_sub.sql* script in the *%SYBASE%\RAX-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 Replication Agent *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 *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to *{rds}.{rdb}*.

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

Initialize Replication Agent

If you have not already done so, initialize Replication Agent and create 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 8.

❖ **To initialize Replication Agent**

1 Use isql or another query processor to log in to the Replication Agent administration port.

2 Use the following command to initialize Replication Agent and create Replication Agent objects in the primary database:

```
pdb_xlog init
```

For general information, see the Replication Agent *Administration Guide*.

Mark a primary table for replication

Mark the test primary table (rax_test) that was created in the Microsoft SQL Server database by the *mssql_create_test_primary_table.sql* script in the *%SYBASE%\RAX-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 Replication Agent administration port.

2 Use the following command to mark the rax_test table for replication:

```
pdb_setreptable rax_test, mark
```

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

See the Replication Agent *Administration Guide* for more information about marking objects in the primary database.

Start replication

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

❖ **To start test replication**

1 Use isql or another query processor to log in to the Replication Agent administration port.

2 Use the following command to put the Replication Agent instance in the *Replicating* state:

```
resume
```

See the Replication Agent *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%\RAX-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%\RAX-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 replicate database and execute the *ase_select_test_replicate.sql* script in the *%SYBASE%\RAX-15_1\scripts\sybase* directory to verify that the transactions generated in the primary database were replicated to the test replicate table in the replicate database.

Clean up the test environment

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

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

1 Log in to the Replication Agent administration port using isql (or another query processor).

2 Use the following command to quiesce the Replication Agent instance:

```
quiesce
```

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

```
pdb_xlog remove, force
```

4 Log in to the replicate Replication Server and execute the following scripts:

- *%SYBASE%\RAX-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 *RDS.RDB* on the with replicate at clause to match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to *{rds}.{rdb}*.

- *%SYBASE%\RAX-15_1\scripts\sybase\rs_drop_test_db_repdef.sql* (to drop the test replication definition)

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

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

6 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_drop_test_primary_table.sql* script in the *%SYBASE%\RAX-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 isql with the -d option.

CHAPTER 2

Replication Agent for Oracle

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

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

Topic	Page
Oracle-specific considerations	33
Replication Agent objects in the Oracle primary database	67
Replication Agent for Oracle setup test scripts	71

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

Oracle-specific considerations

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

The following topics are included in this section:

- Replication Agent connectivity
- Replication Agent permissions
- Redo and archive log setup
- Supplemental logging
- Flashback feature unsupported
- 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
- Running Replication Agent and Oracle on different machines
- Real Application Clusters (RAC)
- Automatic Storage Management

Replication Agent connectivity

Connectivity between the 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 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 Replication Agent instance can connect to it. See the *Oracle Database Net Services Administrator's Guide* for more information.

Replication Agent permissions

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 instrument procedures for replication.
- create table – required to create tables in the primary database.
- create procedure – required to create rs_marker and rs_dump proc procedures.
- create public synonym – required to create synonyms for created tables in the primary database.
- create sequence – required to 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 the Replication Agent for Oracle instance must have read access to the Oracle *redo log* files and the Oracle archive directory that contains the *archive log* files to be accessed for replication. If the Replication Agent is configured to remove old archive files, the user must have update authority to the directory and the *archive log* files. If Oracle redo logs or archived redo logs are stored within ASM, the user who starts 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 Replication Agent for Oracle *must* be installed on a machine where it can directly access the Oracle *redo log* and *archive log* files.

You can access both online and archive logs by default. If you want to access only the online logs, the Replication Agent can be configured to do so, but it requires that you turn auto-archiving off and requires 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, the Replication Agent must be configured to use the directory path where the archive log files are located. To prevent conflicts with other archive file processes, you may wish to configure Oracle to produce archive log files into an additional directory used only for replication. Replication Agent can be configured to remove archive log files when they are no longer needed. Sybase recommends that you configure the Replication Agent to remove archive log files only if an additional directory is used.

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 the Oracle ASM, see "Automatic Storage Management" on page 64 for details.

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;
  ```

- Automatic redo log archiving must be disabled.

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

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

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

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 To 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 To check the setting of the LOG_ARCHIVE_START parameter:

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

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

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

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

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

Note This note applies only when pdb_include _archives is set to false. For redo log file processing after Replication Agent for Oracle is initialized, automatic archiving must *never* be enabled, even temporarily. If automatic archiving is re-enabled or manual archiving is performed, causing a redo log file not yet processed by the 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 the Replication Agent to access archive log files. Set pdb_include_archives to true, set pdb_archive_path to the directory location that contains the archive of the file that has been overwritten, and resume. After catching up, suspend the 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.

Flashback feature unsupported

The new Oracle flashback feature available in Oracle 10g is not supported in Replication Agent for Oracle. Oracle 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 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 replicate 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 Replication Agent *Reference Manual* for details on setting these parameters.

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

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 replicate 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 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 the Replication Agent.

To control the way 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 Replication Agent instance, and it varies according to the primary database type.

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

Table 2-1: 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_0/scripts/hds_oracle_udds.sql
$SYBASE/REP-15_0/scripts/hds_oracle_funcstrings.sql
$SYBASE/REP-15_0/scripts/hds_clt_ase_to_oracle.sql
$SYBASE/REP-15_0/scripts/hds_clt_oracle_to_ase.sql

The 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 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/RAX-15_1/scripts/oracle/hds_oracle_new_udds.sql
$SYBASE/RAX-15_1/scripts/oracle/
hds_oracle_new_setup_for_replicate.sql
$SYBASE/RAX-15_1/scripts/oracle/oracle_create_error_class_1_rs.sql
$SYBASE/RAX-15_1/scripts/oracle/
oracle_create_error_class_2_rssd.sql
$SYBASE/RAX-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_0/scripts/hds_oracle_udds.sql

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

$SYBASE/RAX-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 replicate database:

$SYBASE/RAX-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 replicate database. Use this script *instead of* the *hds_oracle_setup_replicate.sql* script provided in the Replication Server install 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 replicate database:

$SYBASE/REP-15_0/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_0/scripts/hds_clt_ase_to_oracle.sql

$SYBASE/REP-15_0/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/RAX-15_1/scripts/oracle/
  oracle_create_error_class_1_rs.sql

- Apply the following against your RSSD:

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

- Apply the following script at Replication Server:

  $SYBASE/RAX-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

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 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, 4712 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	
BFILE	4GB, locator points to large binary file	image	2GB	
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 53.
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 the following table identifies the replication definition datatypes that should be used.

Table 2-3: 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

Oracle datatype restrictions

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

Replication Server and 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 9.9×10^{125} and 1×10^{-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), 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 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 REF type
- Oracle VARRAY type
- Oracle NESTED TABLE 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 guide for a complete list of Oracle-supplied types.

Oracle large object (LOB) support

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

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

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

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

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

Special handling for *off row* LOBS

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

When an off-row LOB value is updated, the change recorded in the redo log is for the index that holds the LOB's data; the row the LOB belongs to is not changed. As a result, information is missing from the redo log to identify 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, 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 Replication Agent uses the additional column data from the associated operation to correctly build the where clause required to support replication.

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

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

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

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

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

- Missing accompanying change to the same row:

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

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

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

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

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

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

Oracle user-defined types

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

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 43.

❖ **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 ra_source_db.ra_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 replicate database.

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 60.

Oracle does not log information every time a sequence is incremented. Sequence replication occurs when the Replication Agent captures the system table updates that occur when the sequence's cache is refreshed. Therefore, the sequence value replicated when a sequence is marked for replication is the "next" sequence value to be used when the current cache expires. The result is that not every individual increment of a sequence is replicated, but the replicate 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 replicate 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/RAX-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 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 replicate Oracle database, you must create a stored procedure to support sequence replication. To do this, log into the replicate Oracle database as the maintenance user defined in your Replication Server connection to the replicate database. Then, execute the *$SYBASE/RAX-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 replicate database must have sufficient privileges to execute functions in the Oracle DBMS_SQL package. Also, this maintenance user must have authority at the replicate 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 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 replicate database.

- Use the following command to mark the sequence for replication when the sequence name you wish to increment at the replicate 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 wish 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's 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/RAX-15_1/scripts/oracle* directory at the replicate site to create a procedure named rs_update_sequence.

Unmarking a sequence

❖ **To unmark a sequence**

1 Log in to the 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 57.

Enabling replication for sequences

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

1 Log in to the 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 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.

Running Replication Agent and Oracle on different machines

Do the following to run Replication Agent and the primary Oracle data server on different machines.

❖ **To set up Replication Agent and Oracle to run on different machines**

1 Install the Replication Agent on a machine of the same type of hardware and operating system as the machine on which the primary Oracle data server is running.

2 Install the Oracle JDBC driver on the same machine as Replication Agent.

3 If the *timezone.dat* file is not in a location accessible to both machines, copy the *$ORACLE_HOME/oracle/zone.dat* file to the Replication Agent machine.

Note Be sure to copy the *timezone.dat* file of the Oracle server that Replication Agent is reading.

4 Set the Replication Agent pdb_timezone_file configuration parameter to the full path name of the *timezone.dat* file.

5 Make sure the Oracle online and archive logs reside in a location accessible to both machines. Use the ra_devicepath command to point Replication Agent to Oracle log files.

Real Application Clusters (RAC)

Replication Agent for Oracle 15.1 provides support for Oracle 10g RAC environments. When a 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, Replication Agent automatically configures itself to process the redo log information from all nodes.

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

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

Note Replication Agent must have read access to the *tnsnames.ora* file.

pdb_archive_path

The pdb_archive_path configuration parameter identifies the directory path where 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 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. 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 64 for details on how the pdb_archive_path configuration should be specified for ASM usage.

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 Replication Agent to remove archived log files that are no longer needed to support replication. If pdb_archive_remove is set to true, 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 Replication Agent is connected fails for any reason, Replication Agent attempts to reconnect to any surviving instance, choosing from the list of instances defined in the *tnsnames.ora* file entry the Replication Agent is configured to use. No manual intervention or configuration is required. If none of the instances are available, Replication Agent reports an error and continues processing as long as redo log file information is still available.

Automatic Storage Management

Replication Agent for Oracle supports the use of the Oracle Automatic Storage Management (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 Replication Agent for Oracle

The following ASM features are supported by 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, Replication Agent for Oracle recognizes the change and automatically updates its device information for affected log devices.

- 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 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 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, 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, 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 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 Replication Agent *Reference Manual*.

Replication Agent objects in the Oracle primary database

Note This section describes the Replication Agent objects for an Oracle database. For more general information, see the Replication Agent *Administration Guide*.

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

The Replication Agent objects are created by invoking the pdb_xlog command with the init keyword. When you invoke this command, 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 *RAX-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 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.

Replication Agent object names

There are two variables in the 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 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 pdb_xlog command to view the names of Replication Agent transaction log components in the primary database.

See the Replication Agent *Administration Guide* for details on setting up object names.

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

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

```
pdb_xlog
```

The pdb_xlog command returns a list of all the Replication Agent objects.

Table objects

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

Table 2-4: Replication Agent tables

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

Marker objects

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

Table 2-5: 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 Replication Agent objects.

Table 2-6: Replication Agent sequences

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

Marked procedures

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

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

Transaction log truncation

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

Replication Agent provides two options for automatic transaction log truncation:

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

When pdb_include_archives is set to true, the default, and pdb_remove_archives is set false, the 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, Replication Agent deletes from the pdb_archive_path location the archive redo logs that have already been processed. The Replication Agent is not responsible for archiving online transaction logs.

Note Sybase recommends you configure the 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, Replication Agent performs online redo log truncation (either scheduled or manual) by issuing the alter system command with the archive log sequence keywords. The command uses the log sequence number of the redo log file whose contents have been processed by the 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, 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 Replication Agent transaction log manually, at any time, by invoking the pdb_truncate_xlog command at the Replication Agent administration port.

Replication Agent for Oracle setup test scripts

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 Replication Agent software and the basic function of the other components in your replication system.

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

The 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 replicate Replication Server.

7 Test the subscription.

8 Create the 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 replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database
- Installed the Replication Agent software, created a 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 Replication Agent test_connection command
- Confirmed that all servers are running
- Confirmed that the 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/RAX-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 replicate data server and execute the *ase_create_test_replicate_table.sql* script in the *$SYBASE/RAX-15_1/scripts/sybase* directory to create the replicate table in the replicate 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/RAX-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/RAX-15_1/scripts/sybase* directory so that all occurrences of *{pds}.{pdb}* contain the name of the Replication Server connection used by your 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/RAX-15_1/scripts/sybase* directory to test the replication definition.

Create the subscription

The *rs_create_test_db_sub.sql* script in the *$SYBASE/RAX-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 Replication Agent *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 *RDS.RDB* on the with replicate at clause to match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially *{rds}.{rdb}*.

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

Initialize Replication Agent

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

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

1 Use isql or another query processor to log in to the Replication Agent administration port.

2 Use the following command to initialize Replication Agent and create Replication Agent objects in the primary database:

```
pdb_xlog init
```

See the Replication Agent *Administration Guide* for more information.

Mark a primary table for replication

Mark the test primary table (rax_test) that was created in the Oracle database by the *oracle_create_test_table.sql* script in the *$SYBASE/RAX-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 Replication Agent administration port.

2 Use the following command to mark the rax_test table for replication:

```
pdb_setreptable rax_test, mark
```

Note Make sure that replication is enabled for the rax_test table. See the Replication Agent *Administration Guide* for more information.

Start replication

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

❖ **To start test replication**

1 Use isql or another query processor to log in to the Replication Agent administration port.

2 Use the following command to put the Replication Agent instance in the *Replicating* state:

```
resume
```

See the Replication Agent *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/RAX-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/RAX-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 replicate database and execute the *ase_select_test_replicate.sql* script in the in the *$SYBASE/RAX-15_1/scripts/sybase* directory to verify that the transactions generated in the primary database were replicated to the test replicate table in the replicate database.

Clean up the test environment

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

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

1 Log in to the Replication Agent administration port using isql (or another query processor).

2 Use the following command to quiesce the Replication Agent instance:

```
quiesce
```

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

```
pdb_xlog remove, force
```

4 Log in to the replicate Replication Server and execute the following:

- *$SYBASE/RAX-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 *RDS.RDB* on the with replicate at clause to match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially *{rds.rdb}*.

- *$SYBASE/RAX-15_1/scripts/sybase/rs_drop_test_db_repdef.sql* (to drop the test replication definition)

- *$SYBASE/RAX-15_1/scripts/oracle/oracle_drop_rs_primary_connection.sql* (to drop the test connection to the primary database)

5 Log in to the replicate data server and execute the *$SYBASE/RAX-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/RAX-15_1/scripts/oracle/oracle_drop_test_table.sql* script to drop the test primary table.

CHAPTER 3

Replication Agent for UDB

The term “Replication Agent for UDB” refers to an instance of Replication Agent 15.1 software that is configured for a primary database that resides in an IBM DB2 Universal Database (UDB) server.

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

Topic	Page
IBM DB2 Universal Database-specific considerations	79
Replication Agent objects in the UDB primary database	93
Replication Agent for UDB setup test scripts	98

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

IBM DB2 Universal Database-specific considerations

This section describes general issues and considerations that are specific to using Replication Agent 15.1 with the IBM DB2 Universal Database server.

The following topics are included in this section:

- Feature differences in Replication Agent for UDB
- Features not available in Replication Agent for UDB
- IBM DB2 Universal Database Requirements
- IBM DB2 Universal Database Administration Client
- Replication Agent for UDB connectivity parameters
- Handling repositioning in the log

- Replication Agent for UDB behavior
- Character case of database object names
- Format of origin queue ID
- Datatype compatibility

Feature differences in Replication Agent for UDB

The following Replication Agent features have unique behavior in the Replication Agent for UDB:

- Initializing Replication Agent
- Marking a table for replication

Initializing Replication Agent

The Replication Agent for UDB provides the same features for initiliazing Replication Agent and creating its objects in the primary database as other implementations of the Replication Agent. Replication Agent for UDB creates only a few tables in the primary database to store its system information. The Replication Agent for UDB does not create any stored procedures or triggers in the primary database.

Because the Replication Agent for UDB requires access to the IBM DB2 Universal Database transaction log, the user ID that the Replication Agent uses to access the primary database must have either SYSADM or DBADM authority in the database; otherwise, the pdb_xlog init command returns an error. This user ID is stored in the Replication Agent pds_username configuration parameter.

For more information about Replication Agent objects, see “Replication Agent objects in the UDB primary database” on page 93.

Marking a table for replication

The Replication Agent for UDB provides the same features for marking and unmarking tables for replication as other implementations of the Replication Agent. However, the Replication Agent for UDB does *not* create any stored procedures or triggers in the primary database.

When marking a table for replication, Replication Agent for UDB alters the table to set the UDB DATA CAPTURE attribute to DATA CAPTURE CHANGES. When the table is unmarked, the table is altered to return to its original DATA CAPTURE attribute.

Note Do not manually change the DATA CAPTURE attribute of a table that has been marked for replication by Replication Agent for UDB. Doing so may adversely effect replication results.

Features not available in Replication Agent for UDB

The following Replication Agent features are not available with the Replication Agent for UDB:

- Stored procedure replication
- DDL replication

Note When you invoke Replication Agent commands related to these features, you receive an error.

Stored procedure replication

Stored procedure replication is not available with the Replication Agent for UDB. Therefore, the pdb_setrepproc command is not supported.

DDL replication

Replication of data definition language (DDL) commands and system procedures executed in the primary database is not supported.

IBM DB2 Universal Database Requirements

This section provides a summary of all the IBM DB2 Universal Database requirements.

- The database must be at least version 8.2.2 (the same as version 8.1 with FixPak 9) or later.
- The database must have a valid JDK path configured. To determine the JDK_PATH setting, use the following UDB command:

```
get dbm cfg
```

Note The JDK_PATH should point to a 32-bit JDK for all platforms except HP Itanium. For HP Itanium, the JDK_PATH should point to a 64-bit JDK.

- The database LOGARCHMETH1 configuration parameter must be set to LOGRETAIN or DISK:<*path*>. Here, <*path*> is a directory to which logs are archived. This enables archive logging in place of circular logging. To determine the LOGARCHMETH1 setting, use the following UDB command:

  ```
  get db cfg for <db-alias>
  ```

- The UDB connectivity autocommit parameter must be turned on (automcommit=*1*). The autocommit parameter is specified in the DB2 call level interface (CLI) configuration file for the primary database. If the autocommit parameter is not turned on, a deadlock problem can occur.

  - On Windows, the file is:

    %DB2DIR%\sqllib\db2cli.ini

    Here, *%DB2DIR%* is the path to the UDB client installation

  - On UNIX, the file is one of the following:

    - *$DB2DIR/cfg/db2cli.ini*

      Here, *$DB2DIR* is the path to the UDB client installation

    - *$HOME/sqllib/cfg/db2cli.ini*

      Here, *$HOME* is the home directory of the UDB instance owner (for UDB server installation)

- If you plan to use the Replication Agent log truncation feature, the database must have a user temporary system-managed tablespace with a page size of at least 8KB. This space requirement also applies to the Replication Agent initialization process.

- The user ID you specify as the pds_username user must have either SYSADM or DBADM authority to access the primary database transaction log.

- All the UDB environment variables must be set before you start the Replication Agent. Replication Agent uses the IBM DB2 Universal Database CLI driver to connect to the primary UDB database. For UNIX, the DB2 driver is contained in *libdb2.so*, *libdb2.sl*, or *libdb2.a* depending on the operating system. For Windows, the DB2 driver is contained in *db2cli.dll*. The 32-bit versions of the driver libraries are located in the *<library_path>/sqllib/lib32* directory. For all Replication Agent platforms except HP Itanium, the library path environment variable must point to 32-bit libraries, not to 64-bit libraries. The exact name of the library path environment variable depends on the operating system:
  - For Solaris and Linux, the library path variable is named LD_LIBRARY_PATH.
  - For HP-UX, the library path variable is named SHLIB_PATH.
  - For AIX, the library path variable is named LIBPATH.
  - For Windows, the library path variable is named PATH.
  - On Windows, the UDB server or client installation sets all necessary environment variables, so you need do nothing special.
  - On UNIX or Linux, you must source the UDB *db2cshrc* (for C-shell) or the *db2profile* (for Bourne and Korn shells) script before starting the Replication Agent. These scripts are located in one of the following paths:
    - *$DB2DIR/cfg*

      Here, *$DB2DIR* is the path to the UDB client installation
    - *$HOME/sqllib*

      Here, *$HOME* is the home directory of the UDB instance owner (for UDB server installation)

IBM DB2 Universal Database Administration Client

If the Replication Agent for UDB software is installed on a different host machine from the IBM DB2 Universal Database server, you must install the IBM DB2 Universal Database Administration Client on the same host machine as the Replication Agent.

If the Replication Agent for UDB software is installed on the same host machine as the IBM DB2 Universal Database server, a separate IBM DB2 Universal Database Administration Client is not required.

On a Windows system, you must configure an ODBC data source in the IBM DB2 Universal Database Administration Client, then use the database name and database alias specified for that ODBC data source when you configure Replication Agent for UDB connectivity.

On a UNIX system, instead of using ODBC, simply catalog the node and the primary database in UDB. Then, set the Replication Agent pds_datasource_name parameter to the database alias.

❖ **To catalog the remote TCP/IP node from the UDB client**

1 Log in as the DB2 instance owner.

2 Set up the DB2 environment.

In Korn shell, source the *db2profile* file:

```
. /home/db2inst1/sqllib/db2profile
```

In C shell, source the *db2cshrc* file:

```
source /home/db2inst1/sqllib/db2cshrc
```

3 Start the DB2 command-line processor by typing the db2 command.

4 Catalog the remote TCP/IP node using the following command at the DB2 prompt:

```
catalog tcpip node MYNODE remote MYHOST server XXXX
```

Here, *MYNODE* is the node name, *MYHOST* is the host name or IP address of the data server, and *XXXX* is the data server port number.

5 Verify the catalog entry:

```
list node directory
```

DB2 should return something similar to the following:

```
Node 1 entry:
      Node name            = MYNODE
      Comment              =
      Directory entry type = LOCAL
      Protocol             = TCPIP
      Hostname             = MYHOST
      Service name         = XXXX
```

❖ **To catalog the primary database from the UDB client**

1 Catalog the primary database using the following command at the DB2 prompt:

```
catalog database MYDB as MYDB_ALIAS at node MYNODE
```

Here, *MYDB* is the database name, *MYDB_ALIAS* is an alias for the database, and *MYNODE* is the node name used in the catalog command.

2 Verify the catalog entry:

```
list database directory
```

DB2 should return something similar to the following:

```
System Database Directory

Number of entries in the directory = 1

Database 1 entry:

Database alias         = MYDB_ALIAS
Database name          = MYDB
Node name              = MYNODE
Database release level = b.00
Comment                =
Directory entry type   = Remote
```

❖ **To configure pds_datasource_name**

1 In Replication Agent, set the pds_datasource_name parameter to the database alias:

```
ra_config pds_datasource_name, MYDB_ALIAS
```

Here, *MYDB_ALIAS* is the database alias that was used when cataloging the primary database.

2 Also set the following Replication Agent parameters:

pds_database_name
pds_username
pds_password

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

Replication Agent for UDB connectivity parameters

The following Replication Agent configuration parameters are required to configure a connection between the Replication Agent for UDB and a IBM DB2 Universal Database server:

- pds_username – must have DBADM authority, for example, repuser
- pds_password – for user ID specified in pds_username, for example, repuser
- pds_database_name – UDB database name, for example, TEST_DB1
- pds_datasource_name – UDB data source name, for example, TEST_DB1_DS

Handling repositioning in the log

The Replication Agent uses the value of the LTM locator received from the primary Replication Server to determine where it should begin looking in the IBM DB2 Universal Database transaction log for transactions to be sent to the Replication Server.

The Replication Agent for UDB uses the LTM locator value as follows:

- When the value of the LTM locator received from Replication Server and the LTM locator stored by Replication Agent are both zero (0), the Replication Agent positions the Log Reader component at the end of the IBM DB2 Universal Database transaction log.

Warning! In the event that both LTM locator values are zero, two specific conditions could cause data loss:

- Repositioning the Log Reader at the end of the transaction log may cause data loss if there are replicated transactions that have not been processed at the time the Log Reader is repositioned.
- When the Replication Agent Log Reader component goes to the *Replicating* state, it does so asynchronously. When you receive a prompt after invoking the resume command, the Log Reader component may not be finished getting into the *Replicating* state and positioning itself at the end of the log. If you mark a table immediately after the prompt returns from the resume command, the record containing the mark information could be written to the log before the Log Reader component has positioned itself. In that case, the Log Reader component will miss that record and not replicate any subsequent data for that table. To avoid this problem, wait a short time after invoking the resume command before you mark a table for replication.

- When both the value of the LTM locator received from Replication Server and the LTM locator stored by Replication Agent are not zero, Replication Agent uses the LTM locator value it received from Replication Server to determine the starting position of the oldest open transaction and positions the Log Reader component at that location in the IBM DB2 Universal Database transaction log.
- When the value of the LTM locator received from Replication Server is zero (0) and the value of the LTM locator stored by Replication Agent is not zero, Replication Agent uses the LTM locator value it has stored to determine the starting position of the oldest open transaction and positions the Log Reader component at that location in the IBM DB2 Universal Database transaction log.

Replication Agent for UDB behavior

The following Replication Agent issues are unique to Replication Agent for UDB:

- Marking tables immediately after going to *Replicating* state, when the value of the LTM locator is 0 (zero)
- Forcing applications off the primary database with the DB2 FORCE APPLICATION command
- Determining the read buffer size
- Replicating LOBs

Marking tables immediately after resume when LTM locator is zero

When the Replication Agent instance goes to Replicating state, the Log Reader component reads the primary database transaction log and uses the value of the origin queue ID to determine the position in the log to start reading. When the value of the LTM locator is 0 (zero), the Log Reader starts reading at the end of the log.

Because the Log Reader's operation is asynchronous, the Replication Agent instance can return to the operating system prompt after the resume command but *before* the Log Reader has completed its start-up process. If you immediately invoke the pdb_setreptable command to mark a table for replication after the resume command returns, the mark object entry can be placed in the transaction log before the Log Reader finds the end of the log. In that event, the Log Reader will miss the mark table entry and table marking will fail.

To avoid this problem, wait 5 to 10 seconds after invoking the resume command before invoking the pdb_setreptable command to mark a table.

Forcing applications off the database

The DB2 FORCE APPLICATION command causes the data server to drop its connections with an application. The FORCE APPLICATION ALL command causes the data server to drop its connections with all applications.

If you invoke the FORCE APPLICATION command and specify either the Replication Agent application handle or the ALL keyword, the data server drops its connections with the Replication Agent instance. In that event, the Replication Agent receives DB2 error code -30081 and cannot recover, so the Replication Agent instance shuts itself down.

To avoid this situation, invoke the Replication Agent shutdown command *before* using the DB2 FORCE APPLICATION command.

Determining read buffer size

The Replication Agent for UDB LogReader component uses the value of the max_ops_per_scan parameter to determine the maximum number of bytes to be read from the transaction log during each scan. Because the LogReader reads bytes, it requires a buffer to store the bytes read.

The LogReader component determines the maximum size of this buffer by multiplying the value of the max_ops_per_scan parameter by 10. For example, if the value of the max_ops_per_scan parameter is 1000 (the default), the size of the LogReader read buffer is 10,000 bytes.

It is very difficult to identify a minimum buffer size that will always work. The value range of max_ops_per_scan is 25 to 2,147,483,647, which means the smallest size of the buffer is 250 bytes.

If the read buffer size is too small to read one operation, LogReader shuts down the Replication Agent instance and reports a -30081 error. Unfortunately, this error message covers general communication errors, not just an insufficient buffer size.

Replicating LOBs

To replicate large-object columns of a marked table that also contains GRAPHIC or VARGRAPHIC columns, Replication Agent for UDB requires that the table have a primary key.

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 the Replication Agent.

To specify the character case option you want, set the value of the ltl_character_case configuration parameter with one of the following 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 the IBM DB2 Universal Database server, database object names are stored in all uppercase.

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 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 Replication Agent for UDB.

Table 3-1: Replication Agent for UDB origin queue ID

Character	Bytes	Description
0-3	2	Database generation ID
4-19	8	Operation sequence number
20-35	8	Transaction ID
36-51	8	First operation sequence number of oldest active transaction
52-55	2	Operation type (begin = 0, data/LOB = 1, commit/rollback = 7FFF)
56-59	2	LOB sequence ID
60-63	2	Unused

Datatype compatibility

Replication Agent for UDB processes transactions and passes data to the primary Replication Server.

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

The following table describes the default conversion of IBM DB2 Universal Database datatypes to Sybase datatypes.

For each datatype in Table 3-2, lengths in the second column are described as:

- Character datatypes – maximum number of bytes.
- Graphic datatypes – maximum number of characters.
- Numeric datatypes – range from smallest to largest values.
- Temporal datatypes – range from earliest time to latest time.

Table 3-2: IBM DB2 Universal Database to Sybase default datatype mapping

DB2 UDB datatype	DB2 UDB length/range	Sybase datatype	Sybase length/range	Notes
BIGINT	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	bigint	10^{-38} to 10^{38}, 38 significant digits	
BLOB	variable length, 2GB, binary data	image	2GB	
CHAR	254 bytes	char	32K	

DB2 UDB datatype	DB2 UDB length/range	Sybase datatype	Sybase length/range	Notes
CHAR FOR BIT DATA	254 bytes, binary data	binary	32K	
CLOB	variable length, 2GB, character data	text	2GB	
DATE	0001-01-01 to 9999-12-31	char, date, or datetime	32K (char)	If the pdb_convert_datetime parameter is false, DATE values are sent as char datatype strings. If the pdb_convert_datetime parameter is true, DATE values are converted to date or datetime values.
DBCLOB	variable length, 2GB, double-byte character data	unitext or image	2GB	For Replication Server 15.0 and later versions, DBCLOB maps to unitext. For earlier versions of Replication Server, DBCLOB maps to image.
DECIMAL	$-10^{31}+1$ to $10^{31}-1$, 31 digits of precision	decimal	10^{-38} to 10^{38}, 38 significant digits	
DOUBLE				See FLOAT.
FLOAT	8 bytes, -1.79769^{308} to 1.79769^{308}	float	Precision and range corresponds to a C double datatype, approximately 16 significant digits	Extremely small values are truncated to 16 digits to the right of the decimal. Extremely large values retain their precision.
GRAPHIC	127 characters, double-byte character data	unichar	32K	
INTEGER	-2,147,483,648 to 2,147,483,647	int	-2,147,483,648 to 2,147,483,647	
LONG VARCHAR	variable length, 32,700 bytes, character data	text	2GB	
LONG VARCHAR FOR BIT DATA	32,700 bytes, binary data	image	2GB	
LONG VARGRAPHIC	16,350 characters, double-byte character data	unitext or image	2GB	For Replication Server 15.0 and later versions, LONG VARGRAPHIC maps to unitext. For earlier versions of Replication Server, LONG VARGRAPHIC maps to image.

DB2 UDB datatype	DB2 UDB length/range	Sybase datatype	Sybase length/range	Notes
NUMERIC (synonym for DECIMAL)				See DECIMAL.
REAL	-3.402^{38} to 3.402^{38}	decimal	10^{-38} to 10^{38}, 38 significant digits	
SMALLINT	-32,768 to 32,767	smallint	-32,768 to 32,767	
TIME	00:00:00 to 24:00:00	char, time, or datetime	32K (char)	
TIMESTAMP	0001-01-01-00.00.00.000000 to 9999-12-31-24.00.00.000000	char or datetime	32K (char)	If the pdb_convert_datetime parameter is false, TIMESTAMP values are sent as char datatype strings. If the pdb_convert_datetime parameter is true, TIMESTAMP values are converted to datetime values.
VARCHAR	32,672 bytes	varchar	32K	
VARCHAR FOR BIT DATA	32,672 bytes, binary data	varbinary	32K	
VARGRAPHIC	16,336 characters, double-byte character data	univarchar	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 the following table identifies the replication definition datatypes that should be used.

Table 3-3: 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

Replication Agent objects in the UDB primary database

Note This section describes the schema and details of Replication Agent objects for a primary database that resides in the IBM DB2 Universal Database server. For more general information, see the Replication Agent *Administration Guide*.

Replication Agent creates objects in the UDB primary database to assist with replication tasks. Replication Agent also uses the native database transaction log maintained by the IBM DB2 Universal Database server to capture transactions in the primary database for replication.

Replication Agent for UDB creates tables in the primary database for its system information. Replication Agent also creates Java stored procedures used to truncate transaction log files in the primary database. To create the procedures, Replication Agent installs jar files into the primary database.The Replication Agent system tables and procedures are created when the pdb_xlog command is invoked with the init keyword. When you invoke this command, Replication Agent generates a SQL script that is run in the primary database. This script is stored in the *create.sql* file in the *RAX-15_1\inst_name\scripts\xlog* directory.

The *create.sql* script creates the Replication Agent objects. These objects must be created before any tables can be marked for replication in the primary database.

Note The jar files are installed when the pdb_xlog init command is executed. The pdb_xlog remove command uninstalls the jars from the primary database. You must issue pdb_xlog remove command before reinitializing Replication Agent for UDB.

Replication Agent objects

This section describes objects that Replication Agent creates in the primary database to support replication.

Replication Agent object names

There are two variables in the Replication Agent 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 table name to make that name unique in the database.

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

If this value conflicts with the names of existing database objects in your primary database, you can change the value of the pdb_xlog_prefix parameter by using the ra_config command.

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

You can use the pdb_xlog command to view the names of Replication Agent objects in the primary database.

See the Replication Agent *Administration Guide* for details on setting up replication object names.

Table objects

Table 3-4 lists the Replication Agent tables. No permissions are granted on these tables when they are created. All of these tables contain at least one index, and some contain more than one index.

Table 3-4: Replication Agent tables

Table	Database name
System table	*prefix*xlog_system_
Marked objects table	*prefix*marked_objs_*xxx*
LOB columns table	*prefix*blob_columns_*xxx*
Log Admin work table	*prefix*rawork_*xxx*
Proc active table	*prefix*procactive_*xxx*
Force record table	*prefix*force_record_*xxx*
rs_marker shadow table	*prefix*markersh_*xxx*
rs_dump shadow table	*prefix*dumpsh_*xxx*

Java procedure objects

Replication Agent for UDB installs *SYBRAUJAR.jar* and *SYBTRUNCJAR.jar* into the following directories.

- On Windows, the files are installed in *$DB2DIR/SQLLIB/FUNCTION/jar/pds_username*. Here, *$DB2DIR* is the path to the UDB installation, and *pds_username* is the value of pds_username.
- On UNIX, the files are installed in *$HOME/sqllib/function/jar/pds_username*. Here, *$HOME* is the home directory of the UDB instance owner, and *pds_username* is the value of pds_username.

These jar files implement several Java procedures in the UDB primary database. Table 3-5 lists the Java procedures that are created and used in log truncation.

Table 3-5: Java procedures for truncation

Procedure	Database name
Retrieves the name of the log file that contains the current LSN	*prefix*get_log_name_
Retrieves the version of the get_log_name Java class	*prefix*get_version_str_
Truncates the database log file or files from the archive log directory	*prefix*trunc_log_files_
Retrieves the version of the trunc_log_files Java class	*prefix*get_trunc_ver_str_

Getting actual names of the Replication Agent objects

The Replication Agent instance generates the names of its database objects. To find out the actual names of these objects, use the pdb_xlog command.

❖ **To find out the names of Replication Agent objects**

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

  ```
  pdb_xlog
  ```

  The pdb_xlog command returns a list of objects in the primary database.

Marked objects table

One of the Replication Agent objects is the **marked objects table**. The marked objects table contains an entry for each marked table in the primary database. Each marked table entry contains the following information:

- Name of the marked primary object (table)
- Primary object's replicated name
- Type of the primary object (table only, in Replication Agent for UDB)
- "Replication enabled" flag for the primary object
- Owner of the primary object
- "Send owner" flag
- Tablespace ID of the primary object
- Table ID of the primary object
- "Convert datetime" flag

- Original value of the table's DATA CAPTURE attribute

Administering the transaction log

The Replication Agent for UDB supports truncating transaction logs. All IBM DB2 Universal Database transaction logs are maintained through the data server.

Truncating the transaction logs

Replication Agent for UDB can be configured to truncate transaction logs from either the active or the archive log directory. When you have enabled UDB archiving with LOGARCHMETH1, you can also configure a second archive location by setting the LOGARCHMETH2 UDB configuration parameter. UDB then archives logs into the two directories. You can then configure Replication Agent to automatically truncate the processed archives from one of these directories.

To configure Replication Agent to truncate logs from the archive log directory, set the following configuration parameters:

- Set pdb_archive_path to point to the location specified by either LOGARCHMETH1 or LOGARCHMETH2.
- Set pdb_archive_remove to true if you want Replication Agent to delete the archives that are no longer necessary.

  Note The pdb_archive_remove property is set to false by default. You must configure the pdb_archive_path property before setting pdb_archive_remove to true.

- To enable automatic truncation, set truncation_type to interval, and set truncation_interval to a value greater than "0" (zero), which will cause the log files to be deleted at the designated interval. Alternately, set truncation_type to locator_update, which causes truncation to occur each time Replication Agent receives a new LTM Locator value from the primary Replication Server.

- For manual truncation, execute the Replication Agent pdb_truncate_xlog command, which causes Replication Agent to immediately truncate the transaction log based on the most recent truncation point received from the primary Replication Server.

Warning! If you enable truncation without also setting pdb_archive_path, Replication Agent deletes the *primary database* log files it no longer needs from the *active* log directory.

When UDB truncate runs, the oldest LSN for which Replication Agent has not processed a commit/rollback (oldest active LSN) is obtained and the archive log file that contains the LSN is determined. Then, all archive log files up to but not including the file with the oldest active LSN are deleted.

For more information on these properties refer to the Replication Agent *Reference Manual*. For a more detailed description of truncating, see "Chapter 3, Administering Replication Agent" of the Replication Agent *Administration Guide*. For a list of the stored procedures used for truncation, see Table 3-5 on page 96.

Replication Agent for UDB setup test scripts

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 Replication Agent software and the basic function of the other components in your replication system.

These test scripts are located in the *scripts* subdirectory under the Replication Agent base directory, for example, *RAX-15_1/scripts*.

The 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 replicate Replication Server.

7 Test the subscription.

8 Initialize Replication Agent, and create Replication Agent objects 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 IBM DB2 Universal Database server
- Installed a data server to act as a replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database (created a database connection for the replicate database in the RRS)
- Installed the Replication Agent software, created a Replication Agent for UDB 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 Replication Agent test_connection command

- Confirmed that all servers are running
- Confirmed that the Replication Agent for UDB instance is in the *Admin* state

Create the primary table

Use your database access tool (such as Command Line Processor) to log in to the IBM DB2 Universal Database server and execute the *udb_create_test_primary_table.sql* script in the *$SYBASE/RAX-15_1/scripts/ibmudb* 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 replicate data server and execute the *ase_create_test_replicate_table.sql* script in the *$SYBASE/RAX-15_1/scripts/sybase* directory to create the replicate table in the replicate 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/RAX-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* in the *$SYBASE/RAX-15_1/scripts/sybase* directory so that all occurrences of *{pds}.{pdb}* contain the name of the Replication Server connection used by your 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/RAX-15_1/scripts/sybase* directory to test the replication definition.

Create the subscription

The *rs_create_test_db_sub.sql* script in the *$SYBASE/RAX-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 Replication Agent *Administration Guide* for more information about database materialization.

1 Edit the script file (*rs_create_test_db_sub.sql*) so 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*RDS.RDB* on the with replicate at clause match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to *{rds}.{rdb}*.

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

Initialize Replication Agent

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

❖ **To initialize Replication Agent**

1 Use isql or another query processor to log in to the Replication Agent administration port.

2 Use the following command to initialize Replication Agent and to create Replication Agent objects in the primary database:

```
pdb_xlog init
```

For more information, see the Replication Agent *Administration Guide.*

Mark a primary table for replication

Mark the test primary table (rax_test) that was created in the primary database in the IBM DB2 Universal Database server by the *udb_create_test_primary_table.sql* script in the *$SYBASE/RAX-15_1/scripts/ibmudb* directory.

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

1 Use isql or another query processor to log in to the Replication Agent administration port.

2 Use the following command to mark the rax_test table for replication:

```
pdb_setreptable rax_test, mark
```

Note Make sure that replication is enabled for the rax_test table. See the Replication Agent *Administration Guide* for more information.

Start replication

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

❖ **To start test replication**

1 Use isql or another Open Client™ tool to log in to the Replication Agent administration port.

2 Use the following command to put the Replication Agent instance in *Replicating* state:

```
resume
```

Note Wait 5-10 seconds after you invoke the resume command before you execute the test transaction script.

See the Replication Agent *Administration Guide* for more information about the *Replicating* state.

Execute the test transaction script in the primary database

Use your IBM DB2 Universal Database access tool (such as Command Line Processor) to log in to the IBM DB2 Universal Database server and execute the *udb_primary_test_transactions.sql* script in the *$SYBASE/RAX-15_1/scripts/ibmudb* directory to generate transactions in the primary table.

Check results of replication

Use your IBM DB2 Universal Database access tool (such as Command Line Processor) to log in to the IBM DB2 Universal Database server and execute the *udb_select_test_primary.sql* script in the *$SYBASE/RAX-15_1/scripts/ibmudb* directory to view the changes in the test primary table.

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

Clean up the test environment

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

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

1 Log in to the Replication Agent administration port using isql (or another query processor).

2 Use the following command to quiesce the Replication Agent instance:

```
quiesce
```

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

```
pdb_xlog remove, force
```

4 Log in to the replicate Replication Server and execute the following scripts:

- *$SYBASE/RAX-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 from the primary data server and primary database. These values are initially *{pds}.{pdb}*. Also, change the values for*RDS.RDB* on the with replicate at clause match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to *{rds}.{rdb}*.

- *$SYBASE/RAX-15_1/scripts/sybase/rs_drop_test_db_repdef.sql* (to drop the test replication definition)

- *$SYBASE/RAX-15_1/scripts/sybase/ rs_drop_test_primary_connection.sql* (to drop the test connection to the primary database)

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

6 Use your IBM DB2 Universal Database access tool (such as Command Line Processor) to log in to the IBM DB2 Universal Database server and execute the *$SYBASE/RAX-15_1/scripts/ibmudb/ udb_drop_test_primary_table.sql* script to drop the test primary table.

APPENDIX A

Upgrading Replication Agent

Topic	Page
Upgrading Replication Agent for Microsoft SQL Server	105
Upgrading Replication Agent for Oracle	119
Upgrading Replication Agent for UDB	132

Upgrading Replication Agent for Microsoft SQL Server

Unlike previous versions, Replication Agent for Microsoft SQL Server 15.1 is log-based and, to directly access the Microsoft SQL Server transaction log files, must be installed on the same Windows host on which the primary Microsoft SQL Server is running. Replication Agent for Microsoft SQL Server 15.1 cannot be installed on a UNIX or Linux host. Because of this, how you upgrade an existing Replication Agent instance depends on two things: (1) where the existing instance of the previous version of Replication Agent is installed, and (2) the current version of the primary data server.

When you use any of the upgrade procedures described in this section, the new Replication Agent for Microsoft SQL Server 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 Replication Agent for Microsoft SQL Server 15.1 does not support downgrading to an earlier version.

The following sections cover these topics:

- Upgrading a trigger-based Replication Agent (version 12.6 or 15.0) when the primary Microsoft SQL Server is version 2005

- Upgrading a trigger-based Replication Agent (version 12.5, 12.6, or 15.0) when the primary Microsoft SQL Server is version 7 or 2000

Upgrading a trigger-based Replication Agent (version 12.6 or 15.0) when the primary Microsoft SQL Server is version 2005

This section describes how to upgrade Replication Agent 12.6 or 15.0 to 15.1 in the following *specific* situations:

Note All of these upgrades are based on the primary Microsoft SQL Server 2005.

- Replication Agent 15.1 is installed on the *same* Windows host as the previous version of Replication Agent.
- Replication Agent 15.1 is installed on a *different* Windows host than the previous version of Replication Agent.
- Replication Agent 15.1 is installed on a Windows host, but the previous version of Replication Agent is installed on a *UNIX* or *Linux* host.

Note Replication Agent 15.1 must be installed on the same host on which the primary Microsoft SQL Server is running.

Replication Agent 15.1 is installed on the *same* Windows host as the previous version, and the current version of Microsoft SQL Server is 2005

❖ **To upgrade when previous Replication Agent versions and Replication Agent 15.1 are on the same Windows host**

1 For *each* existing Replication Agent for Microsoft SQL Service instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 Install the Replication Agent 15.1 software as described in "Installing the Replication Agent software" in the Replication Agent 15.1 *Installation Guide*. Sybase recommends that you install Replication Agent 15.1 into the same SYBASE directory as the previous version of Replication Agent.

3 On the host on which the primary data server is running, verify that the Microsoft Filter Manager Library is the correct version as described in "The sybfilter driver" on page 8.

4 Verify that your primary data server meets the requirements described in “Microsoft SQL Server requirements” on page 2.

5 Stop the Microsoft SQL Server Analysis Service. From the Microsoft Windows Control Panel, choose Administrative Tools | Services, and find the service named SQL Server Analysis Service(*SERVER*), where *SERVER* is the name of your Microsoft SQL Server data server. Stop this service.

6 Make sure Microsoft SQL Server is configured to allow a remote DAC connection. You can use the Microsoft SQL Server Surface Area Configuration tool to enable a remote DAC connection:

   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.

7 In the primary Microsoft SQL Server, grant each previously existing pds_username user the additional required privileges. See “Replication Agent permissions” on page 7.

8 Use the Replication Agent 15.1 sybfilter driver to make the Microsoft SQL Server transaction log files readable by Replication Agent. For details on installing and using the sybfilter driver, see Appendix B, “Using the sybfilter driver.” However, at this time, it is not necessary to stop and restart Microsoft SQL Server since it will be done in a later step in this procedure.

9 Create the 15.1 version of all valid existing Replication Agent instances:

Note This step creates new Replication Agent 15.1 instances for all valid existing instances of the previous version of Replication Agent, regardless of whether the existing instances are for Oracle, IBM DB2 Universal Database, or Microsoft SQL Server. To complete the upgrade for Oracle or UDB 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 Open a command window.

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

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

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

```
ra_admin -u src_directory
```

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

```
ra_admin -u d:\sybase\RAX-15_0
```

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

10 If necessary, set the RA_JAVA_DFLT_CHARSET environment variable in each of the Replication Agent 15.1 *RUN_instance* scripts 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 Replication Agent 15.1 *Administration Guide*.

11 Determine the primary Microsoft SQL Server DAC port number:

a Using a text editor, open the ERRORLOG file in the root 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,” and you will find an entry similar to this:

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

c Make note of the port number specified; it will be used in a later step in this procedure.

12 To prevent any loss of replicated data, deny users other than the previously existing Replication Agent pds_username users from any further access to the primary databases.

13 For *each* of the previously existing Replication Agent for Microsoft SQL Server instances, 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 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 Replication Agent instance by issuing the quiesce command.

f Shut down the Replication instance by issuing the shutdown command.

14 When all of the previously existing Replication Agent instances have been shut down, stop the Microsoft SQL Server service:

a In Control Panel | Administrative Tools | Services, find the service named SQL Server (*SERVER*). Here, *SERVER* is the name of your SQL Server data server. For example,

```
SQL Server (TEAMSTER)
```

b Stop the service.

15 Restart Microsoft SQL Server in single-user mode by opening a new command window and executing this command:

```
"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.

16 Download and install the Microsoft SQL Server JDBC driver, and set the CLASSPATH environment variable, as described in the Replication Agent 15.1 *Installation Guide*. If the CLASSPATH contains any other Microsoft SQL Server JDBC driver, be sure to remove it. Only the Microsoft SQL Server JDBC driver required by Replication Agent 15.1 can be in the CLASSPATH.

17 Start and log in to *each* of the Replication Agent for Microsoft SQL Server 15.1 instances and do the following:

   a Verify that Microsoft SQL Server has been configured to allow a remote DAC connection.

   b Set the pds_dac_port_number configuration parameter:

      ```
      ra_config pds_dac_port_number, port
      ```

      Here, *port* is the DAC port number you found in step 9.

   c Set the rs_charset configuration parameter to match the Replication Server character set, as described in the Replication Agent 15.1 *Reference Manual*.

   d Use the test_connection command to ensure that Replication Agent can connect to both Microsoft SQL Server and Replication Server.

   e Initialize the Replication Agent instance and migrate the Replication Agent instance's metadata by issuing the ra_migrate command.

      When this command executes in the first Replication Agent 15.1 instance, it will also initialize the Microsoft SQL Server. In subsequent Replication Agent 15.1 instances, it will only initialize the instance and migrate the instance's metadata.

18 Stop the Microsoft SQL Server in single-user mode:

   a 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, password, and Microsoft SQL Server name.

   b Issue the shutdown command.

19 Restart Microsoft SQL Server in multi-user mode (normal start):

   a In Control Panel | Administrative Tools | Services, find the service named SQL Server (*SERVER*). Here, *SERVER* is the name of your Microsoft SQL Server data server. For example,

      ```
      SQL Server (TEAMSTER)
      ```

   b Start the service.

20 Log in to *each* of the Replication Agent Microsoft SQL Server 15.1 instances and resume replication by issuing the resume command.

21 Allow all users to access the primary databases.

Replication Agent 15.1 is installed on a *different* Windows host than the previous version of Replication Agent, and the current version of Microsoft SQL Server is 2005

❖ **To upgrade when Replication Agent 15.1 is on a different Windows host than previous versions**

1 For *each* existing Replication Agent for Microsoft SQL Server instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 Make the installation directory of the previous version of Replication Agent available as the source directory for the upgrade procedure:

- On the Windows host on which the previous version of Replication Agent is running, share the drive on which the Replication Agent is installed.
- On the Windows host on which Replication Agent 15.1 is installed, map a network drive to the host and drive in the previous step. On this mapped network drive, use the installation directory of the previous version of Replication Agent as the source directory for the upgrade.

3 Starting with step 3 of the procedure "Replication Agent 15.1 is installed on the same Windows host as the previous version, and the current version of Microsoft SQL Server is 2005" on page 106, follow the steps to complete the upgrade.

Replication Agent 15.1 is installed on a Windows host, but the previous version is installed on a UNIX or Linux host, and the current version of Microsoft SQL Server is 2005

❖ **To upgrade when Replication Agent 15.1 is on Windows, but the previous version of Replication Agent is on UNIX or Linux, and the current version of Microsoft SQL Server is 2005**

1 For *each* existing Replication Agent for Microsoft SQL Server instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 On the Windows host, create a substitute installation directory and instance subdirectories for the existing instances in the previous version of Replication Agent that is on the UNIX host:

a On the Windows host, in the *SYBASE* directory in which you installed Replication Agent 15.1, create a directory with the name of the previous version of Replication Agent that you are upgrading from. For Replication Agent 12.6, name the directory *RAX-12_6*; for Replication Agent 15.0, name the directory *RAX-15_0*. You will use this newly-created directory as the source directory for the upgrade.

b In the directory you just created, create a subdirectory for *each* existing Replication Agent for Microsoft SQL Server instance that is on your UNIX host. The names of the instance subdirectories you create must exactly match the names of the instance subdirectories on the UNIX host.

c For *each* of the existing Replication Agent instances, copy its configuration (*.cfg*) file from the UNIX host into the instance subdirectory you created in the previous step. The instance configuration file on the UNIX host is located in the *$SYBASE/RAX-nn_n/instname* directory. Here, *RAX-nn_n* is the previous Replication Agent installation directory, and *instname* is the name of the instance.

3 Starting with step 3 in the procedure "Replication Agent 15.1 is installed on the same Windows host as the previous version, and the current version of Microsoft SQL Server is 2005" on page 106, follow the steps to complete the upgrade.

Upgrading a trigger-based Replication Agent (version 12.5, 12.6, or 15.0) when the primary Microsoft SQL Server is version 7 or 2000

Note Replication Agent 15.1 requires that the primary Microsoft SQL Server be version 2005 SP2.

This section describes how to simultaneously upgrade Replication Agent 12.5, 12.6, or 15.0 to 15.1 and upgrade Microsoft SQL Server 7 or 2000 to 2005 in the following specific situations:

- Replication Agent 15.1 is installed on the *same* Windows host as the previous version of Replication Agent
- Replication Agent 15.1 is installed on a *different* Windows host than the previous version of Replication Agent

- Replication Agent 15.1 is installed on a Windows host, but the previous version of Replication Agent is installed on a *UNIX* or *Linux* host

Note Replication Agent 15.1 must be installed on the same host on which the primary Microsoft SQL Server is running.

Replication Agent 15.1 is installed on the *same* Windows host as the previous version of Replication Agent, and the current version of Microsoft SQL Server is 7 or 2000

❖ **To install Replication Agent 15.1 on the same Windows host as the previous version**

1 For *each* existing Replication Agent for Microsoft SQL Server instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 Sybase recommends that you install Replication Agent 15.1 into the same SYBASE directory as the previous version of Replication Agent.

3 On the host on which the primary data server is running, verify that the Microsoft Filter Manager Library is the correct version as described in "The sybfilter driver" on page 8.

4 Create the 15.1 version of all valid existing Replication Agent instances.

Note This step creates new Replication Agent 15.1 instances for all valid existing instances of the previous version of Replication Agent, regardless of whether the existing instances are for Oracle, IBM DB2 Universal Database, or Microsoft SQL Server. To complete the upgrade for Oracle or UDB 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 Open a command window.

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

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

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

```
ra_admin -u src_directory
```

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

```
ra_admin -u d:\sybase\RAX-15_0
```

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

5 If necessary, set the RA_JAVA_DFLT_CHARSET environment variable in each of the Replication Agent 15.1 *RUN_instance* scripts 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 Replication Agent 15.1 *Administration Guide*.

6 To prevent loss of any replicated data, deny users (other than the previously existing Replication Agent pds_username users) any further access to the primary databases.

7 For *each* of the previously existing Replication Agent for Microsoft SQL Server instances, 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 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 Replication Agent instance by issuing the quiesce command

   f Shut down the Replication instance by issuing the shutdown command.

8 When all of the previously existing Replication Agent instances have been stopped, upgrade the Microsoft SQL Server installation to version 2005 SP2. See Microsoft documentation for instructions. Verify that your primary data server meets the requirements described in "Microsoft SQL Server requirements" on page 2.

9 In the primary Microsoft SQL Server, grant each previously existing pds_username user the additional required privileges. See "Replication Agent permissions" on page 7.

10 Download and install the Microsoft SQL Server JDBC driver, and set the CLASSPATH environment variable, as described in the Replication Agent 15.1 *Installation Guide*. If the CLASSPATH contains any other Microsoft SQL Server JDBC driver, be sure to remove it. Only the Microsoft SQL Server JDBC driver required by Replication Agent 15.1 should be in the CLASSPATH.

11 Use the Replication Agent 15.1 sybfilter driver to make the Microsoft SQL Server transaction log files readable by Replication Agent. For details on installing and using the sybfilter driver, see Appendix B, "Using the sybfilter driver." However, at this time, it is not necessary to stop and restart Microsoft SQL Server because it will be done in a later step in this procedure.

12 Determine the primary Microsoft SQL Server DAC port number:

   a Using a text editor, open the *ERRORLOG* file in the root 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," and you will find an entry similar to this:

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

   c Make note of the port number specified; it will be used in a later step in this procedure.

13 Stop the Microsoft SQL Server service:

   a In Control Panel | Administrative Tools | Services, find the service named SQL Server (*SERVER*). Here, *SERVER* is the name of your Microsoft SQL Server data server. For example,

```
SQL Server (TEAMSTER)
```

b Stop the service.

14 Restart Microsoft SQL Server in single-user mode by opening a new command window and executing this command:

```
"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.

15 Start and log in to *each* of the Replication Agent for Microsoft SQL Server 15.1 instances and do the following:

a Verify that Microsoft SQL Server has been configured to allow a remote DAC connection.

b Set the pds_dac_port_number configuration parameter:

```
ra_config pds_dac_port_number, port
```

Here, *port* is the DAC port number you found in step 12.

c Set the rs_charset configuration parameter to match the Replication Server character set, as described in the Replication Agent 15.1 *Reference Manual.*

d Use the test_connection command to ensure that Replication Agent can connect to both Microsoft SQL Server and Replication Server.

e Initialize the Replication Agent instance and migrate the Replication Agent instance's metadata by issuing the ra_migrate command.

When this command executes in the first Replication Agent 15.1 instance, it will also initialize the Microsoft SQL Server. In subsequent Replication Agent 15.1 instances, it will only initialize the instance and migrate the instance's metadata.

16 Stop the Microsoft SQL Server in single-user mode:

a 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, password, and Microsoft SQL Server name.

b Issue the shutdown command.

17 Restart Microsoft SQL Server in multi-user mode (normal start):

a In Control Panel | Administrative Tools | Services, find the service named SQL Server (*SERVER*). Here, *SERVER* is the name of your Microsoft SQL Server data server. For example,

```
SQL Server (TEAMSTER)
```

b Stop the service.

18 Log in to *each* of the Replication Agent Microsoft SQL Server 15.1 instances and resume replication:

a Use isql to log in to the Replication Agent instance:

```
isql -Uusername -Ppassword -SinstanceName
```

Here, *username*, *password*, and *instanceName* are your user ID, password, and Replication Agent instance name.

b Issue the resume command.

19 Allow all users to access the primary databases.

Replication Agent 15.1 is installed on a *different* Windows host than the previous version, and the current version of Microsoft SQL Server is 7 or 2000

❖ **To upgrade when Replication Agent 15.1 is on a *different* host than a previous version, and the current version of Microsoft SQL Server is 7 or 2000**

1 For *each* existing Replication Agent for Microsoft SQL Server instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 Make the installation directory of the previous version of Replication Agent available as the source directory for the upgrade procedure:

a On the Windows host on which the previous version of Replication Agent is running, share the drive on which the Replication Agent is installed.

b On the Windows host on which Replication Agent 15.1 is installed, map a network drive to the host and drive in the previous step. On this mapped network drive, use the installation directory of the previous version of Replication Agent as the source directory for the upgrade.

3 Starting with step 3 in the procedure called “Replication Agent 15.1 is installed on the same Windows host as the previous version of Replication Agent, and the current version of Microsoft SQL Server is 7 or 2000” on page 113, follow the steps to complete the upgrade.

Replication Agent 15.1 is installed on a Windows host but the previous version of Replication Agent is installed on a UNIX or Linux host, and the current version of Microsoft SQL Server is 7 or 2000

❖ **To upgrade when Replication Agent 15.1 is on Windows, but the previous version is on UNIX or Linux**

1 For *each* existing Replication Agent for Microsoft SQL Server instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 On the Windows host, create a substitute installation directory and instance subdirectories for the existing instances in the previous version of Replication Agent that is on the UNIX host:

a On the Windows host, in the *SYBASE* directory in which you installed Replication Agent 15.1, create a directory with the name of the previous version of Replication Agent that you are upgrading from:

- For Replication Agent 12.5, name the directory *rax-12_5*.
- For Replication Agent 12.6, name the directory *RAX-12_6*.
- For Replication Agent 15.0, name the directory *RAX-15_0*.

You will use this newly-created directory as the source directory for the upgrade.

b In the directory you just created, create a subdirectory for *each* existing Replication Agent for Microsoft SQL Server instance that is on your UNIX host. The names of the instance subdirectories you create must exactly match the names of the instance subdirectories on the UNIX host

c For *each* of the existing Replication Agent instances, copy its configuration (*.cfg)* file from the UNIX host into the instance subdirectory you created in the previous step. The instance configuration file on the UNIX host is located in the *$SYBASE/rax-12_5/instname* or the *$SYBASE/RAX-nn_n/instname* directory. Here, *RAX-nn_n* is the previous Replication Agent installation directory, and *instname* is the name of the instance.

3 Starting with step 3 of the procedure called “Replication Agent 15.1 is installed on the same Windows host as the previous version of Replication Agent, and the current version of Microsoft SQL Server is 7 or 2000” on page 113, follow the steps to complete the upgrade.

Upgrading Replication Agent for Oracle

Because Replication Agent for Oracle 15.1 is log-based, it must be installed on the same host on which the primary Oracle server is running to directly access the Oracle transaction log files.

Using any of the upgrade procedures described in this section, the new 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 Replication Agent for Oracle 15.1 does not support downgrading 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 Replication Agent (version 12.6 or 15.0)
- Migrating Replication Agent 15.1 when upgrading Oracle 9i to 10g
- Upgrading a trigger-based Replication Agent 12.5 when the primary Oracle is version 9i or 10g
- Downgrading a log-based Replication Agent 15.1 to a trigger-based Replication Agent 12.5

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

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

Note 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 Replication Agent for Oracle instance, Sybase recommends that you first back up the Replication Agent System Database (RASD), as described in the Replication Agent 15.1 *Administration Guide*. Then, back up the complete existing Replication Agent instance directory.

2 Install the Replication Agent 15.1 software as described in "Installing the Replication Agent software" in the Replication Agent 15.1 *Installation Guide*. Sybase recommends that you install Replication Agent 15.1 into the same *SYBASE* directory as the previous version of Replication Agent.

3 Download and install the Oracle JDBC driver, and set the CLASSPATH environment variable, as described in the 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 Replication Agent 15.1 should be in the CLASSPATH.

4 Create the 15.1 version of all valid existing Replication Agent instances.

Note This step creates new Replication Agent 15.1 instances for all valid existing instances of the previous version of Replication Agent, regardless of whether the existing instances are for Oracle, IBM DB2 Universal Database, or Microsoft SQL Server. To complete the upgrade for Microsoft SQL Server or UDB 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 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 Replication Agent 15.1 *bin* directory:

- On UNIX:

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

- On Windows:

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

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

```
ra_admin -u src_directory
```

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

```
ra_admin -u d:\sybase\RAX-15_0
```

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

5 If necessary, set the RA_JAVA_DFLT_CHARSET environment variable in the 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 Replication Agent 15.1 *Administration Guide*.

6 In the primary Oracle database, grant the previously existing pds_username user any additional required privileges. See “Replication Agent permissions” on page 7.

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

8 Log in to the previously existing 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 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 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 Replication Agent 15.1 instance and migrate the 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 Replication Agent instance values for rs_source_ds, and *source_db* matches the previous 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 Replication Agent and do not need to be changed.

12 In the 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 Replication Agent 15.1 when upgrading Oracle 9i to 10g

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

Note Verify that the 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 Replication Agent moves the truncation point to the end of the log during 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 re-start the Replication Agent and issue the ra_migrate command.

3 As with the log-based Replication Agent upgrade, you may have to re-configure the 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 Replication Agent and upgrading Oracle 9i to Oracle 10g at the same time, you need to migrate Replication Agent 15.1 only once.

Upgrading a trigger-based Replication Agent 12.5 when the primary Oracle is version 9i or 10g

❖ **To upgrade a trigger-based Replication Agent 12.5 when the primary Oracle is 9i or 10g**

1 Back up the existing Replication Agent instance directory that contains the following configuration file:
$SYBASE/rax-12_5/<instance>/<instance>.cfg.

2 Install the Replication Agent 15.1 software on a machine where it can directly read the Oracle redo logs.

3 Create a Replication Agent 15.1 instance with a different name and port number than the Replication Agent 12.5 instance. The *port* and *port+1* must be unique on the machine.

   Do *not* start the instance.

4 Update the appropriate interfaces file (*interfaces* for UNIX, *sql.ini* for Windows) with the new instance name and new port number so that the migration script can isql in to the new Replication Agent 15.1 instance.

5 Update the Replication Agent 15.1 instance configuration file by running the generation script at *$SYBASE/RAX-15_1/bin/gen_RAO_migrate_with_parms.ksh*:

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

./gen_RAO_migrate_with_parms.ksh mySrcRao myuid
mypwd /workdir /path/mySrcRao.cfg
../myTgtRao/myTgtRao.cfg
```

   - *mySrcRao* is the name of the *interfaces* or *sql.ini* file entry for the Replication Agent 12.5.
   - *myuid* is the user ID for logging in to the Replication Agent 12.5 instance.
   - *mypwd* is the password for logging in to the Replication Agent 12.5 instance. If there is no password, use two double-quotes with nothing in between ("").
   - */workdir* is the path name of a directory to use as a work area and where the *<src_instance>_migrate_<date>.cmds* migration file will be created.
   - */path/mySrcRao.cfg* is the full path name of the Replication Agent 12.5 instance configuration file.
   - *../myTgtRao/myTgtRao.cfg* is the path name of the Replication Agent 15.1 instance configuration file that was created in step 3.

   The generation script copies to the Replication Agent 15.1 configuration file or creates parameter initialization commands in the migration file for most of the parameters in the Replication Agent 12.5 configuration file. The generated migration script is a file called */<workdir>/<src_instance>_migrate_<date>.cmds*. It contains Replication Agent commands that you will later run against the Replication Agent 15.1 instance to perform the following tasks:

   - Initialize the primary database.

- Initialize the Replication Agent 15.1 instance (including incrementing the database generation ID).
- Re-mark all the tables, procedures, and LOB columns that were marked in the Replication Agent 12.5 instance.

If the Replication Agent 12.5 and the Replication Agent 15.1 instances are on different machines and both configuration files cannot be accessed at the same time, copy the Replication Agent 12.5 configuration file to a location on the Replication Agent 15.1 instance's machine where it can be read by the generation script.

Note When the Korn shell script is running on Windows and the following message appears, you can ignore it:

```
tail: write error on standard output: The pipe is
being closed.
```

If the Replication Agent 15.1 instance is on a Windows machine that does not have Korn shell available, copy the generation script and the Replication Agent 15.1 instance's configuration file to a UNIX machine from which you can log in to the Replication Agent 12.5 instance. This copy of the Replication Agent 15.1 instance's configuration file is updated by the generation script. After it is updated, copy the configuration file back to the Replication Agent 15.1 instance directory.

Note After the migration script is generated, do *not* mark, unmark, enable, or disable any of the tables, LOB columns, or procedures. Also, do not modify any parameters in the Replication Agent 12.5 instance. If you do, these changes will not be applied to the Replication Agent 15.1 instance.

6 To see what objects will be marked and what LOB columns enabled, examine this generated file: */<workdir>/<src_instance>_migrate_<date>.cmds*.

If you want to change what is marked or enabled, you can make changes to this file. For example, you can set pdb_convert_datetime to true for some tables and procedures and to false for others.

7 Install the Oracle 10.2 JDBC driver for JDK 1.4 and 1.5 on the same machine where you installed Replication Agent 15.1, and add the JDBC driver's path to your CLASSPATH environment variable on this machine, as described in the Replication Agent 15.1 *Installation Guide.*

Note No other Oracle drivers are allowed in the CLASSPATH.

8 Set the RA_JAVA_DFLT_CHARSET environment variable in the *RUN_instance* script to the name of the Java character set that is equivalent to the one being used at the primary database. See the Replication Agent *Administration Guide* for information on setting RA_JAVA_DFLT_CHARSET.

9 Start and log in to the Replication Agent 15.1 instance.

10 Set rs_charset to match the Replication Server character set, as described in the Replication Agent 15.1 *Reference Manual*.

11 Test the primary database connection:

```
test_connection PDS
```

12 At the primary Oracle database, grant the Replication Agent 15.1 primary Oracle user (the user specified by the pds_username configuration parameter) the additional required privileges. See "Replication Agent permissions" on page 34.

13 Prevent users (other than the Replication Agent 15.1 pds_username user) from any further access to the primary database.

14 In the Replication Agent12.5 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 they are all zero, note the Last QID Sent from the last set of statistics.

c Issue the ra_locator update command so that Replication Agent 12.5 retrieves the truncation point from Replication Server.

d Wait, and then issue the ra_locator command and compare the displayed locator with that of the Last QID Sent. If they are different, wait and repeat this step.

15 Quiesce the Replication Agent 12.5 instance.

16 In the Replication Agent 12.5 instance, remove the Replication Agent transaction log:

```
pdb_xlog remove, force
```

17 Shut down the Replication Agent 12.5 instance.

18 In the primary Oracle database:

a Enable supplemental logging of primary key data.

b Enable archiving of redo logs.

c Archive all non-archived redo logs, and force Oracle to start writing to a clean log. For example:

```
alter system archive log current;
```

See "Replication Agent permissions" on page 34 for details.

19 Run the migration script that was generated in step 5 against the Replication Agent 15.1 instance:

```
isql –S <myTgtRAO> -Usa -P -i
/<workdir>/<mySrcRao>_migrate_<date>.cmds
```

This script initializes the primary database, initializes Replication Agent 15.1 (including incrementing the database generation ID), and re-marks all the tables, procedures, and LOB columns that were marked in the Replication Agent 12.5 instance.

20 Allow users access to the primary database.

21 Log in to the RSSD and set the Replication Server's locator to zero:

```
rs_zeroltm source_ds, source_db
```

Here, *source_ds* matches the Replication Agent 15.1 instance value for rs_source_ds, and *source_db* matches the Replication Agent 15.1 instance value for rs_source_db.

Note The *rs_source_ds* and *rs_source_db* values were migrated from Replication Agent 12.5 and should *not* be changed.

22 In the Replication Agent 15.1 instance, resume replication.

23 Sybase recommends that you change the administration user ID and password in the Replication Agent 15.1 instance from the default values to the same values you used in the Replication Agent 12.5 instance.

24 Log out of the Replication Agent 15.1 instance.

25 Update the *interfaces* or *sql.ini* file entries if you want the Replication Agent 12. 5 instance name associated with the Replication Agent 15.1 instance machine and port number.

Downgrading a log-based Replication Agent 15.1 to a trigger-based Replication Agent 12.5

This procedure assumes that you are using a Replication Agent 15.1 instance. If the Replication Agent 12.5 instance no longer exists, create one using the Replication Agent 12.5 ra_admin or administrator command, and then follow the procedure in this section, using the *$SYBASE/RAX-15_1/bin/gen_RAO_migrate_with_parms.ksh* script, *instead of* the *$SYBASE/RAX-15_1/bin/gen_RAO_migrate.ksh* script described next.

The difference between the two generation scripts is that the *gen_RAO_migrate_with_parms.ksh* script copies parameter values in addition to initializing the primary database and the Replication Agent, whereas the *gen_RAO_migrate.ksh* script assumes all parameters are already configured.

Note If you modified the *interfaces* or *sql.ini* file entries during your upgrade, you need to create a new entry (using a different name) to access the Replication Agent 12.5 instance.

❖ **To downgrade from Replication Agent 15.1 to version 12.5**

1 To generate the downgrade script, run the *$SYBASE/RAX-15_1/bin/gen_RAO_migrate.ksh* file:

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

./gen_RAO_migrate.ksh mySrcRao myuid mypwd /workdir
```

- *mySrcRao* is the name of the *interfaces* or *sql.ini* file entry for the Replication Agent 15.1 instance.
- *myuid* is the user ID for logging in to the Replication Agent 15.1 instance.

- *mypwd* is the password for logging in to the Replication Agent 15.1 instance. If no password exists, use two double-quotes with nothing in between ("").
- *workdir* is the path name of a directory to use as a work area and where the *<src_instance>_migrate_<date>.cmds* file will be created.

The script generates a file called */<workdir>/<src_instance>_migrate_<date>.cmds*, which contains Replication Agent commands that you will later run against the Replication Agent 12.5 instance to perform the following tasks:

- Initialize the primary database.
- Initialize the Replication Agent 12.5 instance (including incrementing the database generation ID).
- Re-mark all the tables, procedures, and LOB columns that were marked in the Replication Agent 15.1 instance.

The script does not modify any of the Replication Agent 12.5 parameters except pdb_auto_run_scripts, pdb_dflt_column_repl, and pdb_convert_datetime, which will all be set to the same values as configured in the Replication Agent 15.1 instance.

Note If the following message appears when running the Korn shell script on Windows, you can ignore it:

```
tail: write error on standard output: The pipe is
being closed.
```

After the migration script is generated, do *not* mark, unmark, enable, or disable any of the tables, LOB columns, or procedures. Also, do not modify any parameters in the Replication Agent 15.1 instance. If you do, these changes will not be applied to the Replication Agent 12.5 instance.

2 To see which objects will be marked and which LOB columns will be enabled, examine the following generated file: */<workdir>/<src_instance>_migrate_<date>.cmds*

If you want to change what is marked or enabled, you can make changes to this file. For example, you can set pdb_convert_datetime to true for some tables and procedures and to false for others.

3 Start and log in to the Replication Agent 15.1 instance.

4 Test the primary database connection:

```
test_connection PDS
```

5 Prevent users (other than the Replication Agent pds_username 15.1 user) from any further access to the primary database.

6 Verify that the Replication Agent 15.1 instance is in *Replicating* state and allow replication to finish. To verify that replication has completed:

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

      • Input queue size

      • Output queue size

   b When they are both zero, make note of the Last QID Sent from the last set of statistics.

   c Issue the ra_locator update command so that Replication Agent retrieves the truncation point from Replication Server.

   d Wait, and then issue the ra_locator command and compare the displayed locator with that of the Last QID Sent. If they are different, wait and repeat this step.

7 Quiesce the Replication Agent 15.1 instance.

8 In the Replication Agent 15.1 instance, de-initialize the Replication Agent:

```
pdb_xlog remove, force
```

9 Shut down the Replication Agent 15.1 instance.

10 Be sure that the appropriate Oracle JDBC driver is in your CLASSPATH. The one required for Replication Agent 15.1 will not work with older versions of Replication Agent for Oracle.

Note No other Oracle drivers are allowed in the CLASSPATH.

11 Run the script that was generated in Step 1 above against the Replication Agent 12.5 instance, for example:

```
isql –S <myTgtRAO> -Umyuid -Pmypwd -i
/<workdir>/<mySrcRao>_migrate_<date>.cmds
```

   • *myTgtRAO* is the name of the Replication Agent 12.5 instance.

- *workdir* is the path name of the directory that was used as a work area and where the *<src_instance>_migrate_<date>.cmds* file was created.
- *mySrcRao* is the name of the *interfaces* or *sql.ini* file for the path to the Replication Agent 15.1 instance.

The script initializes the primary database and the Replication Agent (including incrementing the database generation ID), and re-marks all the tables, procedures, and LOB columns that were marked in the Replication Agent 15.1 instance.

12 Allow users access to the primary database.

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

```
rs_zeroltm source_ds, source_db
```

Here, *source_ds* and *source_db* match the Replication Agent 12.5 instance values for the rs_source_ds and rs_source_db parameters.

Note The rs_source_ds and rs_source_db values were migrated from Replication Agent 15.1 and should *not* be changed.

14 In the Replication Agent 12.5 instance, resume replication.

15 Log out of the Replication Agent 12.5 instance.

16 If you created a new *interfaces* or *sql.ini* entry when you upgraded to Replication Agent 15.1, update the entry so the Replication Agent 15.0 instance name is again associated with the old Replication Agent 12.5 instance machine and port number.

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

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

Upgrading Replication Agent for UDB

Replication Agent for UDB 15.1 provides automatic upgrade of Replication Agent for UDB 15.0 instance, and automatic migration of Replication Agent for UDB 15.1 instance when you upgrade IBM DB2 Universal Database version 8 to version 9.

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

Replication Agent for UDB 15.1 does not support:

- Upgrading Replication Agent for UDB version 12.6 or earlier to version 15.1.
- Downgrading Replication Agent for UDB 15.1 to any previous version.
- Migrating Replication Agent for UDB 15.1 when IBM DB2 Universal Database is upgraded from version 6 or 7 to version 8 or 9.

The following sections include these topics:

- Upgrading from Replication Agent for UDB 15.0
- Migrating Replication Agent 15.1 when upgrading UDB 8 to 9

Upgrading from Replication Agent for UDB 15.0

This section describes how to upgrade Replication Agent for UDB 15.0 (which supports only UDB 8) to 15.1.

❖ **To upgrade from Replication Agent for UDB 15.0**

1 For *each* existing Replication Agent for UDB instance, Sybase recommends that you back up the complete existing Replication Agent instance directory.

2 Install the Replication Agent 15.1 software as described in "Installing the Replication Agent software" in the Replication Agent 15.1 *Installation Guide.* Sybase recommends that you install Replication Agent 15.1 into the same *SYBASE* directory as the previous version of Replication Agent.

3 Create the 15.1 version of all valid existing Replication Agent instances.

Note This step creates new Replication Agent 15.1 instances for all valid existing instances of the previous version of Replication Agent, regardless of whether the existing instances are for Oracle, IBM DB2 Universal Database, or Microsoft SQL Server. To complete the upgrade for Microsoft SQL Server or Oracle instances, please 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 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 Replication Agent 15.1 *bin* directory:

- On UNIX:

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

- On Windows:

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

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

```
ra_admin -u src_directory
```

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

```
ra_admin -u d:\sybase\RAX-15_0
```

4 If necessary, set the RA_JAVA_DFLT_CHARSET environment variable in the 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 Replication Agent 15.1 *Administration Guide*.

5 In the primary UDB, grant the previously existing pds_username user any additional required privileges.

6 Prevent users other than the previously existing Replication Agent pds_username user from any further access to the primary UDB database. This prevents any loss of replicated data.

7 Log in to the previously existing 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 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 Replication Agent instance by issuing the quiesce command.

   f Shut down the Replication instance by issuing the shutdown command.

8 Start and log in to the Replication Agent 15.1 instance, and migrate the Replication Agent's metadata by issuing the ra_migrate command.

9 In the Replication Agent 15.1 instance, resume replication by issuing the resume command.

10 Allow all users to access the primary UDB database.

Migrating Replication Agent 15.1 when upgrading UDB 8 to 9

This section describes how to migrate Replication Agent for UDB 15.1 when you upgrade IBM DB2 Universal Database from version 8 to version 9.

❖ **To migrate Replication Agent when upgrading UDB 8 to 9**

1 To prevent any loss of replicated data, deny further access to users (other than the previously existing Replication Agent pds_username user) to the primary UDB database.

2 Log in to the Replication Agent 15.1 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 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 Replication Agent instance by issuing the quiesce command.

   f Shut down the Replication instance by issuing the shutdown command.

3 Follow the steps in the IBM DB2 Universal Database documentation for upgrading UDB version 8 to version 9.

4 Verify that all the primary database requirements are met as described in “IBM DB2 Universal Database Requirements” on page 81.

5 Start the Replication Agent instance, and set the use_rssd configuration parameter to true:

```
ra_config use_rssd, true
```

   Replication Agent for UDB uses this configuration to connect to the RSSD and to reset the locator to zero.

6 Migrate the Replication Agent metadata by issuing the ra_migrate command.

7 After the migration, reset the use_rssd configuration parameter to false:

```
ra_config use_rssd, false
```

8 In the Replication Agent 15.1 instance, resume replication by issuing the resume command.

9 Allow all users to access the primary UDB database.

Note If you are upgrading Replication Agent and upgrading UDB 8 to UDB 9 at the same time, you need to migrate Replication Agent 15.1 only once.

APPENDIX B

Using the sybfilter driver

This appendix describes how to install, configure, use, and troubleshoot the sybfilter driver.

Topic	Page
Overview	137
Requirements	137
Installation and setup	138
Troubleshooting	140
Sybfilter command reference	140

Overview

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 Replication Agent. Before 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%\RAX-15_1\system\<platform>*.

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

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 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%\RAX-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%\RAX-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.

- *User manager* – Use the add command in the management console. The syntax for this command is as follows:

  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 **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**.

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

replication definition A description of a table or stored procedure in a primary database, for which subscriptions can be created. The replication definition, maintained by Replication Server, includes information about the columns to be replicated and the location of the primary table or stored procedure. See also **Replication Server** and **subscription**.

Replication Server The Sybase software product that provides the infrastructure for a robust transaction replication system. See also **Replication Agent**.

RSSD An abbreviation for *Replication Server System Database*, 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 *replicate database*. Contrast with **primary database**. See also **standby data**.

standby site — The location or facility at which standby data servers and standby databases are deployed to support disaster recovery, and normal business operations during scheduled downtime at the primary site. Sometimes called the *alternate site* or *replicate site*. Contrast with **primary site**. See also **standby database**.

subscription — A request for Replication Server to maintain a replicated copy of a table, or a set of rows from a table, in a standby database at a specified location. See also **replication definition** and **Replication Server**.

table — In a relational DBMS, a two-dimensional array of data or a named data object that contains a specific number of unordered rows composed of a group of columns that are specific for the table. See also **database**.

transaction — A unit of work in a database that can include zero, one, or many operations (including insert, update, and delete operations), and that is either applied or rejected as a whole. Each SQL statement that modifies data can be treated as a separate transaction, if the database is so configured. See also **SQL**.

transaction log — Generally, the log of transactions that affect the data managed by a data server. Replication Agent reads the transaction log to identify and acquire the transactions to be replicated from the primary database. See also **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

B

base objects, transaction log 93, 94

C

character case of database object names
  in IBM DB2 Universal Database 89
  in Microsoft SQL Server 14
  in Oracle 41–42
CLASSPATH environment variable 34
commands
  pdb_setrepproc 57
  pdb_setrepseq 62
communications
  JDBC driver 34
  Replication Agent protocols 7
configuration parameters
  ltl_character_case 14, 41–42, 89
  pdb_dflt_object_repl 58
  pdb_xlog_prefix 68, 94
creating
  transaction log 80

D

database objects
  transaction log object names 20–22, 68–69, 94–96
  transaction log prefix 68, 94
datatypes
  IBM DB2 Universal Database 90
  Microsoft SQL Server 15–17
  Oracle 45–50
disabling sequence replication 61–62

E

enabling stored procedure replication 60–61

F

files
  Replication Agent scripts directory 25, 71, 98

I

IBM DB2 Universal Database
  Administration Client 83
  character case 89
  communication error (-30081) 88
  DATA CAPTURE table attribute 81
  datatypes 90
  FORCE APPLICATION command 88
  marked objects table 96
  marking primary tables 80, 87–88
  origin queue ID 89–90
  primary database 79–104
  Replication Agent test setup scripts 98–104
  Replication Agent user ID 80
  requirements 81
  transaction log positioning 86

J

java stored procedures 95
JDBC driver
  Oracle 34

L

Log Reader component

asynchronous operation 87
positioning in transaction log 86
read buffer size 88
log-based Replication Agent
table marking 80, 87–88
ltl_character_case configuration parameter 14, 41–42, 89
LTM locator 86–88
origin queue ID 14, 42, 89–90

M

marked objects table
IBM DB2 Universal Database 96
marker shadow tables 21, 69
marking a primary table
in IBM DB2 Universal Database 80, 87–88
marking a sequence 55–59
Microsoft SQL Server
character case 14
datatypes 15–17
isql tool 13–14
origin queue ID 14
permissions 7
primary database 1–32
Replication Agent user ID 7
Microsoft Windows platforms 2

N

names
transaction log objects 68–69, 94–96

O

operating system
Microsoft Windows platforms 2
Oracle database server
character case 41–42
datatypes 45–50
JDBC driver 34
origin queue ID 42
primary database 33–77
TNS Listener Service 34
origin queue ID
IBM DB2 Universal Database 89–90
Microsoft SQL Server 14
Oracle 42

P

pdb_dflt_object_repl configuration parameter 58
pdb_setrepproc command 57
pdb_setrepseq command 62
pdb_xlog_prefix configuration parameter 68, 94
prefix, transaction log 68, 94
primary databases
IBM DB2 Universal Database 79–104
Microsoft SQL Server 1–32
Oracle 33–77
Oracle database server 33–77
Replication Agent user ID 7, 80
primary tables
marking in IBM DB2 Universal Database 80, 87–88

R

Replication Agent
communications 7
Log Reader component 87
LTM locator 86–88
marked objects table 96
origin queue ID 14, 42, 89–90
primary database user ID 7, 80
scripts directory 25, 71, 98
test scripts 25–32, 71–77, 98–104
transaction log 19, 67, 93
transaction log prefix 68, 94
Replication Agent for Microsoft SQL Server 1–32
datatype compatibility 15
permissions 7
primary database user ID 7
transaction log 19
Replication Agent for Oracle 33–77
datatype compatibility 45–50
JDBC driver 34

Running Oracle Server and Replication Agent on different machines 62
transaction log 67
Replication Agent for UDB 79–104
configuration parameters 85
creating transaction log 80
database communication error (-30081) 88
datatype compatibility 90
marked objects table 96
primary database user ID 80
scan buffer size 88
setup test scripts 98–104
transaction log 93
Replication Server
LTM locator 86–88
RSSD
changes to support Oracle datatypes 43

S

scripts
directory 25, 71, 98
Replication Agent test setup 25–32, 71–77, 98–104
sequence
disabling replication 61–62
marking 59
unmarking 59
sequences 69
marking 55
shadow tables
marker 21, 69
stored procedures
enabling replication 60–61

T

test scripts for Replication Agent setup 25–32, 71–77, 98–104
TNS Listener Service, Oracle 34
transaction logs
base objects 94
creating 80
Log Reader positioning in 86
marked objects table 96
object names 68–69, 94–96
prefix 68, 94
Replication Agent for Microsoft SQL Server 19, 23
Replication Agent for Oracle 67
Replication Agent for UDB 93
shadow tables 21, 69
truncating 22–23, 70
truncation
procedures 95

U

unmarking a sequence 59
user IDs
primary database 7, 80

W

Windows
See Microsoft Windows platforms