SYBASE®

Primary Database Guide

Replication Agent™

15.5

Linux, Microsoft Windows, and UNIX

DOCUMENT ID: DC00269-01-1550-01

LAST REVISED: May 2010

Contents

About This Book

Replication Agent™ 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:

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

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

Chapter 2, “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 3, “Replication Agent for UDB,” describes replication system issues that are specific to IBM DB2 Universal Database (UDB), and details of the Replication Agent for UDB.

Appendix A, “Upgrading and Downgrading 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 Release Bulletin* contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletins 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.

- 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 the information you need to complete the software installation and Replication Agent setup.

- 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.

Java environment

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

- The *Replication Agent Release Bulletin* contains the most up-to-date information about Java and JRE requirements.

- Java documentation available from your operating system vendor describes how to set up and manage 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:

- Oracle
- Microsoft SQL Server
- UDB

Adaptive Server Enterprise

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

You can find more information about Adaptive Server Enterprise on the Sybase Web site at http://www.sybase.com/support/manuals/.

Other sources of information

Use the Sybase Getting Started 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. 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 Sybase Product Manuals Web site is an online version of the books 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.

❖ Finding the latest information on product certifications

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

❖ Finding 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.

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

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

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

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. 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_5/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.
( )	Type parentheses 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 is interpreted as an invalid parameter name.
- Database object names are not case-sensitive in Replication Agent commands. However, to use a mixed-case object name in a command (to match a mixed-case object name in the database), 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 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

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

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

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

For a Section 508 compliance statement for Replication Agent 15.5, 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, have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Replication Agent for Oracle

The term "Replication Agent for Oracle" refers to an instance of Replication Agent software that is 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.

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

Oracle-specific considerations

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

- Unsupported software features
- Unsupported datatypes, data, and structures
- Replication Agent connectivity
- Replication Agent permissions
- Redo and archive log setup
- Supplemental logging
- DDL Replication
- 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 Server set autocorrection command
- Partitioned tables
- Materialized views
- Index-organized tables
- Controlling trigger execution at the replicate database
- Altering replication definitions from the primary data server
- Oracle Data Guard
- Database resynchronization
- Troubleshooting Oracle transactions and operations
- Replicating stored procedures with arguments of boolean type
- Oracle warm standby
- Oracle Flashback
- Replicating XMLTYPE data
- Oracle 9i

Unsupported software features

The following features are not supported:

- Oracle encrypted and virtual columns

- Oracle encrypted partitions
- Oracle label security
- Oracle packaged stored procedures and functions (standalone procedures and functions are supported)
- Oracle schema objects in encrypted tablespaces
- Replication Server parallel DSI
- Replication Server rs_init utility
- Replication Server rs_subcomp utility
- Replication Server automatic materialization
- Replication Server when replicating in an environment where other vendors are replicating

Oracle 11g support

While most Oracle 11g features are supported as of Replication Agent 15.1 ESD #3, the following features are not supported:

- SecureFile – this feature is a redesign of the implementation of large object (LOB) storage in Oracle 11g. You can mark tables containing these types of columns for replication, but the columns are not replicated.
- Virtual columns – Replication Agent supports the replication of tables containing computed (or virtual) columns in Oracle 11g. However, the replication of individual computed columns is not supported. You can mark tables with virtual columns for replication using the force option, but the virtual columns will not be replicated.
- Encrypted tablespaces – You cannot mark tables or columns belonging to an encrypted tablespace for replication. If the tablespace that a marked table or marked column belongs to is subsequently updated so that the tablespace becomes encrypted, Replication Agent will automatically unmark the table or column.

  If the first table modified by a transaction is stored in an encrypted tablespace and subsequent table modifications for the same transaction occur against non-encrypted tables, Replication Agent neither recognizes nor replicates the subsequent non-encrypted operations.This is an issue only when the first operation is for an encrypted table. If the operation against the encrypted table is not the first, only the encrypted operation is ignored. All prior and subsequent non-encrypted operations are processed.

- Encrypted columns – you can mark tables containing encrypted columns for replication using the force option, but these columns are not replicated.
- XML – Oracle 11g XML types are not supported.

Unsupported datatypes, data, and structures

The following datatypes are not supported:

- Oracle-supplied datatypes:
    - "Any" Types (SYS.ANYTYPE, SYS.ANYDATASET), except for SYS.ANYDATA.
    - MLSLABEL
    - All Oracle 11g XML Types (XMLType, URIType, DBURIType, XDBURIType, HTTPURIType) as well as Oracle 10g XMLTYPE data stored as XML (not as CLOB).
    - Spatial Types (MDSYS.SDO_GEOMETRY, SDO_TOPO_GEOMETRY, SDO_GEORASTER)
    - Media Types (ORDSYS.ORDAudio, ORDSYS.ORDImage, ORDSYS.ORDImageSignature, ORDSYS.ORDVideo, ORDSYS.ORDDoc, SI_StillImage, SI_Color, SI_AverageColor, SI_ColorHistogram, SI_PositionalColor, SI_Texture, SI_FeatureList)
    - Datatypes used with Oracle Expression Filter
    - ANYDATA, if it is being replicated to a non-ANYDATA column or if the data exceeds 16KB, which is the size constraint of the Replication Server OPAQUE datatype.
    - BFILE, UROWID, or REF data stored in an ANYDATA column
    - BLOB or Object user-defined datatype data being replicated to a replicate database other than Oracle or ASE. When replicating to Oracle, use ExpressConnect for Oracle since data of these types is not supported by ECDA for Oracle.
    - REF
    - UROWID
- SecureFile LOB
- User-defined datatypes containing LOB data

- User-defined datatypes defined as NOT FINAL

Note Partial updates to Oracle LOBs are not supported.

Data stored in the Oracle XML DB Repository is not supported if it is accessed with the following protocols or methods:

- Internet protocols like FTP, HTTP, HTTPS, and WebDAV
- The Oracle XML DB Repository API

The following user-defined object types and structures are not supported:

- Associative arrays
- Nested tables
- VARRAY
- Nested tables or VARRAY stored in an ANYDATA column

Predefined PL/SQL numeric datatypes

Replication Agent does not support marking procedures containing the following predefined PL/SQL numeric datatypes:

- BINARY_DOUBLE
- BINARY_FLOAT
- NUMBER
- PLS_INTEGER or BINARY_INTEGER

Replication Agent does support marking procedures with the SIMPLE_INTEGER datatype, which is s subtype of PLS_INTEGER. However, Replication Agent does not support marking procedures containing any other subtypes of the aforementioned types, including NATURAL, NATURALN, PSOTIVE, POSITIVEN, and SIGNTYPE.

Replication Agent connectivity

Connectivity between the Replication Agent for Oracle and the Oracle data server is through the Oracle Java Database Connectivity (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*.

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.
- alter database – required for Replication Agent to read from a Data Guard standby database transaction log.
- alter on *TABLE_NAME* – required to replicate user-defined datatypes if table-level supplemental logging has not been enabled for the specified *TABLE_NAME*.
- execute on DBMS_FLASHBACK – required to execute DBMS_FLASHBACK.get_system_change_number.
- execute on SYS.DBMS_LOCK – required to generate commit log records at the primary database.
- select on SYS.RECYCLEBIN$ – required to use Oracle Flashback with Replication Agent.
- select on SYS.OPQTYPE$ – required for DDL replication and XMLTYPE data replication.
- alter any procedure – required to manage procedures for replication.
- create table – required to create tables in the primary database.
- create procedure – required to create rs_marker and rs_dump proc procedures.
- create public synonym – required to create synonyms for created tables in the primary database.
- create sequence – required to support replication.

- drop public synonym – required to drop created synonyms.
- select on SYS.ARGUMENT$ – required to process procedure DDL commands.
- SYS.ATTRIBUTE$ – required to process Oracle types.
- select on SYS.CCOL$ – required to support table replication (column constraint information).
- select on SYS.CDEF$ – required for table (constraint information) replication support.
- select on SYS.COL$ – required for table (column information) replication support.
- select on SYS.COLLECTION$ – required to support table replication.
- select on SYS.COLTYPE$ – required to support table replication.
- select on SYS.CON$ – required for table (constraint information) replication support.
- select on SYS.IND$ – required to identify indexes.
- select on SYS.INDCOMPART$ – required to identify indexes.
- select on SYS.INDPART$ – required to identify indexes.
- select on SYS.INDSUBPART$ – required to identify indexes.
- select on SYS.LOB$ – required for LOB replication support.
- select on SYS.LOBCOMPPART$ – required to support partitioned LOB replication.
- select on SYS.LOBFRAG$ – required to support partitioned LOB replication.
- select on SYS.MLOG$ – required to filter materialized view log tables.
- select on SYS.NTAB$ – required to support table replication.
- select on SYS.OBJ$ – required for processing procedure DDL commands in the repository.
- select on SYS.PROCEDUREINFO$ – required for procedure replication support.
- select on SYS.SEQ$ – required to support sequence replication.
- select on SYS.SNAP$ – required to filter out materialized view tables.

- select on SYS.TAB$ – required to support table replication.
- select on SYS.TABCOMPART$ – required to support partitioned table replication.
- select on SYS.TABPART$ – required to support partitioned table replication.
- select on SYS.TABSUBPART$ – required to support partitioned table replication.
- select on SYS.TS$ – required to identify tablespace encryption in Oracle 11g.
- select on SYS.TYPE$ – required to process Oracle predefined and user-defined types.
- select on SYS.USER$ – required for Oracle user identification.

Note The permissions for SYS.CON$ and SYS.CDEF$ are required to handle the constraint information in the CREATE and ALTER TABLE DDL operations.

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.

Replication Agent for Oracle requires the alter system privilege to issue the alter system archive log command. If Replication Agent is configured to access only online Oracle redo logs, Replication Agent issues the alter system archive log sequence command when the online redo log is no longer needed for replication (as when all data from the log has been replicated). Regardless of online or archive log processing, Replication Agent uses the alter system privilege to issue the alter system archive log current command when Replication Agent is instructed to move processing to the end of the Oracle log. By issuing the alter system archive log current command, Replication Agent insures that the current redo log file does not contain old data. Replication Agent moves processing to the end of the Oracle redo log when requested by the move_truncpt options of the pdb_xlog init command. Replication Agent may also move processing to the end of the Oracle redo log during migration from one version of Replication Agent to another.

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.

By default, you can access both online and archive logs. You can configure Replication Agent to access only the online logs, but doing so requires you to turn auto-archiving off and requires Replication Agent to issue manual archive log commands to Oracle.

Accessing archive logs

When you are using the default, for archive log files to be accessed, configure Replication Agent to use the directory path where archive log files are located. By default, an Oracle instance creates multiple directories under the flash recovery area specified by the DB_RECOVERY_FILE_DEST parameter of the Oracle ALTER SYSTEM command, each directory corresponding to and named after a separate day. However, Replication Agent requires archived redo log files to reside in a single directory. Consequently, you must configure Oracle to archive to a single directory to be read by Replication Agent.

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

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

Note This section discusses how to access 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 44.

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

- Redo log archiving must be enabled:

```
alter database ARCHIVELOG;
```

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

Verify that log archiving is enabled:

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

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

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

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

Accessing archive log files

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

The rman_enabled parameter enables Replication Agent to use the Oracle RMAN utility to truncate old archive log files. See the *Replication Agent Reference Manual*.

Setting archiving for Replication Agent

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

When the pdb_include_archives configuration parameter is set to false, Replication Agent for Oracle also requires you to disable automatic archiving of Oracle redo logs. Archiving is performed manually by Replication Agent as the data in the online 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

❖ Disabling 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 11g

❖ **Disabling automatic archiving**

1 To change the LOG_ARCHIVE_START parameter, either manually edit the server 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 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, enter the following:

```
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 If 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 enabled or if manual archiving is performed, causing a redo log file not yet processed by the Replication Agent to be overwritten, the data in the lost redo log file is not 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

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 enter the following:

```
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, supplemental logging of primary key information is enabled.

Table-level supplemental logging

To replicate updates to user-defined object type attributes, Replication Agent must enable table-level supplemental logging. Table-level supplemental logging can be enabled manually as follows:

```
ALTER TABLE THE_TABLE ADD SUPPLEMENTAL LOG DATA (ALL)
COLUMNS;
```

Here, *THE_TABLE* is the name of the table on which supplemental logging is being enabled. Verify that table-level supplemental logging has been enabled with the following command:

```
select count(*) from ALL_LOG_GROUPS where
LOG_GROUP_TYPE='ALL COLUMN LOGGING' and OWNER=THE_OWNER
```

```
    and TABLE_NAME=THE_TABLE
```

Here, *THE_OWNER* is the table owner. If this command returns a value of 1, table-level supplemental logging has been enabled for this table.

You can also enable supplemental logging from Replication Agent for Oracle using the ra_set_autocorrection command as described in the *Replication Agent Reference Manual*.

DDL Replication

Replication of data definition language (DDL) commands is supported only between Oracle databases. You cannot replicate DDL commands from Oracle to non-Oracle replicate databases.

Using the pdb_setrepddl command, Replication Agent for Oracle can disable or enable the replication of specific DDL commands by object, owner, statement, or user. See the *Replication Agent Reference Manual* for details on using the pdb_setrepddl command.

Setting *ddl_username* and *ddl_password*

To replicate DDL in Oracle, the pdb_setrepddl command must be used to set filtering rules accordingly. You must also set the Replication Agent ddl_username and ddl_password parameters. The ddl_username parameter is the database user name that should be used to execute the replicated DDL command at the 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*.

Special usage notes

The value of the ddl_username parameter cannot be the same as the value of the maintenance user defined in Replication Server for the replicate 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 replicate 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. See the *Replication Server Reference Manual*.

DDL commands and objects filtered from replication

The following DDL commands are not replicated:

alter database
alter system
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
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. 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*.

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 fails. 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 treats the character case of database object names sent 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 1-1 illustrates the format of the origin queue ID for the Replication Agent for Oracle.

Table 1-1: 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

The Replication Agent installation includes 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 and must be applied when the installed Replication Server is version 15.0.1 or earlier:

$SYBASE/RAX-15_5/scripts/oracle/hds_oracle_new_setup_for_replicate.sql
$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_1_rs.sql
$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_2_rssd.sql
$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_3_rs.sql

The following Replication Server scripts should be run manually against the RSSD when the installed Replication Server is version 15.0.1 or earlier:

$SYBASE/RAX-15_5/scripts/oracle/hds_oracle_funcstrings.sql
$SYBASE/RAX-15_5/hds_oracle_udds.sql
$SYBASE/RAX-15_5/hds_clt_ase_to_oracle.sql

❖ Applying the script changes for user-defined datatypes

1 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_5/scripts/oracle/hds_oracle_new_setup_replicate.sql

This script defines 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.

2 To correctly define the Oracle error class for Replication Server 15.0.1 or an earlier version:

- Apply the following script at Replication Server:

  $SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_1_rs.sql

- Apply the following against your RSSD:

  $SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_2_rssd.sql

- Apply the following script at Replication Server:

  $SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_3_rs.sql

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 1-2 describes the conversion of Oracle datatypes to Sybase datatypes.

Table 1-2: Recommended Oracle datatype mapping

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
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.

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.
BFILE	4GB, locator points to large binary file	image	2GB	
BLOB	4GB, variable-length binary large object	image	2GB	
BOOLEAN	1 byte	rs_oracle_decimal	17 bytes.	The BOOLEAN datatype is only for use with PL/SQL.
CHAR	255 bytes	char	32K	
CLOB	4GB, variable-length character large object	image or unitext	2GB	For Replication Server 15.0 and later versions, the CLOB datatype maps to unitext. For earlier versions of Replication Server, the NCLOB datatype maps to image.
DATE	8 bytes, fixed-length	datetime or rs_oracle_datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 9999 AD. If pdb_convert_datetime is true, the Sybase datatype used should be datetime. The value replicated is YYYYMMDD HH:MM:SS.sss. If pdb_convert_datetime is false, the Sybase datatype used should be rs_oracle_datetime. The format replicated is MM/DD/YYYY HH:MI:SS.
INTERVAL DAY(n) TO SECOND(n)	Variable-length	rs_oracle_interval		

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
INTERVAL YEAR(n) TO MONTH	Variable-length	rs_oracle_interval		
LONG	2GB, variable-length character data	text		
LONG RAW	2GB, variable-length binary data	image		
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.
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.

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
NVARCHAR2	2000 bytes, variable-length, multibyte character data	univarchar or varchar	32K	
RAW	2000 bytes, variable-length binary data	rs_oracle_binary	32K	
ROWID	6 bytes, binary data representing row addresses	rs_oracle_rowid	32K	
SIMPLE_INTEGER	4 bytes representing signed integers	integer		SIMPLE_INTEGER is new as of Oracle 11g and is only for use with PL/SQL. **Note** Marking procedures with PLS_INTEGER and predefined PL/SQL numeric datatypes other than SIMPLE_INTEGER is not supported.
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		
UDD object type	Variable length character data	rs_rs_char_raw	32K	See "Oracle user-defined types" on page 31.
VARCHAR2	2000 bytes, variable-length character data	varchar	32K	

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
XMLTYPE	4GB, variable-length character large object	text	2GB	XMLTYPE data is implicitly handled as Oracle CLOB data.

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 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

The Oracle ANYDATA datatype

Replication Agent supports the replication of data stored in ANYDATA columns under the following conditions:

- Both the primary and replicate databases are Oracle databases.
- Both the primary database table and replicate database tables have the same ANYDATA columns.

The pdb_ignore_unsupported_anydata configuration parameter is provided to determine how Replication Agent handles data of unsupported datatypes stored in columns of type ANYDATA. For information on how to use this parameter, see the *Replication Agent Reference Manual*.

The Oracle XMLTYPE datatype

Replication Agent supports the replication of data stored in XMLTYPE columns from an Oracle primary database to an Oracle or Adaptive Server Enterprise replicate database. Replication Agent also supports the replication of data stored in XMLTYPE tables from an Oracle primary database to an Oracle replicate database. Replication Agent only supports the replication of XMLTYPE columns and tables stored as CLOB data.

XML encoding

Replication Agent ignores the XML encoding statement in XMLTYPE data because CLOB data is always encoded in the Oracle database character set.

XMLTYPE in the Oracle transaction log

Even for a partial updates, Oracle logs the entire contents of the CLOB column in which XMLTYPE data is stored. Replication Agent also replicates the entire contents of the CLOB column.

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

  - The precision of the floating point representation has the same range limitation as the integer representation.
  - If the floating point value is outside the Sybase range of 2^{31} - 1 and -2^{31} (2,147,483,647 and -2,147,483,648), 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 Agent does not support replication of the following special values for BINARY_FLOAT and BINARY_DOUBLE datatypes:

- NaN (not a number)
- Inf (positive infinity)
- -Inf (negative infinity)

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 replicated value is "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 earlier than 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.

The Oracle ANYDATA datatype

Replication Agent does not support the replication of data stored in ANYDATA columns under the following circumstances:

- The replicate database table column is not of type ANYDATA. An attempt to replicate data stored in an ANYDATA column to a column that is not of the ANYDATA type will cause the Replication Server Data Server Interface (DSI) thread to fail.
- The size of the data stored in an ANYDATA column exceeds the maximum size of the Replication Server opaque datatype, which is 16K.
- Replication Agent does not replicate data of the following Oracle datatypes or structures stored in a column of type ANYDATA:
  - BFILE
  - Nested tables
  - REF
  - UROWID
  - VARRAY

The pdb_ignore_unsupported_anydata configuration parameter is provided to determines how Replication Agent handles data of unsupported datatypes stored in columns of type ANYDATA.

ANYDATA and character set conversion

Replication Server will not perform character set conversion for character data stored in an ANYDATA column if the setting of the Replication Agent rs_charset configuration parameter is different from the setting of the Replication Server RS_charset configuration parameter.

See also

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

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

Replication Agent Reference Manual for information on the pdb_ignore_unsupported_anydata configuration parameter.

The Oracle XMLTYPE datatype

Replication Agent supports the replication of XMLTYPE columns and tables only if they are stored as CLOB data. Replication Agent does not support the replication of XMLTYPE data in object-relational XML storage or binary XML storage.

Replication Agent supports the replication of XMLTYPE columns from an Oracle primary database to an Oracle or Adaptive Server Enterprise replicate database, but no other platforms are supported. Replication Agent replicates XMLTYPE tables only from an Oracle primary database to an Oracle replicate database.

Note The XMLTYPE datatype is supported for Oracle 10g only.

The Oracle ROWID datatype

When Replication Agent replicates ROWID data, the value replicated always represents the value stored in the table in the primary database and has no relationship to the ROWID value in the replicate database. There is no attempt to convert or adjust ROWID data to match the data in the replicate database.

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 “Compromising transaction integrity” on page 27, 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 replicate site.

Replication of LOB columns

Oracle logs all LOB data (except for *BFILE* datatypes) in the Oracle redo log. This allows the Replication Agent to apply each individual LOB change. However, for *BFILE* data, the same technique is used and the same limitation exists—*BFILE* data is not logged but read from the database at the time the rest of the transaction is processed. If two consecutive transactions modify the same *bfile*, the same inconsistency described in “Compromising transaction integrity” on page 27 can occur. For instructions on enabling and disabling replication for LOB columns, see the *Replication Agent Administration Guide*.

Compromising transaction integrity

Because of the way Replication Agent processes the LOB column data when replicating transactions, it is possible to compromise transaction integrity. For example, if two transactions change the data in a LOB column and the Log Reader does not process the first transaction until after the second transaction has been committed, when the LOB data is read from the primary database, the value of that data is the result of the second transaction. In this event, the value of the LOB data in the first transaction is never sent to the replicate database. After the second transaction is processed by the Log Reader, the primary and replicate databases are synchronized again, but for a period of time between processing the first and second transactions, the replicate database contains data that does not match the originating transaction.

This problem occurs only when a LOB column is changed more than once by a sequence of transactions. The period of time over which the problem exists could be significant if the replication system throughput is slow or if a replication system component fails. As soon as the last transaction that changes the LOB column is processed at the replicate site, the problem is corrected.

Limitations

Replication Agent does not support the replication of partial updates to LOB columns. For example, use of the Oracle DBMS_LOB.WRITE() function, which updates LOB data from a specified offset, is not replicated.

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, the column data that identifies the changed values and lookup 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 replicate 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
```

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

The following transaction sequences are not supported for LOB columns defined with the disable storage in row clause and result in a failure to supply the LOB data to the replicate 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';
```

Replicating CLOB and NCLOB datatypes

Oracle NCLOB (National Character Large Object) is a datatype that stores large character data using a multibyte national character set. Similarly, the CLOB datatype may also store character data using a multi-byte national character set, when the Oracle database is defined with a double-byte or variable-width character set.

By default, the byte order of the multi-byte characters stored in the NCLOB datatype (and CLOB when the database is defined with a double-byte or variable-width character set) is converted during replication to big-endian byte order. This allows the data to be transmitted over networks using big-endian order, which is the common network byte order.

The datatype in a replication definition for an NCLOB or CLOB should be unitext. This prevents Replication Server from attempting character set conversion on the data. If the Replication Server version does not support unitext (Replication Server version 12.6 and earlier) use the image datatype.

If the target database that is to receive this NCLOB or CLOB data is installed on a little-endian platform, the database may not automatically convert the replicated data from the sent big-endian order to the little-endian order. To support replicating NCLOB or CLOB data to a database server that does not provide the necessary conversion from big-endian (network order) to little-endian, force the byte order to be sent by the Replication Agent using the lr_ntext_byte_order parameter by specifying a value of big (for big-endian) or little (for little-endian).

The lr_ntext_byte_order parameter is available for Microsoft SQL Server and Oracle, and is important for replication between two databases that reside on different platforms. For example, for replication between Oracle and Microsoft SQL Server, the primary database stores the data in big-endian byte order, but the replicate database stores data in little-endian byte order because Microsoft SQL Server only runs on Windows. Therefore, set the lr_ntext_byte_order parameter to little to force the Replication Agent to convert the data to little-endian (the format expected by SQL Server). However, if the replicate database is not a Microsoft SQL Server, determine its byte order and set the lr_ntext_byte_order parameter accordingly.

When replication occurs between Oracle and Oracle using CLOB data (and the primary database is defined with a double-byte or variable-width character set), the CLOB data will contain multi-byte national character sets and is replicated. For ECDA for Oracle to correctly apply this data, set the ECDA configuration property rep_unitext to "1". This configuration in ECDA for Oracle instructs ECDA to tell Oracle the incoming CLOB data is in multi-byte format.

Note The default behavior of Replication Agent for Oracle is to force any Unicode data to big-endian order as defined by the ltl_big_endian_unitext configuration parameter. To allow the lr_ntext_byte_order configuration parameter to successfully override the Oracle byte order, you must also set the ltl_big_endian_unitext configuration parameter to false whenever the lr_ntext_byte_order parameter 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 the Replication Server. Valid values are true and false. When setting this parameter, you must know how lr_ntext_byte_order is set. If lr_ntext_byte_order 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. ltl_big_endian_unitext is true, by default. The ltl_big_endian_unitext and lr_ntext_byte_order configuration parameters have differences:

- When ltl_big_endian_unitext is true, Replication Agent for Oracle sends all Unicode data in big-endian order.
- When ltl_big_endian_unitext is false, Replication Agent for Oracle allows Unicode data to be sent in a byte order that is used when the data is stored in the transaction log file.

lr_ntext_byte_order forces the result of Unicode data that is read from the transaction log to be in the correct byte order, regardless of how it normally exists in the transaction log file.

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 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_char_raw. If you are using Replication Server 15.1 or earlier, see “Replication Server and RSSD scripts” on page 17 first.

❖ Creating 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:

```
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_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.

Replicating object type attributes

To replicate updates to user-defined object type attributes, Replication Agent must enable table-level supplemental logging. Table-level supplemental logging can be enabled manually. Replication Agent also attempts to enable this logging when marking a table that contains a user-defined object type. However, for Replication Agent to mark such a table, there must already be an Oracle user specified by the pds_username parameter that has ALTER permission granted for the table.

If table-level supplemental logging has not been enabled for a table containing a user-defined object type and Replication Agent encounters an update log record in the Oracle log, Replication Agent changes its status from Replicating to Admin with the following error:

```
There is insufficient column data in the log to support
Oracle UDD update command processing. Please make sure
table-level supplemental logging is enabled.
```

In this case, use the pdb_skip_op to skip this log record. See the *Replication Agent Reference Manual*.

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. See "Enabling and disabling replication for sequences" on page 39.

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.

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. Execute the *$SYBASE/RAX-15_5/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. Log into the replicate Oracle database as the maintenance user defined in your Replication Server connection to the replicate database. Execute the *$SYBASE/RAX-15_5/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**

1. Log in to the Replication Agent instance with the administrator login.
2. 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.

   - 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. 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 replicate database that you wish to increment.

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

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

4 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_5/scripts/oracle* directory at the replicate site to create a procedure named rs_update_sequence.

❖ Unmarking a sequence

1. Log in to the Replication Agent instance with the administrator login.
2. 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.

   - 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. 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. 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.

   To force the unmark, use the following command:

   ```
   pdb_setrepseq pdb_seq, unmark, force
   ```
5. 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

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

Note By default, no sequences are marked 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. See “Marking a sequence for replication” on page 36.

❖ Enabling replication for a marked sequence

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

2 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 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 is replicated.

4 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 a marked sequence**

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

2 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 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 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.

❖ **Setting 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 provides support for Oracle 10g and 11g 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.

Configure Replication Agent 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 44 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.

Note The rman_enabled parameter enables Replication Agent to use the Oracle RMAN utility to truncate old archive log files. See the *Replication Agent Reference Manual*.

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 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. For Oracle 10g, coarse-grained striping extents are 1MB in size. For Oracle 11g, coarse-grained striping extents can be 1, 2, 4, 8, 16, 32, or 64MB 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

When using 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
```

Archive logs stored in and managed by ASM are owned by the corresponding unique Oracle database name. If the Oracle database name differs from the global unique database name, the pdb_archive_path configuration parameter must be set to both the name of the ASM disk group and the globally unique name of the database in which the archive logs are stored:

```
pdb_archive_path=+DISK_GROUP1/database_name
```

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 be manually removed.

Note The rman_enabled parameter enables Replication Agent to use the Oracle RMAN utility to truncate old archive log files. See the *Replication Agent Reference Manual*.

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. For Oracle 10g or 11g, set asm_username as follows:

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

Alternately, for Oracle 11g, you can set asm_username as follows:

```
asm_username="sys as sysasm"
```

See the *Replication Agent Reference Manual*.

Replication Server *set autocorrection* command

The Replication Server set autocorrection command prevents failures that would otherwise be caused by missing or duplicate rows in a replicated table. The set autocorrection command corrects discrepancies that may occur during materialization by converting each update or insert operation into a delete followed by an insert.

You can use autocorrection in two ways:

- To set autocorrection from Replication Agent for one or all tables in the primary database, use the Replication Agent ra_set_autocorrection command as described in the *Replication Agent Reference Manual*.
- To set autocorrection from Replication Server, use the set autocorrection command in a replication definition. You must do this from Replication Server, however, because Replication Agent cannot alter the autocorrection setting on a replication definition.

Replication Agent for Oracle does not support use of the autocorrection feature for large-object (LOB), LONG, LONG RAW, or user-defined datatypes. Also, pds_username must have the ALTER ANY TABLE privilege to execute the following commands:

- `ALTER TABLE` ***`tablename`*** `ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`
- `ALTER TABLE` *`tablename`* `DROP SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`

Partitioned tables

Replication Agent supports Oracle partitioning functionality. Partitioning allows a table, index, or index-organized table to be subdivided into smaller pieces, where each piece of such a database object is called a partition. Each partition has its own name and may optionally have its own storage characteristics. Any table can be partitioned into many separate partitions except those tables containing columns with LONG or LONG RAW datatypes.

Unstructured data (such as images and documents) stored in a LOB column in the database can also be partitioned. When a table is partitioned, all the columns reside in the tablespace for that partition, with the exception of LOB columns, which can be stored in their own tablespace. For additional information about Oracle Partitioning, see the Oracle Database VLDB and Partitioning Guide at http://download.oracle.com/docs/cd/B28359_01/server.111/b32024/toc.htm.

Replicating the *truncate partition* command

There are two ways to replicate the truncate partition command:

- Use the lr_send_trunc_partition_ddl configuration property
- Wrap the truncate partition command in a stored procedure, and replicate the procedure

Using lr_send_trunc_partition_ddl

A configuration property has been added to Replication Agent, lr_send_trunc_partition_ddl, which can be used to determine whether truncate partition commands are sent as DDL or DML to the replicate database. The configuration can be:

- true (default) – the truncate partition command is sent as a DDL command (alter table). Use this setting to replicate to Oracle.
- false – the truncate partition is sent as a DML operation.Use this setting when replicating to databases that treat truncate partition commands as DML (for example, Microsoft SQL Server).

For information about Replication Agent configuration properties, see the *Replication Agent Reference Manual*.

Wrapping the *truncate partition* command

Alternately, you can wrap the truncate partition command in a stored procedure definition and replicate the procedure.

For example, to replicate truncate partition commands from an Oracle primary to an Adaptive Server Enterprise replicate, create the following stored procedure at the primary database:

```
create procedure sp_truncate_partition
as
begin
execute immediate ‘ALTER TABLE myTable TRUNCATE
```

```
PARTITION part1';
end;
```

Create a corresponding stored procedure at the replicate database:

```
create proc sp_truncate_partition as
truncate table myTable part1
```

Mark the sp_truncate_partition procedure for replication. When sp_truncate_partition is executed at the primary database, the truncate partition command is replicated to the replicate database.

Materialized views

A materialized view is a stored view query result. The data on which the view is defined is referred to as the master table (or tables). The materialized view is stored in its own table, which is refreshed based on changes to the master table. A materialized view may be local, in which it is defined on the same database as the master table, or remote, in which the materialized view is defined on a different database than the master table.

Oracle supports the following types of materialized views:

- Read-only – materialized view content is derived from the corresponding master table or tables, and the view content cannot be changed.
- Writeable – materialized view content can be changed temporarily, but any changes are overwritten when the table containing the materialized view is refreshed based on changes to the corresponding master table or tables.
- Updateable – updates made to a materialized view are written back to the corresponding master table or tables when the materialized view is refreshed.

For a complete description of materialized views, see the Oracle documentation.

Replication and materialized views

The following sections describe Replication Agent behavior with respect to Oracle materialized views.

Materialized view DDL

By default, Replication Agent does not replicate Oracle DDL commands used for materialized views, for example, the CREATE MATERIALIZED VIEW, ALTER MATERIALIZED VIEW, or DROP MATERIALIZED VIEW commands. Materialized view DDL commands are disabled from replication unless otherwise specified using the pbd_setrepddl command. To enable materialized view DDL commands for replication, see the pdb_setrepddl command in the *Replication Agent Reference Manual*.

Materialized views at the primary and replicate databases

A materialized view may exist on both the primary database and the replicate database. Such a situation might arise, for example, if materialized view DDL has been enabled for replication with the pdb_setrepddl command or if the replicate database has been materialized from a primary database dump.

If the master table on which the materialized view is defined exists in the primary database, Replication Agent will replicate this master table. The materialized view at the replicate database will refresh according to the contents of the replicated master table. Under no circumstances will Replication Agent replicate the table in which a materialized view is stored in the primary database, and you should not attempt to replicate such a table.

If the materialized view is remote—meaning that the master table on which the materialized view is defined does not exist in the primary database—the materialized view at the replicate database must be redirected so that it points to the database on which the master table is located. If the replicate database is not redirected, a refresh of the materialized view will fail at the replicate database. In redirecting the replicate database, instead of recreating the materialized view, you should recreate the Oracle database link that the replicate database uses to connect to the database containing the master table.

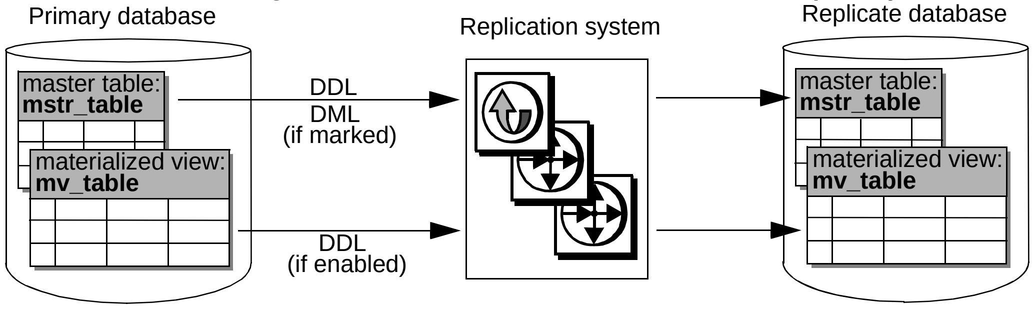
Writeable and updatable materialized views

Instead of replicating changes to the table containing a materialized view, Replication Agent replicates changes to the master table, if the master table has been marked for replication. Replication Agent therefore does not replicate changes made to a writeable materialized view. However, because changes made to an updatable materialized view are written back to the corresponding master table or tables when the materialized view is refreshed, Replication Agent will replicate changes made to an updatable materialized view on the primary database to the corresponding master table on the replicate database. Changes made to an updatable materialized view on the replicate database will only affect the local master table unless bidirectional replication has been enabled.

Materialized view replication scenarios

In Figure 1-1, a materialized view and a corresponding master table reside on both the primary database and the replicate database.

Figure 1-1: Master table and materialized view on primary database



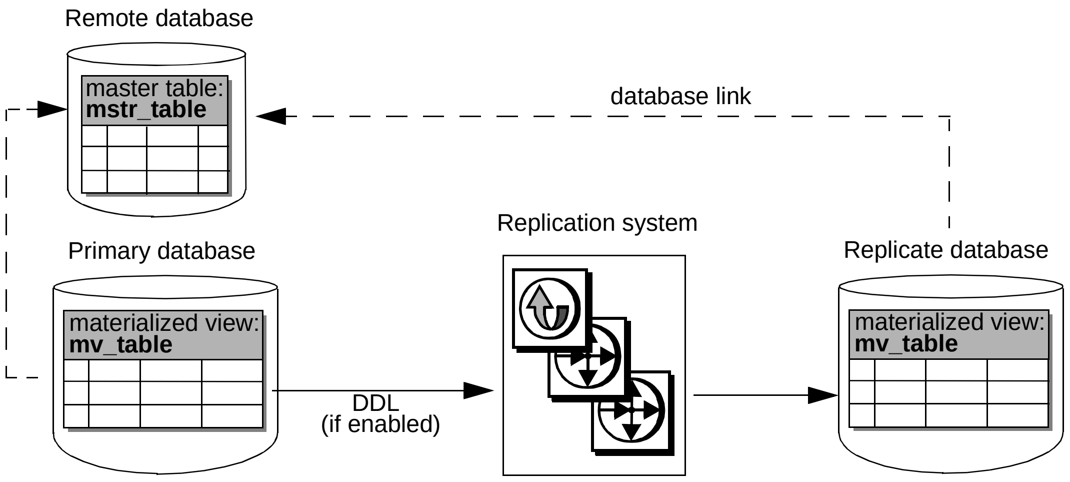
In this situation, DDL commands affecting the master table can be replicated as well as objects affected by the DML that are marked for replication.

DDL commands affecting the materialized view will not be replicated unless such DDL is enabled with the pdb_setrepddl command. Since a materialized view also exists on the replicate database, all master tables on which the materialized view is defined must also be replicated. Otherwise, the contents of the materialized view on the replicate database may become invalid.

If the materialized view on the primary database is updatable, changes made to this view are written back to the corresponding master table and, if the master table has been marked for replication, replicated to the replicate database. If the materialized view on the replicate database is updatable, changes made to this view are written back to the corresponding master table on the replicate database, but the master table on the primary database will not be changed accordingly unless bidirectional replication has been enabled.

In Figure 1-2, the master table on which the primary database materialized view is defined resides on a different, or remote, database.

Figure 1-2: Master table on remote database, materialized view on primary database

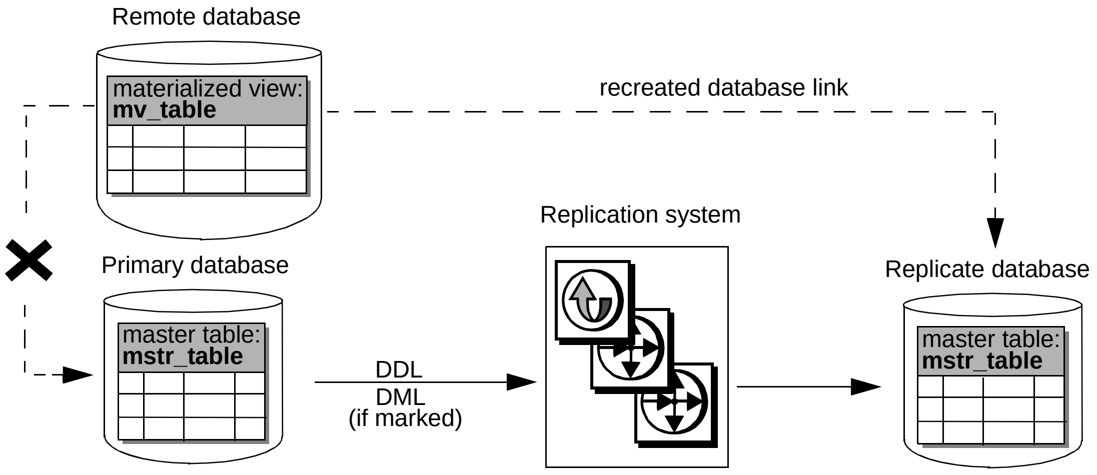


In this situation, neither DML nor DDL affecting the master table will be replicated. DDL commands affecting the materialized view will not be replicated unless such DDL is enabled with the pdb_setrepddl command. Since the materialized view also exists on the replicate database, a database link must be created so that it points to the database containing the master table on which the materialized view is defined.

If the materialized views on the primary and replicate databases are both updatable and are properly linked to the master table at the remote database, changes made to one of these views are written back to the master table, and the changes will be reflected in both materialized views upon refresh.

In Figure 1-3, a materialized view resides on a remote database, and the master table on which the materialized view is defined resides on the primary database. A copy of this master table also resides on the replicate database, and the database link between the remote and primary databases is subsequently broken.

Figure 1-3: Master table on primary database, materialized view on remote database

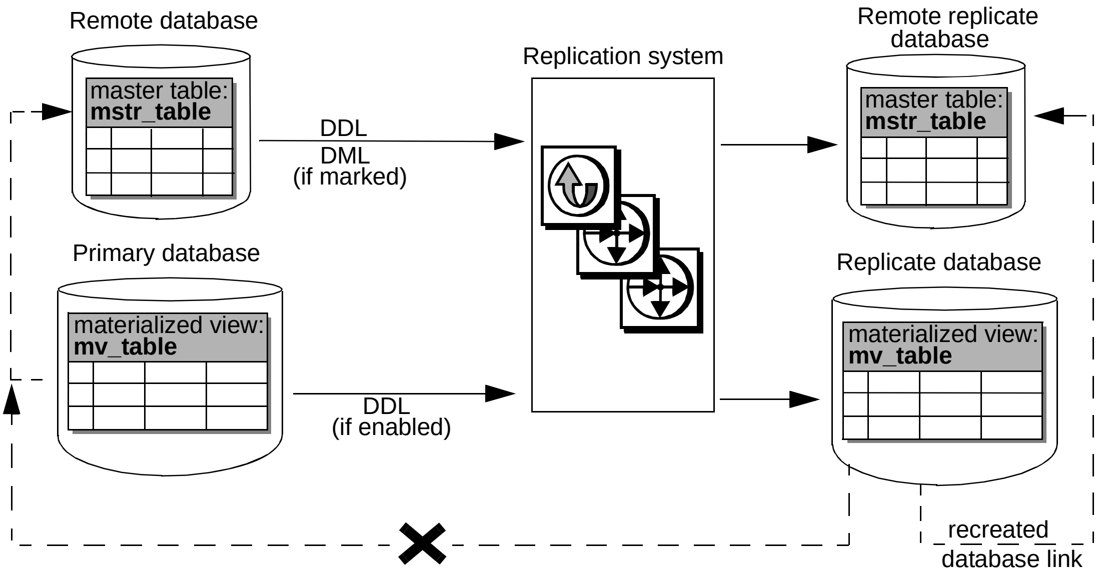


In this situation, DDL commands affecting the master table at the primary database can be replicated as well as DML commands that are marked for replication. DDL commands affecting the materialized view cannot be replicated because there is no corresponding materialized view on the replicate database. If the database link between the remote and primary databases is broken, the remote database must create a link to the replicate database before a refresh occurs.

If the materialized view on the remote database is updatable, changes made to this view are written back to the master table on the database to which the remote database is currently linked.

In Figure 1-4, a master table resides on two different remote databases, one of which is a replicate database. A materialized view resides on the primary database and the replicate database. The materialized view at the replicate database is initially defined by the master table on the remote database, but its database link becomes broken, and the replicate database recreates a link to the remote replicate database instead.

Figure 1-4: Master tables on remote databases, materialized views on primary and replicate databases



If the materialized view on the replicate database is updatable, changes made to this view are written back to the master table on the database to which the replicate database is currently linked. Before the database link between the replicate database and the remote database becomes broken, updates to the materialized view on the replicate database are written back only to the master table on the remote database. After a link is created between the replicate database and the remote replicate database, updates to the materialized view on the replicate database are written back only to the master table on the remote replicate database.

Index-organized tables

Replication Agent can replicate DML for the following types of index-organized tables (IOTs) for Oracle 10g and 11g databases:

- Simple IOTs
- IOTs with including and overflow clauses
- IOTs with composite partitions
- IOTs with mapping tables

- Index-compressed IOTs
- IOTs with row dependencies
- IOTs with large objects
- IOTs with secondary indexes

Replication Agent cannot replicate IOTs with nested tables and columns of type VARRAY.

Controlling trigger execution at the replicate database

Trigger execution in a replication system can cause problems when both the transaction that caused a trigger to fire and the transactions that resulted from the trigger are replicated. This may result in data duplication in the case where a trigger causes data to be recorded twice for a single operation performed on an audited table. It may also result in data inconsistency in the case where a trigger results in DML commands being performed twice: once as a result of the trigger firing at the primary database and a second time as a result of the trigger firing at the replicate database to which trigger-altered data has already been replicated. To avoid data duplication and inconsistency, it is important to control trigger execution in the replication system. However, Oracle provides no session-level command to disable trigger execution.

Replication Server allows you to disable trigger execution at the session or connection level. You can control trigger firing each time a PL/SQL command is executed against the replicate database. Controlling trigger execution at the replicate database eliminates data duplication and inconsistency caused by the absence of any trigger control at the replicate database.

For a complete description of the Replication Server rs_triggers_reset function, see the *Replication Server Reference Manual*. For complete instructions on controlling trigger execution at the replicate database, see Chapter 10, “Oracle Replicate Data Server Issues” in the *Replication Server Heterogeneous Replication Guide*.

Altering replication definitions from the primary data server

To avoid having to quiesce the replication system before altering a replication definition, you can issue the Replication Server alter replication definition command from the primary data server and make schema changes to primary database objects at the same time. The propagation of changes to a replication definition can be automatically coordinated with data replication without having to stop the replication process.

To issue the Replication Server alter replication definition command from the primary data server, a stored procedure named rs_send_repserver_cmd must be created in the primary Oracle database. The SQL for creating this procedure is contained in the appropriate connection profile on Replication Server. For a list of connection profiles, use the Replication Server admin show_connection_profiles as described in the *Replication Server Reference Manual*.

For a full description of rs_send_repserver_cmd and the alter replication definition Replication Server command, see the *Replication Server Reference Manual*.

Security considerations

When the rs_send_repserver_cmd procedure is invoked at the primary data server, Replication Agent passes corresponding Replication Command Language (RCL) directly to Replication Server. You should therefore consider carefully to whom execution privileges are assigned for the rs_send_repserver_cmd procedure, and assign privileges as appropriate for your environment and security policy.

Limitations

The rs_send_repserver_cmd procedure cannot be used to alter replication definitions for tables that contain columns of the following types:

- BINARY ROWID
- BINARY UROWID
- DATE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- MLSLABEL

- RAW
- REF
- TIMESTAMP
- TIMESTAMP WITH LOCAL TZ
- TIMESTAMP WITH TZ

Note If you manually change a table-level replication definition in Replication Server, you must then suspend and resume replication in the Replication Agent to ensure that the Replication Agent clears and refreshes its cache.

Oracle Data Guard

Data Guard is a disaster-protection architecture consisting of a primary Oracle database and one or more standby Oracle databases to which the primary database copies its data. These standby copies can be used in the event that the primary Oracle database fails. Replication Agent supports replication of data from an Oracle database system that uses Data Guard.

Recommended configuration

Although Replication Agent can be configured to replicate data from either the Data Guard primary database or a Data Guard standby database, Sybase recommends that you configure Replication Agent to read from a Data Guard standby database transaction log. This way, if the primary Data Guard database fails over to the standby Data Guard database, Replication Agent is already connected to a working Oracle database, and replication is not disrupted.

If you configure Replication Agent to read from a Data Guard standby database transaction log, pds_username must have alter database permission.

Database resynchronization

You can avoid having to quiesce the primary database when you initialize Replication Agent for Oracle if your replication system is configured for database resynchronization. For information on configuring database resynchronization, see Chapter 13, "Resynchronizing Oracle Replicate Databases" in the *Replication Server Heterogeneous Replication Guide*.

Troubleshooting Oracle transactions and operations

The ra_dumptran and ra_helpop commands return information for use in troubleshooting a specified Oracle database transaction or database operation, respectively.

The ra_dumptran and ra_helpop commands use information gathered by Oracle LogMiner to help you troubleshoot Replication Agent for Oracle. Oracle LogMiner consists of Oracle procedures and views that allow you to obtain detailed information about database activities from the Oracle redo logs. To use ra_dumptran and ra_helpop, you must install Oracle LogMiner, or these commands will return errors.

❖ Setting up Replication Agent and Oracle to use *ra_dumptran* and *ra_helpop*

1. Go to *$ORACLE_HOME/rdbms/admin*.
2. Log in as a “sys as sysdba” user.
3. Execute the Oracle LogMiner installation script:

   ```
   @dbmslm.sql
   ```

4. After LogMiner is installed, create a public synonym so that you do not have to log in as the owner to execute LogMiner functions:

   ```
   CREATE PUBLIC SYNONYM DBMS_LOGMNR FOR
      SYS.DBMS_LOGMNR;
   ```

   Note This step is required if you are using Oracle 10g.

5. Grant the following privileges to pds_username:

   - EXECUTE_CATALOG_ROLE
   - select on V_$LOGMNR_CONTENTS
   - select on V_$LOGMNR_LOGS
   - select any transaction

   Note These privileges are not required for replication, only for using the ra_dumptran and ra_helpop commands. The ra_migrate command will verify that these privileges have been granted to pds_username. If these privileges have not been granted at the time ra_migrate is invoked, a warning message is returned and logged in the Replication Agent log file.

6 Use ra_dumptran and ra_helpop according to instructions provided in the *Replication Agent Reference Manual*.

Replicating stored procedures with arguments of boolean type

To replicate an Oracle stored procedure with an argument of type BOOLEAN, Replication Agent sends the boolean argument to Replication Server as an integer. Replication Server then uses a function string to convert the argument back to a boolean value so that the stored procedure can be executed on the replicate database. On Replication Server, you must manually create this function string for each Oracle stored procedure that has an argument of type BOOLEAN.

Some replicate databases do not support boolean-type stored procedure arguments. In these cases, the Oracle stored procedure BOOLEAN argument should be mapped to an integer argument in the corresponding stored procedure at the replicate database. A function string is then unnecessary.

The following examples illustrate how to replicate an Oracle stored procedure with an argument of type BOOLEAN to a Oracle replicate database and to a non-Oracle replicate database.

Example 1: Replicating to an Oracle replicate database

The user wants to replicate a stored procedure defined by the following PL/SQL:

```
CREATE PROCEDURE boolproc (a IN BOOLEAN, b INT) AS
BEGIN
   IF (a = true) THEN
      DBMS_OUTPUT.PUT_LINE('True');
   ELSE
      DBMS_OUTPUT.PUT_LINE('False or NULL');
   ENDIF;
END;
```

The user manually creates a replication definition on Replication Server using the following RCL:

```
create function replication definition ra$xxx_boolproc
with primary at myprimary.pdb
with all functions named boolproc (
@"a" rs_oracle_decimal
@"b" rs_oracle_decimal )
searchable parameters(@"a", @"b")
```

```
send standby all parameters
```

Note If the Replication Agent pdb_auto_create_repdefs configuration parameter is set to true, a replication definition will be created automatically.

The user then marks the stored procedure for replication:

```
pdb_setrepproc boolproc, mark
```

The user creates a function string on Replication Server:

```
create function string ra$xxx_boolproc.boolproc
for rs_oracle_function_class
output language
'begin execute immediate "begin ra_user.boolproc
(?a!param?=1,?b!param?);;end;;";;end;;'
go
```

The user then creates a subscription on Replication Server for the replication definition:

```
create subscription sub_intproc
for ra$xxx_boolproc
with replicate at myreplicate.rdb
go
```

The stored procedure will be replicated when it is executed at the primary database:

```
EXECUTE boolproc(true,1);
```

Example 2: Replicating to a non-Oracle database

The user wants to replicate a stored procedure defined by the following PL/SQL:

```
CREATE PROCEDURE boolproc (a IN BOOLEAN, b INT) AS
BEGIN
   IF (a = true) THEN
      DBMS_OUTPUT.PUT_LINE('True');
   ELSE
      DBMS_OUTPUT.PUT_LINE('False or NULL');
   ENDIF;
END;
```

The user manually creates a replication definition on Replication Server using the following RCL:

```
create function replication definition ra$xxx_boolproc
with primary at myprimary.pdb
with all functions named boolproc (
@"a" rs_oracle_decimal
@"b" rs_oracle_decimal )
searchable parameters(@"a", @"b")
send standby all parameters
```

Note If the Replication Agent pdb_auto_create_repdefs configuration parameter is set to true, a replication definition will be created automatically.

The user then marks the stored procedure for replication:

```
pdb_setrepproc boolproc, mark
```

Adaptive Server Enterprise does not support boolean-type stored procedure arguments, so the user maps the Oracle stored procedure BOOLEAN argument to an integer argument for the corresponding stored procedure at the replicate database.

The user creates a stored procedure, defined by the following Transact-SQL, at the replicate database:

```
create proc boolproc (@a int, @b int) as
if @a = 1
   print 'True'
else
   print 'False or NULL'
go
```

The user then creates a subscription on Replication Server for the replication definition:

```
create subscription sub_intproc
for ra$xxx_boolproc
with replicate at myreplicate.rdb
go
```

The stored procedure will be replicated when it is executed at the primary database:

```
EXECUTE boolproc(true,1);
```

However, the boolproc procedure at the replicate Adaptive Server Enterprise will be invoked with an integer value instead of a boolean argument:

```
boolproc 1, 1
go
```

Oracle warm standby

Using Replication Server, you can create and maintain a warm standby environment for an Oracle database. In standby mode, Replication Agent:

- scans the transaction log and keeps the Replication Agent System Database (RASD) current.
- does not send any LTL to Replication Server
- continues to perform log truncation.

When the active database fails or when you want to perform maintenance on the active database, you can switch to the standby database. For instructions on switching the active and standby database, see Chapter 12, “Managing Heterogeneous Warm Standby for Oracle” in the *Replication Server Heterogeneous Replication Guide*.

To function in warm standby mode:

- Replication Agent must be installed on both the primary and standby side and must also be successfully initialized. The Replication Agent on the standby side should be running in standby mode with the ra_standby parameter set to true.
- Replication Agent should have the rs_source_ds and rs_source_db parameters configured as physical connections to Replication Server.
- Replication Agent should enable or disable the replication of DDL statements as desired using the pdb_setrep_ddl command.
- Replication Agent should set the pdb_auto_create_repdefs, pdb_dflt_column_repl, pdb_dflt_object_repl, and pdb_automark_tables parameters to true.

For details on using these commands and configuration parameters, see the *Replication Agent Reference Manual*. For detailed steps involved in creating and managing warm standby for Oracle, see the *Replication Agent Heterogeneous Replication Guide*.

Oracle Flashback

Replication Agent can replicate Oracle Flashback operations performed at the table level and the transaction level:

- Replication Agent can replicate Flashback Table commands to restore a database back to a System Change Number (SCN), timestamp, or restore point within the threshold specified by the Oracle UNDO_RETENTION parameter. If a table has been marked for replication, Replication Agent can replicate any DML changes that result from executing the contents of the UNDO_SQL column of the Oracle FLASHBACK_TRANSACTION_QUERY view.
- Replication Agent will replicate all of the following Oracle DDL commands:
    - DROP TABLE *table* (when the Recycle Bin is enabled)
    - FLASHBACK TABLE *table* TO BEFORE DROP
    - PURGE TABLE *table*
    - PURGE INDEX *index*
    - PURGE TABLESPACE *tablespace*
    - PURGE RECYCLEBIN
    - PURGE DBA_RECYCLEBIN

Note Since the replicate database may not be an exact copy of the primary database, these DDL commands may not execute successfully at the replicate database or may have a different result at the replicate database. For example, the PURGE DBA_RECYCLEBIN command will purge more objects at the replicate database if the Recycle Bin for the replicate database contains more objects than the primary database Recycle Bin.

- Replication Agent can read from the Oracle Flash Recovery Area if the pdb_archive_path configuration parameter is set to that location.

Requirements

To use Oracle Flashback with Replication Agent:

- The user ID for pds_username must have select permission on SYS.RECYCLEBIN$.
- For Replication Agent to replicate the Oracle PURGE DBA_RECYCLEBIN command, the user ID for ddl_username must have sysdba permission and should be suffixed with "as sysdba". For example:

```
ra_config ddl_username, "myuser as sysdba"
go
```

- Replication Agent will replicate Flashback operations from Oracle 10g and 11g databases but not from earlier versions of Oracle.

Limitations

Replication of Oracle Flashback commands is subject to the following limitations:

- If the replicate database does not have the Recycle Bin enabled, Flashback commands, the PURGE command, and any command that accesses Recycle Bin objects will fail, even if a DROP TABLE command has been successfully replicated.
- Replication Agent does not support the translation of DDL commands between a primary and replicate database of different types. Consequently, DDL replication must be disabled when replicating from an Oracle primary database to a non-Oracle replicate database, and Flashback DDL commands cannot be replicated in this case.
- If an Oracle FLASHBACK TABLE command is issued with the RENAME TO clause, Replication Agent will not automatically update the replication definition with the new table name. You must do this manually.
- Replication Agent reconstructs Flashback commands based on the original object name, not on the object Recycle Bin name. When there are multiple versions of an object in the Recycle Bin, Replication Agent reconstructs a Flashback command to use the most recent version of the object that exists in the Recycle Bin at the replicate database. Consequently, subsequent Oracle commands that affect the Recycle Bin may result in inconsistency between the primary and replicate databases.

  For example, the primary Oracle database contains the following dropped versions of table TAB1:

```
SQL> SELECT object_name as recycle_name, original_name, type FROM
recyclebin;
RECYCLE_NAME                      ORIGINAL_NAME  TYPE
--------------------------------  -------------  -----
BIN$zyxwvutsrqponmlkjihgfedcba$1  TAB1           TABLE
BIN$zyxwvutsrqponmlkjihgfedcba$2  TAB1           TABLE
BIN$zyxwvutsrqponmlkjihgfedcba$3  TAB1           TABLE
```

The replicate Oracle database contains the following dropped versions of table TAB1:

```
SQL> SELECT object_name as recycle_name, original_name, type FROM
recyclebin;
```

```
RECYCLE_NAME                      ORIGINAL_NAME  TYPE
--------------------------------  -------------  -----
BIN$abcdefghijklmnopqrstuvwxyz$1  TAB1           TABLE
BIN$abcdefghijklmnopqrstuvwxyz$2  TAB1           TABLE
BIN$abcdefghijklmnopqrstuvwxyz$3  TAB1           TABLE
```

The following Flashback command is executed at the primary Oracle database:

```
FLASHBACK TABLE "BIN$zyxwvutsrqponmlkjihgfedcba$2" TO BEFORE DROP;
```

Because Replication Agent reconstructs Flashback commands based on the original object name and uses the most recent version of a dropped object in the Flashback command, the following command is executed at the replicate Oracle database:

```
FLASHBACK TABLE "BIN$abcdefghijklmnopqrstuvwxyz$3" TO BEFORE DROP;
```

If BIN$zyxwvutsrqponmlkjihgfedcba$2 differs in content from BIN$abcdefghijklmnopqrstuvwxyz$3, the primary and replicate databases have become inconsistent.

Disabling Oracle Recycle Bin

If you do not intend to use the Recycle Bin at the replicate Oracle database, you can disable it manually. To disable the Recycle Bin, which requires the sysdba privilege, enter the following command, and then restart the replicate Oracle database:

```
ALTER SYSTEM SET RECYCLEBIN=OFF SCOPE=SPFILE;
```

Or, if you are using a version of Oracle that does not have the RECYCLEBIN parameter, enter the following:

```
ALTER SYSTEM SET "_recyclebin"=FALSE SCOPE=BOTH;
```

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

Dropped objects and article status

If a marked table is dropped while the Recycle Bin is enabled at the primary database, the ra_helparticle command still reports the status of the corresponding article as Current. The ra_helparticle command will only report the status of the corresponding article as Dropped after the dropped table is purged from the primary database Recycle Bin.

Disabling Flashback replication with Recycle Bin disabled

If the replicate Oracle database does not have the Recycle Bin enabled, any command that accesses Recycle Bin objects at the replicate database will fail. You should therefore disable the replication of Flashback DDL commands in one of the following ways:

- Use the pdb_setrepddl command to prevent the replication of Flashback DDL commands. See the *Replication Agent Reference Manual*.
- Add a "warning" error action for Flashback DDL execution failure to Replication Server so that replication can continue with the replicate Recycle Bin disabled. Run the following Replication Server script against the RSSD:

  $SYBASE/RAX-15_5/scripts/oracle/hds_oracle_setup_flashback_errors.sql

  After you run this script, you must restart Replication Server.

  Note This script uses the rs_oracle_error_class default error class as a template. If you are using custom error class and want Replication Server to continue replicating without interruption, you must instruct Replication Server to display warning messages 38305 and 38307 in its error log:

  ```
  assign action warn for your_error_class to 38305,
  38307
  ```

  where *your_error_class* is the name of your custom error class.

  To remove the "warning" error action for Flashback DDL execution failure to Replication Server, run the following Replication Server script against the RSSD:

  $SYBASE/RAX-15_5/scripts/oracle/hds_oracle_remove_flashback_errors.sql

Replicating XMLTYPE data

When an Oracle table is created with an XMLTYPE column but without any XML schema specification, a hidden CLOB column is automatically created to store the XML data. The XMLTYPE column becomes a virtual column for the hidden CLOB column. In the corresponding Oracle base table, the hidden column immediately follows the XMLTYPE column that it represents and is named SYS_NC*nnnnn*$, where *nnnnn* represents the position of the hidden column in the base table. For example, a table is created in the Oracle database with the following DDL:

```
CREATE TABLE sampletable
( col1 INT,
, col2 INT,
, xml1 XMLTYPE
, xml2 XMLTYPE);
```

The Oracle database creates hidden columns named SYS_NC00004$ and SYS_NC00006$, which respectively correspond to the xml1 and xml2 columns. These hidden CLOB columns cannot be accessed directly. However, they can be viewed by querying the col$ and obj$ base tables of the Oracle data dictionary. See the Oracle documentation for information.

Example 1: Replicating XMLTYPE column data from Oracle to Oracle

The user wants to replicate a table defined by the following DDL:

```
CREATE TABLE sampletable
( col1 INT
, col2 INT
, xml1 XMLTYPE
, xml2 XMLTYPE);
```

The user manually creates a replication definition on Replication Server using the following RCL:

```
create replication definition ra$xxx_sampletable
with primary at myprimary.pdb
with all tables named sampletable (
col1 int,
col2 int,
xml1 as SYS_NC00004$ text,
xml2 as SYS_NC00006$ text )
primary key (col1, col2)
```

```
go
```

Note If the Replication Agent pdb_auto_create_repdefs configuration parameter is set to true, a replication definition will be created automatically.

The user then marks the table for replication:

```
pdb_setreptable sampletable, mark
```

Because of the hidden CLOB columns, replication must be enabled for the table using pdb_setrepcol:

```
pdb_setrepcol sampletable, enable
```

The user creates a corresponding table at the replicate Oracle database:

```
CREATE TABLE sampletable
( col1 INT
, col2 INT
, xml1 XMLTYPE
, xml2 XMLTYPE);
```

Note If DDL replication has been enabled, it is unnecessary to create the replicate table manually.

Example 2: Replicating XMLTYPE column data from Oracle to Adaptive Server Enterprise

The user wants to replicate a table defined by the following DDL:

```
CREATE TABLE sampletable
( col1 INT
, col2 INT
, xml1 XMLTYPE
, xml2 XMLTYPE);
```

The user manually creates a replication definition on Replication Server using the following RCL:

```
create replication definition ra$xxx_sampletable
with primary at myprimary.pdb
with all tables named sampletable (
col1 int,
col2 int,
xml1 as SYS_NC00004$ text,
xml2 as SYS_NC00006$ text )
```

```
primary key (col1, col2)
go
```

Note If the Replication Agent pdb_auto_create_repdefs configuration parameter is set to true, a replication definition will be created automatically.

The user then marks the table for replication:

```
pdb_setreptable sampletable, mark
```

Because of the hidden CLOB columns, replication must be enabled for the table using pdb_setrepcol:

```
pdb_setrepcol sampletable, enable
```

The user creates a corresponding table at the replicate Adaptive Server Enterprise database using the hidden column names from the primary database table:

```
create table sampletable
( col1 int,
col2 int,
SYS_NC00004$ text,
SYS_NC00006$ text)
go
```

Note The XMLTYPE columns from the Oracle primary database table map to text columns in the Adaptive Server Enterprise replicate database table.

Example 3: Replicating an XMLTYPE table from Oracle to Oracle

A table, defined by the following DDL, must be replicated:

```
CREATE TABLE sampletable OF XMLTYPE;
```

This statement creates a simple XMLTYPE table with one implicit CLOB column that can be accessed through a default pseudocolumn named XMLDATA.

The user manually creates a replication definition on Replication Server using the following RCL and the hidden column SYS_NC_OID$, which contains the object ID for sampletable:

```
create replication definition ra$xxx_sampletable
with primary at myprimary.pdb
with all tables named sampletable (
SYS_NC_OID$ rs_oracle_binary,
XMLDATA text )
primary key (SYS_NC_OID$)
go
```

The Replication Server name for the Oracle RAW datatype is rs_oracle_binary.

Note If the Replication Agent pdb_auto_create_repdefs configuration parameter is set to true, a replication definition will be created automatically.

The user then marks the table for replication:

```
pdb_setreptable sampletable, mark
```

The user creates a corresponding table at the replicate Oracle database:

```
CREATE TABLE sampletable OF XMLTYPE;
```

Oracle 9i

Replication Agent continues to support for Oracle 9i when Oracle 10g or 11g is running in 9i compatibility mode.

Limitations

The following Replication Agent features cannot be used with Oracle 9i:

- Oracle index-organized tables and the ANYDATA datatype
- Autocorrection
- Replication of XMLTYPE data
- Flashback
- RAC
- ASM

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_5\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 nonalphanumeric 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 nonalphanumeric characters allowed in a database object name are the $, #, and _ characters.

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.

❖ **Finding 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 1-4 lists the tables that are considered Replication Agent objects.

Table 1-4: Replication Agent tables

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

Marker objects

Table 1-5 lists the Replication Agent objects related to Replication Server markers. No permissions are granted when these objects are created.

Table 1-5: Replication Agent marker objects

Object	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 1-6 lists the Oracle sequences that are considered Replication Agent objects.

Table 1-6: Replication Agent sequences

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

Marked procedures

Table 1-7 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 1-7: 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 perform 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.

CHAPTER 2

Replication Agent for Microsoft SQL Server

The term “Replication Agent for Microsoft SQL Server” refers to an instance of the Replication Agent software that is 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.

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

Microsoft SQL Server-specific considerations

This section describes general issues and considerations that are specific to using Replication Agent 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*.

- Microsoft SQL Server requirements
- Microsoft SQL Server restrictions
- Unsupported software features
- Unsupported datatypes
- Applying Microsoft SQL Server patches
- DDL Replication
- Replication Agent connectivity
- Replication Agent permissions and roles
- The sybfilter driver
- Initialization of the primary data server and Replication Agent
- Microsoft sqlcmd tool
- Character case of database object names
- Format of origin queue ID
- Datatype compatibility
- Replicating ntext datatypes
- Installing Replication Agent and the data server on different machines
- Altering replication definitions from the primary data server
- Upgrading the Microsoft SQL Server service pack
- Replication Server set autocorrection command
- Computed columns
- Microsoft SQL Server 2008

Microsoft SQL Server requirements

Observe the following requirements for Microsoft SQL Server:

- Replication Agent supports Microsoft SQL Server 2005 Service Pack 2, and the database compatibility level must be set to “SQL Server 2005 (90)”. Replication Agent also supports Microsoft SQL Server 2008, but support is limited to functionality also present in Microsoft SQL Server 2005.
- 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.

Microsoft SQL Server restrictions

Microsoft SQL Server imposes the following restrictions when used as the primary database with Replication Agent:

- Use of the TRUNCATE TABLE command on a table that has been marked for replication is forbidden.
- Dropping a stored procedure that has been marked for replication is forbidden.

Unsupported software features

The following features are not supported for Sybase replication:

- Microsoft SQL Server clusters
- Microsoft SQL Server virtual computed columns
- Replication Server parallel DSI (for non-Adaptive Server Enterprise databases)
- Replication Server warm standby (for non-Adaptive Server Enterprise databases)

- Replication Server rs_init utility (for non-Adaptive Server Enterprise databases)
- Replication Server rs_subcomp utility (for non-Adaptive Server Enterprise databases)
- Replication Server automatic materialization (for non-Adaptive Server Enterprise databases)
- Replication Server when replicating in an environment where other vendors are replicating (for non-Adaptive Server Enterprise databases)
- Some Microsoft SQL Server 2008 features. See "Unsupported Microsoft SQL Server 2008 features" on page 99.

Unsupported datatypes

The following datatypes are not supported for Sybase replication:

- cursor
- table
- xml
- Some Microsoft SQL Server 2008 datatypes. See "Unsupported Microsoft SQL Server 2008 datatypes" on page 98.

Applying Microsoft SQL Server patches

The following is the procedure for applying Microsoft SQL Server patches.

❖ **Applying Microsoft SQL Server patches**

1 Before applying a patch, be sure that all data has been replicated to the replicate site.

Note All activities must stop before this step, and all users except the pds_username should log off from the primary database.

For each existing Replication Agent for Microsoft SQL Server instance, verify that it is in *Replicating* state and allow replication to finish. To verify that replication has finished, quiesce the Replication Agent instance by issuing the quiesce command.

Note It may take a while for the command to return because Replication Agent reads all data from the log file and sends it to the Replication Server.

2 Before applying the service patch disable the database triggers.

If the pdb_automark_tables configuration parameter is set to true, log on to the primary database and disable the automark trigger by issuing:

```
DISABLE TRIGGER ra_createtable_trig_ ON DATABASE
```

where ra_createtable_trig_ is the name of the automark trigger created by Replication Agent.

3 Apply the service patch using the instructions in the Microsoft documentation.

4 Regenerate the objects in the Microsoft SQL Server system resource database.

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

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

  where *instanceName* is the name of the Microsoft SQL Server instance.

- Log in to the Replication Agent instance:

  ```
  isql -U username -P password -S instanceName
  ```

- Reinitialize Microsoft SQL Server :

  ```
  server_xlog remove, force
  go
  ```

  ```
  server_xlog init
  go
  ```

- Restart Microsoft SQL Server in multi-user mode.

5 If the pdb_automark_tables configuration parameter is set to true before applying the patch:

- Log on to the primary database, and enable the automark trigger by issuing the following command:

  ```
  ENABLE TRIGGER ra_createtable_trig_ ON DATABASE
  ```

  where ra_createtable_trig_ is the name of the automark trigger created by Replication Agent.

- Log on to the primary database, and enable the DDL trigger by issuing the following command:

  ```
  ENABLE TRIGGER ra_ddl_trig_ ON DATABASE
  ```

  where ra_ddl_trig_ is the name of the DDL trigger created by Replication Agent.

6 Zero the LTM locator, and move the truncation point to the end of the log:

- Zero the LTM locator by logging in to RSSD and issuing the following command:

  ```
  rs_zeroltm < ra_instance > , < pdb_name > "
  ```

- Move the truncation point to the end of log by logging in to Replication Agent and issuing the following command:

  ```
  pdb_init move_truncpt
  ```

7 Resume replication or other operations in Replication Agent. The primary database will be accessible to users.

DDL Replication

Replication of data definition language (DDL) commands is supported, but only to Microsoft SQL Server.

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. See the *Replication Agent Reference Manual* for details on using the pdb_setrepddl command.

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, 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*.

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, 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. 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. 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*.

Replication Agent connectivity

Replication Agent for Microsoft SQL Server uses the 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.

See the *Replication Agent Administration Guide*.

Replication Agent permissions and roles

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

The following permissions must be granted to the user specified by the pds_username parameter:

- 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

The user specified by the pds_username parameter must be added to the following roles in the primary database:

- db_owner – 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.
- sysadmin – 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 standalone mode. Subsequent executions of pdb_xlog init do not modify the server again and do not require the additional permission or configurations.

First-time initialization

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

❖ Initializing 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):

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

To execute sp_configure with both parameters to change a configuration option or to run the RECONFIGURE statement, you must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the sysadmin and serveradmin fixed server roles.

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 *Replication Agent 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.

   a. Update your *sql.ini* file with the instance name and port number of your Replication Agent instance.

   b. Log in to the Replication Agent instance:

      ```
      isql -U username -P password -S instanceName
      ```

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

   c. Issue the following command.

      ```
      test_connection PDS
      ```

9. Initialize the Microsoft SQL Server data server and Replication Agent using the server_xlog and pdb_xlog commands:

   ```
   server_xlog init
   ```

   ```
   pbd_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 99. The sp_SybSetLogforReplTable, sp_SybSetLogforReplProc, and sp_SybSetLogforLOBCol procedures are created in the mssqlsystemresource database with execute permission granted to Public.

10. Stop the Microsoft SQL Server service again.

   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.

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.

❖ **Subsequently initializing 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:

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

   To execute sp_configure with both parameters to change a configuration option or to run the RECONFIGURE statement, you must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the sysadmin and serveradmin fixed server roles.

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 *Replication Agent 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.

❖ Cleaning 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 Remove the special marking procedures from the mssqlsystemresource database:

```
drop procedure sp_SybSetLogforReplTable;

drop procedure sp_SybSetLogforReplProc;

drop procedure sp_SybSetLogforLOBCol;
```

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

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

Microsoft sqlcmd tool

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

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 fails. 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 in which 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 in which 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 in which 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 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 2-1 illustrates the format of the origin queue ID for the Replication Agent for Microsoft SQL Server.

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

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

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
bigint	-2^{63} to 2^{63} - 1	bigint	-2^{63} to 2^{63} - 1	
binary	Fixed length up to 8000 bytes	binary	32K	
bit	Integer with value of 0 or 1	bit	Integer with value of 0 or 1	
char	Fixed length up to 8000 characters	char	32K	
datetime	Date and time from 01/01/1753 to 12/31/9999	datetime	Date and time from 01/01/1753 to 12/31/9999	
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
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.
image	Variable length up to 2^{31} - 1 bytes	image	2GB	
int	-2^{31} to 2^{31} - 1	int	-2^{31} to 2^{31} - 1	
money	Monetary from -2^{63} to 2^{63} - 1	money	Monetary from -2^{63} to 2^{63} - 1	
nchar	Fixed length Unicode up to 4000 characters	unichar or char	32K	Actual maximum length is @@ncharsize * number of characters.
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.
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	The nvarchar(max) datatype cannot be replicated to data servers other than Microsoft SQL Server. For Replication Server 15.0 and later versions, nvarchar(max) maps to unitext. For earlier versions of Replication Server, nvarchar(max) maps to image.
numeric	Synonym for decimal datatype	numeric	Synonym for decimal datatype	
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.
smalldatetime	Date and time from 01/01/1900 to 06/06/2079	datetime	Date and time from 01/01/1900 to 06/06/2079	
smallint	Integer with value from -2^{15} to 2^{15} - 1	smallint	Integer with value from -2^{15} to 2^{15} - 1	
smallmoney	Monetary from -214,748.3648 to 214,748.3647	smallmoney	Monetary from -214,748.3648 to 214,748.3647	

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
sql_variant	Any datatype except text, ntext, timestamp, and sql_variant, up to 8000 bytes	varchar or opaque	32K	For replication to Replication Server 15.0 and earlier versions, the Sybase datatype should be varchar. For replication to Replication Server 15.1 or later, the Sybase datatype should be opaque.
text	Variable length up to 2^{31} - 1 characters	text	2GB	
timestamp	Database-wide unique number	timestamp or varbinary	Database-wide unique number	For replication to Replication Server 15.0 and earlier versions, the Sybase datatype should be varbinary(8). For replication to Replication Server 15.1 or later, the Sybase datatype should be timestamp.
tinyint	Integer with value from 0 to 255	tinyint	Integer with value from 0 to 255	
uniqueidentifier	Globally unique identifier	char	Globally unique identifier	No Sybase equivalent. Map to char(38).
varbinary	Variable length up to 8000 bytes	varbinary	32K	
varbinary(max)	Variable length up to 2^{31} - 1 bytes	image	2GB	The varbinary(max) datatype cannot be replicated to data servers other than Microsoft SQL Server.
varchar	Variable length up to 8000 characters	varchar	32K	
varchar(max)	Variable length up to 2^{31} - 1 characters	text	2GB	The varchar(max) datatype cannot be replicated to data servers other than Microsoft SQL Server.

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 2-3 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

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. By default, the ltl_big_endian_unitext property is true. 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.

Installing Replication Agent and the data server on different machines

If you are installing Replication Agent on a machine separate from that on which Microsoft SQL Server is running, observe the following requirements:

- The Replication Agent machine must have the same type of operating system and hardware as the machine on which Microsoft SQL Server is running.
- On the Replication Agent machine, map the network drives that contain the primary Microsoft SQL Server database transaction log files, and use the ra_devicepath command to point Replication Agent to the mapped location for the database transaction log files.
- The sybfilter driver must be installed on the same machine as the primary Microsoft SQL Server. When configuring the sybfilter driver, use directory and drive names that are recognizable to the primary Microsoft SQL Server.

Altering replication definitions from the primary data server

To avoid having to quiesce the replication system before altering a replication definition, you can issue the Replication Server alter replication definition command from the primary data server and make schema changes to primary database objects at the same time. The propagation of changes to a replication definition can be automatically coordinated with data replication without having to stop the replication process.

To issue the Replication Server alter replication definition command from the primary data server, a stored procedure named rs_send_repserver_cmd must be created in the primary Microsoft SQL Server database. The SQL for creating this procedure is contained in the appropriate connection profile on Replication Server. For a list of connection profiles, use the Replication Server admin show_connection_profiles as described in the *Replication Server Reference Manual*.

For a full description of rs_send_repserver_cmd and the alter replication definition Replication Server command, see the *Replication Server Reference Manual*.

Security considerations

When the rs_send_repserver_cmd procedure is invoked at the primary data server, Replication Agent passes corresponding Replication Command Language (RCL) directly to Replication Server. You should therefore consider carefully to whom execution privileges are assigned for the rs_send_repserver_cmd procedure, and assign privileges as appropriate for your environment and security policy.

Limitations

The rs_send_repserver_cmd procedure cannot be used to alter replication definitions for tables that contain columns of the following types:

- nvarchar(max)
- varbinary(max)
- varchar(max)

Upgrading the Microsoft SQL Server service pack

Log locator values and some Replication Agent transaction log objects are lost when updating the Microsoft SQL Server service pack. After upgrading the service pack and before performing migration tasks on Replication Agent, reset the locator values at Replication Agent and Replication Server.

❖ **Upgrading the Microsoft SQL Server service pack**

1 Make sure that all data has been completely replicated.

2 Upgrade the Microsoft SQL Server service pack.

3 Set the value of the LTM Locator stored in the Replication Agent transaction log to zero:

```
ra_locator zero
```

4 Reset the locator value to zero for the primary database:

```
rs_zeroltm data_server, database
```

where *data_server* is the data server on which the primary database resides and *database* is the primary database.

5 Migrate Replication Agent:

```
ra_migrate
```

Replication Server *set autocorrection* command

The Replication Server set autocorrection command prevents failures that would otherwise be caused by missing or duplicate rows in a replicated table. The set autocorrection command corrects discrepancies that may occur during materialization by converting each update or insert operation into a delete followed by an insert.

You can set autocorrection from Replication Agent for one or all marked tables in the primary database by using the ra_set_autocorrection command as described in the *Replication Agent Reference Manual*. To set autocorrection from Replication Server, use the set autocorrection command in a replication definition. You must do this from Replication Server because Replication Agent cannot alter the autocorrection setting on a replication definition. See the *Replication Server Administration Guide* for more information.

Computed columns

Tables containing computed columns that are physically stored in a table–columns marked as PERSISTED in Microsoft SQL Server–can be marked for replication, and these columns will be replicated. Tables containing virtual computed columns–computed columns that are not physically stored in a table–can be marked for replication, but these columns will not be replicated. To maintain consistency between the primary and replicate databases for a marked table containing a virtual computed column, make sure that the expression defining the virtual computed column is the same in both the primary and replicate databases.

Microsoft SQL Server 2008

Replication Agent does not support some functionality for Microsoft SQL Server 2008.

Unsupported Microsoft SQL Server 2008 datatypes

Replication Agent does not support the following Microsoft SQL Server 2008 datatypes:

- date
- datetime2
- datetimeoffset
- filestream
- geography
- geometry
- hierarchyid
- time
- large user-defined datatypes

A table containing columns of these types cannot be marked, even by using pdb_setreptable with the force keyword.

Unsupported Microsoft SQL Server 2008 features

Replication Agent does not support the following Microsoft SQL Server 2008 features:

- column sets
- MERGE SQL statements
- procedures with table-valued parameters
- sparse columns
- transparent data encryption (TDE)

A table or stored procedure that uses these features cannot be marked, even by using pdb_setreptable with the force keyword.

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_5\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 nonalphanumeric 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 nonalphanumeric characters allowed in a database object name are the $, #, @, and _ characters.

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 2-4 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 2-4: Replication Agent table objects

Object	Name
DDL shadow table	*prefix*ddl_trig_*xxx*
Object marking table	*prefix*markObject_*xxx*
Object verifying table	*prefix*checkObject_*xxx*

Microsoft SQL Server system tables

At initialization, Replication Agent creates some system tables in the primary database using the Microsoft SQL Server sp_replicationdboption stored procedure. These tables are also removed with the sp_replicationdboption stored procedure when the Replication Agent pdb_xlog remove command is used. Do not modify these tables directly. For more information about the sp_replicationdboption stored procedure, see the Microsoft SQL Server documentation.

Procedure objects

Table 2-5 lists the procedure objects that are considered Replication Agent objects. The sp_SybSetLogforReplTable, sp_SybSetLogforReplProc, and sp_SybSetLogforLOBCol procedures are created in the Microsoft SQL Server mssqlsystemresource system database. Although execute permission on these procedures is granted to Public, only the 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 2-5 have no effect when executed outside the context of replication.

Table 2-5: Replication Agent procedure objects

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

Marker objects

Table 2-6 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 2-6: Replication Agent marker objects

Object	Name
Transaction log marker procedure	rs_marker_*xxx*
Dump marker procedure	rs_dump_*xxx*
Transaction log marker shadow table	*prefix*markersh_*xxx*
Dump marker shadow table	*prefix*dumpsh_*xxx*

Trigger objects

Table 2-7 lists Replication Agent trigger objects.

Table 2-7: Replication Agent trigger objects

Object	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.

❖ **Using 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, see 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.

❖ Setting 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.

CHAPTER 3 Replication Agent for UDB

The term "Replication Agent for UDB" refers to an instance of Replication Agent software that is configured for a primary database that resides in a UDB server.

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

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

UDB-specific considerations

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

- Unsupported software features
- Unsupported datatypes
- Feature differences in Replication Agent for UDB
- UDB Requirements
- Running Replication Agent on a different machine
- 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
- Replication Server set autocorrection command
- Large identifiers
- Value compression

Unsupported software features

The following features are not supported:

- UDB data definition language (DDL) commands
- UDB stored procedures
- UDB row compression
- Replication Server parallel DSI
- Replication Server rs_init utility
- Replication Server rs_subcomp utility
- Replication Server automatic materialization
- Replication Server when replicating in an environment where other vendors are replicating

Unsupported datatypes

The following datatypes are not supported:

- ROWID
- XML
- User-defined datatypes

The following datatypes are not supported when the replicate database is UDB:

- BLOB
- CLOB
- DBCLOB
- LONG VARCHAR

- LONG VARGRAPHIC

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 UDB 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.

See "Replication Agent objects in the UDB primary database" on page 125.

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.

Marking or unmarking all tables simultaneously

Marking or unmarking all tables at once in the primary database using pdb_setreptable all, mark or pdb_setreptable all, unmark is not supported in Replication Agent for UDB. You must mark or unmark each table individually.

Unavailable features

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

- Stored procedure replication–The replication of stored procedures is not available with the Replication Agent for UDB. Therefore, the pdb_setrepproc command is not supported.
- DDL replication–The replication of data definition language (DDL) commands and system procedures executed in the primary database is not supported.
- Altering replication definitions from the primary data server–This involves stored procedure replication, which is not supported.
- Automatically creating replication definitions–This requires use of the Replication Agent rs_create_repdef command, which is not available for Replication Agent for UDB.

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

UDB Requirements

This section provides a summary of all the UDB 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. The JDK_PATH configuration parameter should contain the full path to the directory above the *bin* directory, which contains the *java* executable. To determine the database manager JDK_PATH setting, use the following UDB command:

  ```
  get dbm cfg
  ```

  Note A 64-bit IBM UDB instance requires a 64-bit JDK, and a 32-bit UDB instance requires a 32-bit JDK.

- If Replication Agent is installed on Solaris, AIX or HP Itanium, a 64-bit UDB instance must be configured. This can be a server or client instance.

- 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>
  ```

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

  %DB2DIR%\sqllib\db2cli.ini

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

- To initialize Replication Agent without error, the database must have a table space created with the following characteristics:

  - The table space should be a user temporary table space. By default, user temporary table spaces are not created when a database is created.
  - The table space must be a system-managed space (SMS).
  - The PAGESIZE parameter must be set to 8192 (8 kilobytes) or greater.

- 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 UDB CLI driver to connect to the primary UDB database. For UNIX, the driver is contained in *libdb2.so*, *libdb2.sl*, or *libdb2.a*, depending on the operating system. For Windows, the UDB driver is contained in *db2cli.dll*. Replication Agent also uses UDB API libraries to read the transaction log. The library path environment variable must therefore be set for Replication Agent to load the correct driver and API libraries at runtime.

For UNIX and Linux, the 32-bit and 64-bit versions of the libraries are located in the *$HOME/sqllib/lib32* and *$HOME/sqllib/lib64* directories, respectively, where *$HOME* is the home directory of the UDB instance owner. If Replication Agent is installed on AIX, Solaris, or HP Itanium, the library path environment variable must point to the 64-bit libraries. For Windows and Linux, the library path environment variable must point to the 32-bit libraries.

The exact name of the library path environment variable depends on the operating system. For Linux, the library path variable is named LD_LIBRARY_PATH. For Windows, the library path variable is named PATH.

On Windows, the UDB server or client installation sets all necessary environment variables. 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 at *$HOME/sqllib*, where *$HOME* is the home directory of the UDB instance owner (for a UDB client or server instance).

Running Replication Agent on a different machine

If the Replication Agent for UDB software is installed on a different host machine from the UDB server, you must install the UDB 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 UDB server, a separate UDB Administration Client is not required.

If the Replication Agent for UDB software is installed on AIX, Solaris, or HP Itanium, a 64-bit UDB client instance must be configured. On Windows and Linux, a 32-bit UDB client instance may be configured.

Configuring UDB connectivity

On a Windows system, you must configure an ODBC data source in the UDB 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. Set the Replication Agent pds_datasource_name parameter to the database alias.

❖ **Cataloging the remote TCP/IP node from the UDB client**

1 Log in as the UDB instance owner.

Logging in sets up your UDB environment variables by executing the environment scripts. You can also execute these scripts manually as follows.

In Korn shell, source the *db2profile* file:

```
. $HOME/sqllib/db2profile
```

In C shell, source the *db2cshrc* file:

```
source $HOME/sqllib/db2cshrc
```

Here, *$HOME* is the home directory of the UDB instance owner.

2 Start the UDB command-line processor by typing the db2 command.

3 Catalog the remote TCP/IP node using the following command at the UDB 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.

4 Verify the catalog entry:

```
list node directory
```

UDB 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
```

❖ **Cataloging the primary database from the UDB client**

1 Catalog the primary database using the following command at the UDB 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 tcpip node command.

2 Verify the catalog entry:

```
list database directory
```

UDB 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
```

❖ **Configuring 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

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 UDB server:

- pds_username – must have DBADM authority, for example, repuser

- pds_password – for user ID specified in pds_username, for example, repuser_pwd
- 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 UDB 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 UDB transaction log.

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

- 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.
- If you mark a table for replication, insert data into the table, and then resume replication, the data will not be replicated if the LTM Locators for Replication Agent and Replication Server are zero (as they would be at the beginning of replication). This problem occurs because when both LTM Locators are zero, resuming replication repositions the Log Reader component at the end of the log, skipping over any previous transactions. To avoid this problem when the LTM Locators for Replication Agent and Replication Server are zero, mark the table for replication after you have issued the resume command.

- 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 UDB 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 UDB 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 UDB 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 misses the mark table entry, and table marking fails.

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 UDB 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 UDB 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 UDB FORCE APPLICATION command.

Determining read buffer size

The Replication Agent for UDB LogReader component uses the value of the lr_read_buffer_size 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.

It is very difficult to identify a minimum buffer size that will always work. The value range of lr_read_buffer_size is 10000 to 2147483647.

If the read buffer size is too small to read one operation, Replication Agent goes into the *Admin* state, and the Log Reader component shuts down and reports a -2650 error. Unfortunately, this error message covers general communication errors, not just an insufficient buffer size.

Replicating LOBs

When replication is enabled for a LOB column, Replication Agent makes an entry in the prefixBLOB_COLUMNS_ table to support replication for that column.

When Replication Agent processes a transaction that affects a LOB column, the LOB data may not be stored in the transaction log because of its possible size. Instead, the Replication Agent Log Reader component reads the LOB data directly from the primary database at the time it processes the transaction.

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.

For instructions on enabling and disabling replication for LOB columns, see the *Replication Agent Administration Guide*.

Compromising transaction integrity

Because of the way Replication Agent processes the LOB column data when replicating transactions, it is possible to compromise transaction integrity. For example, if two transactions change the data in a LOB column and the Log Reader does not process the first transaction until after the second transaction has been committed, when the LOB data is read from the primary database, the value of that data is the result of the second transaction. In this event, the value of the LOB data in the first transaction is never sent to the replicate database. After the second transaction is processed by the Log Reader, the primary and replicate databases are synchronized again, but for a period of time between processing the first and second transactions, the replicate database contains data that does not match the originating transaction.

This problem occurs only when a LOB column is changed more than once by a sequence of transactions. The period of time over which the problem exists could be significant if the replication system throughput is slow or if a replication system component fails. As soon as the last transaction that changes the LOB column is processed at the replicate site, the problem is corrected.

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 fails. 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 UDB 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 UDB 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: UDB to Sybase default datatype mapping

UDB datatype	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	
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.
DECFLOAT(16)	8 bytes, -9.999999999999999 x 10^{384} to -1.0 x 10^{-383} and 1.0 x 10^{-383} to 9.999999999999999 x 10^{384}	float	Precision and range corresponds to a C double datatype, approximately 16 significant digits	Some loss of precision may result.
DECFLOAT(34)	16 bytes, -9.9999999999999999 9999999999999999 x 10^{6144} to -1.0 x 10^{-6143} and 1.0 x 10^{-6143} to 9.9999999999999999 9999999999999999 x 10^{6144}	float	Precision and range corresponds to a C double datatype, approximately 16 significant digits	Some loss of precision may result.

UDB datatype	UDB length/range	Sybase datatype	Sybase length/range	Notes
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.
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	

UDB datatype	UDB length/range	Sybase datatype	Sybase length/range	Notes
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

DECFLOAT datatypes

Replication Agent for UDB supports the replication of the DECFLOAT datatype. Both 16 and 34 digits of precision are supported.

When Replication Agent for UDB replicates DECFLOAT columns from the UDB primary to replicate databases that do not support the DECFLOAT type or an equivalent type, Replication Agent maps the DECFLOAT type to the FLOAT type. Consequently, some loss of precision may result.

In addition to a number value, a DECFLOAT column may contain special values that are not supported by Replication Agent, such as positive and negative INFINITY, NAN, and SNAN. Replication of these values is not supported. In these cases, Replication Agent will replicate special values to NULL, if the column is nullable, or to '0.0' if the column is not nullable.

XML datatype

Replication of the XML datatype is not supported by Replication Agent for UDB. If you attempt to mark a table that has an XML column, Replication Agent will report an error. You can mark a table containing an XML column using the force option of the pdb_setreptable command, but this column will not be replicated.

Replication Server *set autocorrection* command

The Replication Server set autocorrection command prevents failures that would otherwise be caused by missing or duplicate rows in a replicated table. The set autocorrection command corrects discrepancies that may occur during materialization by converting each update or insert operation into a delete followed by an insert.

You can set autocorrection from Replication Agent for one or all marked tables in the primary database by using the ra_set_autocorrection command as described in the *Replication Agent Reference Manual*. To set autocorrection from Replication Server, use the set autocorrection command in a replication definition. You must do this from Replication Server because Replication Agent cannot alter the autocorrection setting on a replication definition. See the *Replication Server Administration Guide* for more information.

Large identifiers

Replication Agent for UDB supports UDB 9.5 large identifiers–authorization ID, column, and schema names up to 128 bytes long.

To support UDB 9.5 large identifiers, you must migrate Replication Agent instances from versions 15.0, 15.1, and 15.2 to 15.5, to accommodate the modified Replication Agent system tables. You must also migrate Replication Agent instances when UDB is upgraded from an earlier version to 9.5 to replicate tables with large identifiers. See “Upgrading Replication Agent for UDB” on page 153 for upgrade procedures.

Value compression

Replication Agent for UDB supports value compression–tables created with the VALUE COMPRESSION clause–but not row compression.

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 UDB 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 UDB 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_5\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 JAR files from the primary database. You must issue pdb_xlog remove command before reinitializing Replication Agent. See “Java procedure objects” on page 127.

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 system 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.

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.

Note If more than one Replication Agent instance is configured for a UDB server–one Replication Agent instance for each database–a unique primary database user name must be specified in the pds_username configuration parameter for each Replication Agent instance. This is required to install and uninstall these .jar files.

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.

❖ Finding 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
- Autocorrection flag

Administering the transaction log

The Replication Agent for UDB supports truncating transaction logs. All UDB 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 By default, the pdb_archive_remove property is set to false. 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. All archive log files up to but not including the file with the oldest active LSN are deleted.

For more information on these properties see the *Replication Agent Reference Manual*. For a more detailed description of truncating, see “Chapter 3, Administering Replication Agent” in the *Replication Agent Administration Guide*. For a list of the stored procedures used for truncation, see Table 3-5 on page 128.

APPENDIX A

Upgrading and Downgrading Replication Agent

Warning! If you upgrade Replication Agent to the most current version, attempt to use features new to the current release of Replication Agent, and then subsequently downgrade to a previous version, the version to which you downgrade may not function properly. For this reason, you should make a backup copy of an instance of the Replication Agent version to which you have upgraded before attempting to use any new feature.

Upgrading Replication Agent for Oracle

Replication Agent for Oracle 15.5 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 15.5 instances will have the same configuration as previously existing instances, including instance names, administrative user IDs and passwords, and administrative port numbers.

- Upgrading Replication Agent 15.1 or 15.2 to 15.5

- Migrating Replication Agent 15.5 when upgrading Oracle 10g to 11g

---

Note For upgrades within a common release level, as in the case of an ESD applied to a particular version of Replication Agent, you should use the ra_admin -u option applied to a particular instance of Replication Agent or to all instances of Replication Agent. See the *Replication Agent Administration Guide*.

Upgrading Replication Agent 15.1 or 15.2 to 15.5

This section describes how to upgrade Replication Agent for Oracle 15.1 or 15.2 to 15.5.

Note Replication Agent 15.5 must be installed on the same host on which the primary Oracle server is running.

❖ Upgrading Replication Agent 15.1 or 15.2 to 15.5

1. For each existing Replication Agent for Oracle instance, Sybase recommends that you first back up the RASD as described in the *Replication Agent Administration Guide*. Back up the complete existing Replication Agent instance directory.
2. Install the Replication Agent 15.5 software as described in "Installing the Replication Agent software" in the *Replication Agent Installation Guide*. Sybase recommends that you install Replication Agent 15.5 into the same *SYBASE* directory as the earlier version of Replication Agent.
3. Download and install the Oracle JDBC driver, and set the CLASSPATH environment variable, as described in the *Replication Agent Installation Guide*. If the CLASSPATH contains another Oracle JDBC driver, remove it. Only the Oracle JDBC driver required by Replication Agent 15.5 should be in the CLASSPATH.

4 Create the 15.5 version of all valid existing Replication Agent instances.

Note This step creates new Replication Agent 15.5 instances for all valid existing instances of the earlier version of Replication Agent, regardless of whether the existing instances are for Oracle, Microsoft SQL Server, or UDB. 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 to the *SYBASE* directory in which Replication Agent 15.5 is installed and sourcing the *SYBASE* script:

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

b Change to the Replication Agent 15.5 *bin* directory:

- On UNIX:

```
cd $SYBASE/RAX-15_5/bin
```

- On Windows:

```
cd %SYBASE%\RAX-15_5\bin
```

c Create new versions of all valid existing instances:

```
ra_admin -u src_directory
```

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

```
ra_admin -u d:\sybase\RAX-15_1
```

For information about the instances that did not upgrade successfully, see the administration logs (*…/RAX-15_5/admin_logs*). After you correct the problem, re-run this command. This command does not affect 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 15.5 *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*.

6 If necessary, override the default maximum amount of memory available to the JRE by setting the RA_JAVA_MAX_MEM environment variable in the Replication Agent 15.5 *RUN_instance* script. Replication Agent 15.5 does not set the RA_JAVA_MAX_MEM environment variable in the executable or run scripts, which allows the JVM to use its default for the maximum heap size. See the *Replication Agent Administration Guide*.

7 In the primary Oracle database, grant the previously existing pds_username user any additional required privileges. See "Replication Agent permissions and roles" on page 83.

8 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.

9 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):

      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.

10 Start and log in to the Replication Agent 15.5 instance and migrate the Replication Agent metadata by issuing the ra_migrate command.

11 Allow all users to access the primary Oracle.

12 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 15.5, requiring the previous value held by Replication Server to be replaced. The rs_source_ds and rs_source_db values are migrated from the earlier version of Replication Agent and do not need to be changed.

---

13 In the Replication Agent 15.5 instance, do the following:

- To have automatic archiving turned off, set pdb_include_archives to false. 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. See "Redo and archive log setup" on page 9.
- Resume replication by issuing the resume command.

Migrating Replication Agent 15.5 when upgrading Oracle 10g to 11g

Replication Agent for Oracle migration to support upgrading Oracle 10g to Oracle 11g is similar to upgrading Replication Agent for Oracle 15.1 or 15.2 to Replication Agent for Oracle 15.5.

Note Quiesce the Replication Agent before upgrading Oracle 10g to Oracle 11g. The replication environment must have completed processing of all transactions before upgrading Oracle because the Replication Agent moves the truncation point to the end of the log during Replication Agent migration.

❖ Migrating from Oracle 10g to Oracle 11g

1 Follow the steps that Oracle provides in their documentation for upgrading from Oracle 10g to Oracle 11g.

2 After upgrading Oracle, re-start the Replication Agent and issue the ra_migrate command.

3 As with the log-based Replication Agent upgrade process, you may need to reconfigure 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 10g to Oracle 11g at the same time, migrate Replication Agent 15.5 only once.

Upgrading Replication Agent for Microsoft SQL Server

Replication Agent for Microsoft SQL Server must be installed on the same Windows host on which the primary Microsoft SQL Server is running, and Replication Agent for Microsoft SQL Server cannot be installed on a UNIX or Linux host. Before upgrading, you must therefore consider where the existing instance of the earlier version of Replication Agent is installed and 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 instances will have the same configuration as previously existing instances, including instance names, administrative user IDs and passwords, and administrative port numbers.

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

This section describes how to upgrade Replication Agent 15.0 to 15.5 in the following specific situations:

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

- Replication Agent 15.5 is installed on the same Windows host as the earlier version of Replication Agent.
- Replication Agent 15.5 is installed on a different Windows host than the earlier version of Replication Agent.
- Replication Agent 15.5 is installed on a Windows host, but the earlier version of Replication Agent is installed on a UNIX or Linux host.

Note Replication Agent 15.5 must be installed on the same host on which the primary Microsoft SQL Server is running.

Replication Agent 15.5 is installed on the same Windows host as the earlier version, and the current version of Microsoft SQL Server is 2005

❖ Upgrading when the previous Replication Agent version and Replication Agent 15.5 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 15.5 software as described in “Installing the Replication Agent software” in the *Replication Agent Installation Guide*. Sybase recommends that you install Replication Agent 15.5 into the same SYBASE directory as the earlier 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 84.
4. Verify that your primary data server meets the requirements described in “Microsoft SQL Server requirements” on page 77.
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. Determine the primary Microsoft SQL Server DAC port number, and make sure Microsoft SQL Server is configured to allow a remote DAC:

   ```
   sp_configure 'remote admin connections', 1;
   GO
   RECONFIGURE;
   GO
   ```

   To execute sp_configure with both parameters to change a configuration option or to run the RECONFIGURE statement, you must be granted the ALTER SETTINGS server-level permission. The ALTER SETTINGS permission is implicitly held by the sysadmin and serveradmin fixed server roles.
7. In the primary Microsoft SQL Server, grant each previously existing pds_username user the additional required privileges. See “Replication Agent permissions and roles” on page 83.

8 Use the Replication Agent 15.5 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 is done in a later step in this procedure.

9 Create the 15.5 version of all valid existing Replication Agent instances:

Note This step creates new Replication Agent 15.5 instances for all valid existing instances of the earlier version of Replication Agent, regardless of whether the existing instances are for Oracle, Microsoft SQL Server, or UDB. 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 to the Replication Agent 15.5 *bin* directory:

```
cd %SYBASE%\RAX-15_5\bin
```

c Create new versions of all valid existing instances:

```
ra_admin -u src_directory
```

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

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

For information about instances that did not upgrade successfully, see the administration logs (*...\RAX-15_5\admin_logs*). After you correct the problem, re-run this command. Be aware that this command does not affect 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 15.5 *RUN_instance* scripts 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*.

11 If necessary, override the default maximum amount of memory available to the JRE by setting the RA_JAVA_MAX_MEM environment variable in the Replication Agent 15.5 *RUN_instance* script. Replication Agent 15.5 does not set the RA_JAVA_MAX_MEM environment variable in the executable or run scripts, which allows the JVM to use its default for the maximum heap size. See the *Replication Agent Administration Guide*.

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:

   ```
   2008-11-09 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 is used in a later step in this procedure.

13 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.

14 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.

15 When all 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.

16 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.

17 Download and install the Microsoft SQL Server JDBC driver, and set the CLASSPATH environment variable, as described in the *Replication Agent Installation Guide*. If the CLASSPATH contains any other Microsoft SQL Server JDBC driver, remove it. Only the Microsoft SQL Server JDBC driver required by Replication Agent 15.5 can be in the CLASSPATH.

18 Start and log in to each of the Replication Agent for Microsoft SQL Server 15.5 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 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 15.5 instance, it will also initialize the Microsoft SQL Server. In subsequent Replication Agent 15.5 instances, it will only initialize the instance and migrate the instance's metadata.

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

   a 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.

20 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.

21 Log in to each of the Replication Agent Microsoft SQL Server 15.5 instances and resume replication by issuing the resume command.

22 Allow all users to access the primary databases.

Replication Agent 15.5 is installed on a different Windows host than the earlier version of Replication Agent, and the current version of Microsoft SQL Server is 2005

❖ Upgrading when Replication Agent 15.5 is on a different Windows host than earlier 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 earlier version of Replication Agent available as the source directory for the upgrade procedure:

- On the Windows host on which the earlier version of Replication Agent is running, share the drive on which the Replication Agent is installed.
- On the Windows host on which Replication Agent 15.5 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 earlier version of Replication Agent as the source directory for the upgrade.

3 Starting with step 3 of the procedure “Replication Agent 15.5 is installed on the same Windows host as the earlier version, and the current version of Microsoft SQL Server is 2005” on page 137, follow the steps to complete the upgrade.

Replication Agent 15.5 is installed on a Windows host, but the earlier version is installed on a UNIX or Linux host, and the current version of Microsoft SQL Server is 2005

❖ Upgrading when Replication Agent 15.5 is on Windows, but the earlier 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 earlier 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 15.5, create a directory with the name of the earlier version of Replication Agent that you are upgrading from. 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. Within each of these instance subdirectories that you just created, create two subdirectories named *log* and *scripts*.

   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 15.5 is installed on the same Windows host as the earlier version, and the current version of Microsoft SQL Server is 2005” on page 137, follow the steps to complete the upgrade.

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

Note Replication Agent 15.5 requires that the primary Microsoft SQL Server be version 2005 SP2.

This section describes how to simultaneously upgrade Replication Agent 15.0 to 15.5 and upgrade Microsoft SQL Server 7 or 2000 to 2005 in the following specific situations:

- Replication Agent 15.5 is installed on the same Windows host as the earlier version of Replication Agent
- Replication Agent 15.5 is installed on a different Windows host than the earlier version of Replication Agent
- Replication Agent 15.5 is installed on a Windows host, but the earlier version of Replication Agent is installed on a UNIX or Linux host

Note Replication Agent 15.5 must be installed on the same host on which the primary Microsoft SQL Server is running.

Replication Agent 15.5 is installed on the same Windows host as the earlier version of Replication Agent, and the current version of Microsoft SQL Server is 7 or 2000

❖ Installing Replication Agent 15.5 on the same Windows host as the earlier 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 15.5 into the same SYBASE directory as the earlier 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 84.
4. Create the 15.5 version of all valid existing Replication Agent instances.

   Note This step creates new Replication Agent 15.5 instances for all valid existing instances of the earlier version of Replication Agent, regardless of whether the existing instances are for Oracle, Microsoft SQL Server, or UDB. 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 to the Replication Agent 15.5 *bin* directory:

      ```
      cd %SYBASE%\RAX-15_5\bin
      ```

   c. Create new versions of all valid existing instances:

      ```
      ra_admin -u src_directory
      ```

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

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

      For information about the instances that did not upgrade successfully, see the administration logs (*…\RAX-15_5\admin_logs*). After you correct the problem, re-run this command. This command does not affect 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 15.5 *RUN_instance* scripts 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*.

6 If necessary, override the default maximum amount of memory available to the JRE by setting the RA_JAVA_MAX_MEM environment variable in the Replication Agent 15.5 *RUN_instance* script. Replication Agent 15.5 does not set the RA_JAVA_MAX_MEM environment variable in the executable or run scripts, which allows the JVM to use its default for the maximum heap size. See the *Replication Agent Administration Guide*.

7 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.

8 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.

9 When all 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 77.

10 In the primary Microsoft SQL Server, grant each previously existing pds_username user the additional required privileges. See “Replication Agent permissions and roles” on page 83.

11 Download and install the Microsoft SQL Server JDBC driver, and set the CLASSPATH environment variable, as described in the *Replication Agent Installation Guide*. If the CLASSPATH contains any other Microsoft SQL Server JDBC driver, remove it. Only the Microsoft SQL Server JDBC driver required by Replication Agent 15.5 should be in the CLASSPATH.

12 Use the Replication Agent 15.5 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 is done in a later step in this procedure.

13 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:

   ```
   2008-11-09 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 is used in a later step in this procedure.

14 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.

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. Start and log in to each of the Replication Agent for Microsoft SQL Server 15.5 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 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 15.5 instance, it will also initialize the Microsoft SQL Server. In subsequent Replication Agent 15.5 instances, it will only initialize the instance and migrate the instance's metadata.

17. Stop the Microsoft SQL Server in single-user mode:

    a. 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.

18. 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.

19 Log in to each of the Replication Agent Microsoft SQL Server 15.5 instances and resume replication:

a 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.

20 Allow all users to access the primary databases.

Replication Agent 15.5 is installed on a different Windows host than the earlier version, and the current version of Microsoft SQL Server is 7 or 2000

❖ Upgrading when Replication Agent 15.5 is on a different host than a earlier 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 earlier version of Replication Agent available as the source directory for the upgrade procedure:

a On the Windows host on which the earlier 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 15.5 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 earlier version of Replication Agent as the source directory for the upgrade.

3 Starting with step 3 in the procedure called “Replication Agent 15.5 is installed on the same Windows host as the earlier version of Replication Agent, and the current version of Microsoft SQL Server is 7 or 2000” on page 144, follow the steps to complete the upgrade.

Replication Agent 15.5 is installed on a Windows host but the earlier version of Replication Agent is installed on a UNIX or Linux host, and the current version of Microsoft SQL Server is 7 or 2000

❖ Upgrading when Replication Agent 15.5 is on Windows, but the earlier 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 earlier 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 15.5, create a directory with the name of the earlier version of Replication Agent that you are upgrading from. 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. Within each of these instance subdirectories that you just created, create two subdirectories named *log* and *scripts*.
   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 of the procedure called “Replication Agent 15.5 is installed on the same Windows host as the earlier version of Replication Agent, and the current version of Microsoft SQL Server is 7 or 2000” on page 144, follow the steps to complete the upgrade.

Upgrading Replication Agent 15.1 or 15.2 to 15.5

This section describes how upgrade Replication Agent 15.1 or 15.2 to 15.5.

Note Replication Agent 15.5 must be installed on the same host on which the primary Microsoft SQL Server is running.

❖ Upgrading Replication Agent 15.1 or 15.2 to 15.5

1. For each existing Replication Agent for Microsoft SQL Server instance, Sybase recommends that you first back up the Replication Agent System Database (RASD) as described in the *Replication Agent Administration Guide*. Back up the complete existing Replication Agent instance directory.
2. Install the Replication Agent 15.5 software as described in "Installing the Replication Agent software" in the *Replication Agent Installation Guide*. Sybase recommends that you install Replication Agent 15.5 into the same *SYBASE* directory as the earlier version of Replication Agent.
3. Use the Replication Agent 15.5 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 is done in a later step in this procedure.
4. Create the 15.5 version of all valid existing Replication Agent instances.

   Note This step creates new Replication Agent 15.5 instances for all valid existing instances of the earlier version of Replication Agent, regardless of whether the existing instances are for Oracle, Microsoft SQL Server, or UDB. 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. Set the SYBASE environment variables by changing to the *SYBASE* directory in which Replication Agent 15.5 is installed and executing the *SYBASE.bat* script.
   c. Change to the Replication Agent 15.5 *bin* directory:

      ```
      cd %SYBASE%\RAX-15_5\bin
      ```

d Create new versions of all valid existing instances:

```
ra_admin -u src_directory
```

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

```
ra_admin -u d:\sybase\RAX-15_1
```

For information about the instances that did not upgrade successfully, see the administration logs (*…\RAX-15_5\admin_logs*). After you correct the problem, re-run this command. This command does not affect 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 15.5 *RUN_instance* scripts 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*.

6 If necessary, override the default maximum amount of memory available to the JRE by setting the RA_JAVA_MAX_MEM environment variable in the Replication Agent 15.5 *RUN_instance* script. Replication Agent 15.5 does not set the RA_JAVA_MAX_MEM environment variable in the executable or run scripts, which allows the JVM to use its default for the maximum heap size. See the *Replication Agent Administration Guide*.

7 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.

8 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.

9 When all 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.

10 Download and install the Microsoft SQL Server JDBC driver, and set the CLASSPATH environment variable, as described in the *Replication Agent Installation Guide*. If the CLASSPATH contains any other Microsoft SQL Server JDBC driver, remove it. Only the Microsoft SQL Server JDBC driver required by Replication Agent 15.5 should be in the CLASSPATH.

11 Start and log in to each of the Replication Agent for Microsoft SQL Server 15.5 instances and do the following:

- a Set the rs_charset configuration parameter to match the Replication Server character set, as described in the *Replication Agent Reference Manual*.
- b Use the test_connection command to ensure that Replication Agent can connect to both Microsoft SQL Server and Replication Server.
- c Initialize the Replication Agent instance and migrate the Replication Agent instance metadata by issuing the ra_migrate command.

  When this command executes in the first Replication Agent 15.5 instance, it also initializes the Microsoft SQL Server. In subsequent Replication Agent 15.5 instances, it only initializes the instance and migrates the instance metadata.

12 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.

13. Log in to each of the Replication Agent Microsoft SQL Server 15.5 instances and resume replication:

    a. 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.

14. Allow all users to access the primary databases.

Upgrading Replication Agent for UDB

Replication Agent for UDB 15.5 provides automatic upgrade of Replication Agent for UDB version 15.0 and later instances, and automatic migration of a Replication Agent for UDB instance when you upgrade the UDB from version 8.2 or 9.1 to version 9.5.

When you use any of the upgrade procedures described in this section, the new Replication Agent for UDB 15.5 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 15.5 does not support:

- Upgrading Replication Agent for UDB version 12.6 or earlier to version 15.0 or later.
- Migrating Replication Agent for UDB 12.6 when UDB 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 or later to 15.5
- Migrating Replication Agent when UDB is upgraded from version 8.2 or 9.1 to version 9.5

Upgrading from Replication Agent for UDB 15.0 or later to 15.5

This section describes how to upgrade Replication Agent from version 15.0 or later to the current version.

❖ Upgrading from Replication Agent for UDB 15.0 or later to 15.5

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 15.5 software as described in "Installing the Replication Agent software" in the *Replication Agent Installation Guide*. Sybase recommends that you install Replication Agent 15.5 into the same *SYBASE* directory as the earlier version of Replication Agent.
3. Create the 15.5 version of all valid existing Replication Agent instances.

   Note This step creates new Replication Agent 15.5 instances for all valid existing instances of the earlier version of Replication Agent, regardless of whether the existing instances are for Oracle, Microsoft SQL Server, or UDB. To complete the upgrade for Microsoft SQL Server or Oracle 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 to the *SYBASE* directory in which Replication Agent 15.5 is installed and sourcing the *SYBASE* script:

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

   b. Change to the Replication Agent 15.5 *bin* directory:

      - On UNIX:

        ```
        cd $SYBASE/RAX-15_5/bin
        ```

      - On Windows:

        ```
        cd %SYBASE%\RAX-15_5\bin
        ```

   c. Create new versions of all valid existing instances:

      ```
      ra_admin -u src_directory
      ```

Here, *src_directory* is the full path name of the earlier version's Replication Agent 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 15.5 *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*.

5 If necessary, override the default maximum amount of memory available to the JRE by setting the RA_JAVA_MAX_MEM environment variable in the Replication Agent 15.5 *RUN_instance* script. Replication Agent 15.5 does not set the RA_JAVA_MAX_MEM environment variable in the executable or run scripts, which allows the JVM to use its default for the maximum heap size. See the *Replication Agent Administration Guide*.

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

7 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.

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):

      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 15.5 instance, and migrate the Replication Agent's metadata by issuing the ra_migrate command.
10. In the Replication Agent 15.5 instance, resume replication by issuing the resume command.
11. Allow all users to access the primary UDB database.

Migrating Replication Agent when UDB is upgraded from version 8.2 or 9.1 to version 9.5

This section describes how to migrate Replication Agent from version 15.0 or later to the current version while upgrading UDB from version 8.2 or 9.1 to version 9.5.

❖ **Migrating Replication Agent when upgrading UDB**

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 15.5 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):

      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 UDB documentation for upgrading UDB.
4. Verify that all the primary database requirements are met as described in "UDB Requirements" on page 110.

   Note If the use_rssd configuration parameter was set to true before migration, skip the following step.

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.

   Note If the use_rssd configuration parameter was set to true before migration, skip the following step.

7. In the Replication Agent 15.5 instance, resume replication by issuing the resume command.
8. Allow all users to access the primary UDB database.

Note If you are upgrading Replication Agent and upgrading UDB from version 8.2 or 9.1 to version 9.5 at the same time, you need to migrate Replication Agent only once.

Downgrading Replication Agent for Oracle

Replication Agent for Oracle can be downgraded from version 15.5 to version 15.2 ESD #2. Downgrading may be necessary if the upgrade process fails. A downgrade may also be necessary if replication fails after the upgrade process has completed, if there are changes to DDL and how DDL is handled by Replication Agent, or if there are changes to the content or structure of the Replication Agent System Database (RASD).

If you are using any new features from Replication Agent 15.5, downgrading is not supported. For a list of the new features for Replication Agent 15.5, see the *Replication Agent New Features*.

Downgrading Replication Agent 15.5 to 15.2 ESD #2

This section describes how to downgrade Replication Agent 15.5 to 15.2 ESD #2.

Note Replication Agent 15.5 must be installed on the same platform on which the primary Oracle server is running.

❖ **Downgrading Replication Agent 15.5 to 15.2 ESD #2**

1. Change to the Replication Agent 15.5 *bin* directory:

   - On UNIX:

     ```
     cd $SYBASE/RAX-15_5/bin
     ```

   - On Windows:

     ```
     cd %SYBASE%\RAX-15_5\bin
     ```

2. Prepare for downgrade by running the ra_downgrade_prepare command at the Replication Agent instance from which you are downgrading (the current version):

   ```
   ra_downgrade_prepare inst_path
   ```

   Here, *inst_path* is the absolute path of the Replication Agent instance to which you are downgrading (the earlier version).

The ra_downgrade_prepare command extracts the contents of the Replication Agent System Database (RASD) to a file named *timestamp.export*, where *timestamp* is a timestamp taken at the moment ra_downgrade_prepare was invoked. This file is located in the *import* subdirectory under the directory specified by the rasd_backup_dir configuration parameter of the Replication Agent instance to which you are downgrading (the earlier version). The path to this file is returned if ra_downgrade_prepare executes successfully.

3 Complete the downgrade by running the ra_downgrade_accept command at the Replication Agent instance to which you are downgrading (the earlier version):

```
ra_downgrade_accept timestamp.export
```

Here, *timestamp.export* is the file to which the ra_downgrade_prepare command extracted RASD contents.

If the ra_downgrade_accept command executes successfully, Replication Agent shuts down.

4 Start the Replication Agent instance to which you have downgraded (the earlier version), and resume replication:

```
resume purge
```

The purge keyword is needed here to purge data from the Replication Server inbound queue for the connection to which this Replication Agent is connected. Purging prevents any duplicate records from being created in Replication Server as a result of the change in OQID formats between the earlier and later versions of Replication Agent.

Downgrading Replication Agent for Microsoft SQL Server

Replication Agent for Microsoft SQL Server can be downgraded from version 15.5 to version 15.2 ESD #2. Downgrading may be necessary if the upgrade process fails. A downgrade may also be necessary if replication fails after the upgrade process has completed, such as when new features fail to function as expected, or for any of the following reasons:

- Changes to DDL and how DDL is handled by Replication Agent
- Changes to the format of the origin queue ID (OQID)

- Changes to the content or structure of the Replication Agent System Database (RASD)
- Changes to Replication Agent system objects in the primary database

If you are using any new features from Replication Agent 15.5, downgrading is not supported. For a list of the new features for Replication Agent 15.5, see the *Replication Agent New Features*.

Downgrading Replication Agent 15.5 to 15.2 ESD #2

This section describes how to downgrade Replication Agent 15.5 to 15.2 ESD #2.

❖ **Downgrading Replication Agent 15.5 to 15.2 ESD #2**

1 Change to the Replication Agent 15.5 *bin* directory:

```
cd %SYBASE%\RAX-15_5\bin
```

2 Prepare for downgrade by running the ra_downgrade_prepare command at the Replication Agent instance from which you are downgrading (the current version):

```
ra_downgrade_prepare inst_path
```

Here, *inst_path* is the absolute path of the Replication Agent instance to which you are downgrading (the earlier version).

The ra_downgrade_prepare command extracts the contents of the Replication Agent System Database (RASD) to a file named *timestamp.export*, where *timestamp* is a timestamp taken at the moment ra_downgrade_prepare was invoked. This file is located in the *import* subdirectory under the directory specified by the rasd_backup_dir configuration parameter of the Replication Agent instance to which you are downgrading (the earlier version). The path to this file is returned if ra_downgrade_prepare executes successfully.

3 Complete the downgrade by running the ra_downgrade_accept command at the Replication Agent instance to which you are downgrading (the earlier version):

```
ra_downgrade_accept timestamp.export
```

Here, *timestamp.export* is the file to which the ra_downgrade_prepare command extracted RASD contents.

If the ra_downgrade_accept command executes successfully, Replication Agent shuts down.

4 Start the Replication Agent instance to which you have downgraded (the earlier version), and resume replication:

```
resume purge
```

The purge keyword is needed here to purge data from the Replication Server inbound queue for the connection to which this Replication Agent is connected. Purging prevents any duplicate records from being created in Replication Server as a result of the change in OQID formats between the earlier and later versions of Replication Agent.

Downgrading Replication Agent for UDB

Replication Agent for UDB can be downgraded from version 15.5 to version 15.2 ESD #2. Downgrading may be necessary if the upgrade process fails. A downgrade may also be necessary if replication fails after the upgrade process has completed.

If you are using any new features from Replication Agent 15.5, downgrading is not supported. For a list of the new features for Replication Agent 15.5, see the *Replication Agent New Features*.

Downgrading Replication Agent 15.5 to 15.2 ESD #2

This section describes how to downgrade Replication Agent 15.5 to 15.2 ESD #2.

❖ Downgrading Replication Agent 15.5 to 15.2 ESD #2

1 Change to the Replication Agent 15.5 *bin* directory:

- On UNIX:

  ```
  cd $SYBASE/RAX-15_5/bin
  ```

- On Windows:

  ```
  cd %SYBASE%\RAX-15_5\bin
  ```

2 Log in to the Replication Agent for UDB 15.5 instance, and prepare for downgrade by running the ra_downgrade_prepare command at the Replication Agent instance from which you are downgrading (the current version):

```
ra_downgrade_prepare inst_path
```

Here, *inst_path* is the absolute path of the Replication Agent instance to which you are downgrading (the earlier version). The ra_downgrade_prepare command returns a message indicating that the downgrade was successful or an error message indicating that problems occurred.

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

4 Start and log in to the Replication Agent 15.2 instance, and run the ra_migrate command:

```
ra_migrate
```

5 Resume replication.

```
resume
```

APPENDIX B Using the sybfilter driver

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.

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

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.

❖ **Installing and setting 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_5\system\<platform>*.

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

   - Use winx86 if your operating system is 32-bit version of Windows Server 2003, Windows Server 2008, Windows Vista, or Windows XP.
   - Use winx64 if your operating system is 64-bit version of Windows Server 2003 or Windows XP.
   - Use winvistax64 if your operating system is 64-bit version of Windows Server 2008 or Windows Vista.

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. The sybfilter driver must be installed on the same machine as the primary Microsoft SQL Server.

   ---

3 In 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_5\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_5\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. Use directory and drive names that are recognizable to the primary Microsoft SQL Server.

- User manager – use the add command in the management console. The syntax for this command is:

  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:

  ```
  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 automatically refreshes all log paths to the sybfilter driver 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, 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, 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 similar to:

```
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:

```
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:

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

Workaround: Add write permission for the configuration file.

Microsoft SQL Server log files are locked

Problem: After restarting the machine on which Replication Agent for Microsoft SQL Server resides, you cannot open the Microsoft SQL Server log files because they are locked.

Workaround: Restart the sybfilter management console. Issue the stop command followed by the start command to restart the sybfilter driver. Restart the primary Microsoft SQL Server data server.

Using the trace log

Use sybfilter trace log information to diagnose and troubleshoot problems.

❖ **Using the trace log**

1 Turn on tracing from the sybfilter management console with the trace command and the appropriate trace flag. For example, to find out why a Microsoft SQL Server log file is unreadable after a restart, turn on tracing with the T3 flag before restarting Microsoft SQL Server:

```
trace T3
```

2 Open the sybfilter trace log file *sybfilter.trc* to view logged messages.

3 Turn off tracing from the sybfilter management console:

```
trace off
```

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* – path of the database log.

start

Start the sybfilter driver.

stop

Stop the sybfilter driver.

trace

Trace sybfilter driver execution.

Syntax:

```
trace [T1] [T2] [T3] [T4] | all | off
```

- T1 – log routine trace messages.
- T2 – log operation status informational messages.
- T3 – log normal messages.
- T4 – log error messages.
- all – log all messages for the T1, T2, T3, and T4 flags.
- off – turn tracing off.

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. • SQL Anywhere™ (formerly 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 replicate 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. Bulk copying provides an alternative to using SQL insert and select commands to transfer data.
bulk materialization	A materialization method whereby subscription data in a replicate 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 replicate database. See also **failover**.
failover	A procedure that switches user and client access from a primary database to a replicate 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 replicate 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 replicate databases are different types, with different SQL extensions and different command features. See also **function**.
gateway	Connectivity software that allows two or more computer systems with different network architectures to communicate.
inbound queue	A stable queue managed by Replication Server to spool messages received from a 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 replicate database. Specifically, latency is the time elapsed between committing an original transaction in the primary database and committing the replicated transaction in the replicate 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 replicate 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 replicate 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 replicate 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 replicate database, initializing the replicate database so that the system can begin replicating transactions. See also **atomic materialization**, **bulk materialization**, and **nonatomic 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 replicate 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 replicate 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 replicate 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 **replicate 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 **replicate 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**.
replicate data	The data managed by a replicate database, which is the destination (or target) of a replication system. See also **data replication** and **replicate database**.
replicate database	A database that contains data replicated from another database (the primary database) through a replication system. The replicate database is the database that receives replicated data in a replication system. Sometimes called the standby database. Contrast with **primary database**. See also **replicate data**.
replicated data	A set of data that is replicated from a primary database to a replicate database by a replication system. See also **primary database**, **replication system**, and **replicate database**.

replicated transaction	A primary transaction that is replicated from a primary database to a replicate database by a transaction replication system. See also **primary database**, **primary transaction**, **replicate 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 replicate 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 nonprocedural 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 replicate 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 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 replicate 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 replicate database. See also **data replication** and **disk replication**.
transactional consistency	A condition in which all transactions in the primary database are applied in the replicate 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