SYBASE®

Primary Database Guide

Sybase® Replication Agent™

12.6

[Linux, Microsoft Windows, and UNIX]

DOCUMENT ID: DC00269-01-1260-01

LAST REVISED: September 2005

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

Contents

About This Book

Sybase® Replication Agent™ version 12.6 extends the capabilities of Replication Server® by supporting non-Sybase primary data servers in a Sybase replication system.

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

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

Audience

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

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

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

How to use this book

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

This book is organized as follows:

Chapter 1, “Replication Agent for UDB,” describes replication system issues that are specific to DB2 Universal Database, and details of the Sybase Replication Agent for DB2 Universal Database.

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

Chapter 3, “Replication Agent for Microsoft SQL Server,” describes replication system issues that are specific to Microsoft SQL Server, and details of the Sybase Replication Agent for Microsoft SQL Server.

Chapter 4, “Replication Agent for Oracle,” describes replication system issues that are specific to Oracle, and details of the Sybase Replication Agent for Oracle.

Related documents

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

Replication Agent

- The Sybase Replication Agent *Administration Guide* introduces replication concepts and Sybase replication technology, and describes Replication Agent features and operations, and how to set up, administer, and troubleshoot the Replication Agent software.
- The Sybase Replication Agent *Reference Manual* describes all Replication Agent commands and configuration parameters in detail, including syntax, examples, and usage notes.
- The Sybase Replication Agent *Installation Guide* describes how to install the Sybase Replication Agent software. It includes an installation and setup worksheet that you can use to collect all of the information you need to complete the software installation and Replication Agent setup.
- The Sybase Replication Agent 12.6 release bulletin contains last-minute information that was too late to be included in the books.

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

Java environment

Sybase Replication Agent 12.6 requires a Java Runtime Environment (JRE) on the machine that acts as the Replication Agent host.

- The Sybase Replication Agent 12.6 release bulletin contains the most up-to-date information about Java and JRE requirements.
- Java documentation available from your operating system vendor describes how to set up and manage your Java environment.

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

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

Replication Server

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

Primary data servers

Sybase recommends that you or someone at your site be familiar with the software and database administration tasks for the non-Sybase data server(s) supported by Sybase Replication Agent:

- DB2 Universal Database
- Informix Dynamic Server
- Microsoft SQL Server
- Oracle

Sybase Adaptive Server

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

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

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

Other sources of information

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

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

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

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

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

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **To find the latest information on product certifications**

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

❖ **To find the latest information on component certifications**

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

3 Select Search to display the availability and certification report for the selection.

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

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

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **To find the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

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

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

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

Style conventions

The following style conventions are used in this book:

- In a sample screen display, commands that you should enter exactly as shown appear like this:

```
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
  ```

  where *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 (tables, columns, stored procedures, etc.) 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-12_6/inst_name/log* subdirectory.

Syntax conventions

The following syntax conventions are used in this book:

Table 1: Syntax conventions

Key	Definition
{ }	Curly braces indicate that you must choose at least one of the enclosed options. Do not type the braces when you enter the command.
[ ]	Brackets mean that choosing one or more of the enclosed options is optional. Do not type the brackets when you enter the command.
( )	Parentheses are to be typed as part of the command.
|	The vertical bar means you can select only one of the options shown.
,	The comma means you can choose as many of the options shown as you like, separating your choices with commas that you type as part of the command.

Statements that show the syntax of commands appear like this:

ra_config *param*[, *value*]

The words *param* and *value* in the syntax are variables or user-supplied words.

Character case

The following character case conventions are used in this book:

- All command syntax and command examples are shown in lowercase. However, Replication Agent command names are *not* case sensitive. For example, PDB_XLOG, Pdb_Xlog, and pdb_xlog are equivalent.
- Names of configuration parameters are case sensitive. For example, Scan_Sleep_Max is not the same as scan_sleep_max, and the former would be interpreted as an invalid parameter name.
- Database object names are *not* case sensitive in Replication Agent commands. However, if you need to use a mixed-case object name in a command (to match a mixed-case object name in the database), you must delimit the object name with quote characters. For example:

```
pdb_setreptable "TableName", mark
```

Accessibility features

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

Sybase Replication Agent version 12.6 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

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

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

For a Section 508 compliance statement for Sybase Replication Agent version 12.6, see Sybase Accessibility at http://www.sybase.com/detail?id=1028493.

If you need help

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

CHAPTER 1

Replication Agent for UDB

The term “Replication Agent for UDB” refers to an instance of the Sybase Replication Agent version 12.6 software that is configured for a primary database that resides in an IBM DB2 Universal Database server.

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

Note For information on the basic features and operation of Sybase Replication Agent version 12.6, refer to the Sybase Replication Agent *Administration Guide* and *Reference Manual*.

DB2 Universal Database-specific issues

This section describes general issues and considerations that are specific to using Sybase Replication Agent version 12.6 with the IBM DB2 Universal Database server.

The following topics are included in this section:

- Log-based Replication Agent
- DB2 Universal Database Administration Client
- Replication Agent connectivity
- Database logging issues
- Handling repositioning in the log
- Replication Agent behavior

- Character case of database object names
- Format of origin queue ID
- Datatype compatibility

Log-based Replication Agent

The Replication Agent for UDB is a log-based Replication Agent, unlike the Sybase Replication Agent implementations for Informix and Microsoft SQL Server.

Because of its log-based implementation, and because of differences in the behavior of the IBM DB2 Universal Database server compared to other data servers supported by Sybase Replication Agent, the Replication Agent for UDB uses different methods to acquire transaction information from the primary database. In addition, a few of the common Sybase Replication Agent features either work differently or are not available with the Replication Agent for UDB.

Feature differences in Replication Agent for UDB

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

- Creating a transaction log
- Marking a table for replication
- Support for rs_marker and rs_dump function calls
- Support for stored procedures
- Transaction log truncation

Creating a transaction log

The Replication Agent for UDB provides the same features for creating and removing the “transaction log” as other implementations of the Sybase Replication Agent. However, since the actual transaction log used by Replication Agent for UDB is the native transaction log of the DB2 Universal Database server, Replication Agent for UDB creates fewer 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 DB2 Universal Database transaction log, the user ID that the Replication Agent uses to access the primary database must have either SYSADM or DBADM authority in the database. This user ID is stored in the Replication Agent pds_username configuration parameter.

Note If the user ID specified in the pds_username parameter does not have either SYSADM or DBADM authority in the primary database, the pdb_xlog create command returns an error.

For more information about the Replication Agent for UDB transaction log, see "Replication Agent for UDB transaction log" on page 14.

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 Sybase 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 change the DATA CAPTURE attribute of a table marked for replication with the Replication Agent for UDB.

Features not available in Replication Agent for UDB

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

- Stored procedure replication
- Transaction log truncation
- DDL replication

Note When Replication Agent commands related to these features are invoked, they return an error.

Stored procedure replication

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

Transaction log truncation

Because the Replication Agent for UDB uses the native database transaction log of the DB2 Universal Database server, it cannot control log truncation as it does in other implementations of the Sybase Replication Agent. Therefore, the pdb_truncate_xlog command is not supported in Replication Agent for UDB.

The following log truncation-related configuration parameters have no effect in Replication Agent for UDB:

- truncation_interval
- truncation_type

Note For information about archiving the DB2 Universal Database transaction log, see "Archiving the transaction log" on page 17.

DB2 Universal Database Administration Client

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

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

In either case, you must configure an ODBC data source in the DB2 Universal Database Administration Client, then use the database name and database alias specified for that ODBC data source when you configure Replication Agent for UDB connectivity.

Note When you install the DB2 Universal Database Administration Client software on a Microsoft Windows platform, an environment variable named *LC_ALL* is created. This environment variable may conflict with certain Sybase utilities, such as isql and dsedit. If a conflict occurs, you can simply remove the *LC_ALL* environment variable.

To configure the DB2 Universal Database Administration Client for the Java version required by Sybase Replication Agent 12.6, you must run the *usejdbc2* script in the *path_name/sqllib/java12* directory, where *path_name* is the path where you installed the DB2 client. You must have rx privileges for the script and rw privileges for the *sqllib/java* directory and files.

DB2 UNIX environment

If the Replication Agent for UDB is installed on a UNIX platform, you must set up the DB2 UNIX environment *before* you start the Replication Agent instance, so the Replication Agent can find the UDB libraries it needs. To set up the DB2 UNIX environment, source the *db2cshrc* file in the *db2_path/sqllib* directory, where *db2_path* is the path where you installed the DB2 software.

ODBC autocommit parameter

Replication Agent for UDB requires the ODBC autocommit parameter to be turned on (autocommit=1). The ODBC autocommit parameter is specified in the DB2 call level interface (CLI) configuration file (*<installation_directory>\sqllib\db2cli.ini* for Windows, *<installation_directory>/cfg/db2cli.ini*for UNIX) for the primary database. If the ODBC autocommit parameter is not turned on, a deadlock problem can occur.

Replication Agent connectivity

Connectivity between the Replication Agent for UDB and the DB2 Universal Database server is by way of the DB2 Universal Database JDBC driver.

The DB2 Universal Database JDBC driver is installed on the Replication Agent host machine when you install the DB2 Universal Database Administration Client software. You must include the path of the JDBC driver in the *CLASSPATH* environment variable before you start the Replication Agent for UDB.

For more information on configuring the DB2 Universal Database JDBC driver, see the Sybase Replication Agent *Installation Guide*.

Configuration parameters

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

- pds_username – must have SYSADM or DBADM authority
- pds_password – for user ID specified in pds_username
- pds_database_name – database name
- pds_datasource_name – database alias
- pds_connection_type – UDBODBC (literal value)

Database logging issues

There are two database logging issues specific to DB2 Universal Database:

- DB2 Universal Database LOGRETAIN parameter
- Replication Agent read buffer size

The LOGRETAIN parameter

The Replication Agent for UDB requires the DB2 Universal Database LOGRETAIN parameter be set to RECOVERY MODE for the primary database.

To see if LOGRETAIN FOR RECOVERY is set, execute the following:

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

Note This also requires archive logging, not circular logging.

Read buffer size

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

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

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

If the read buffer size is too small to read one operation, LogReader reports an error and shuts down the Replication Agent instance.

The error Replication Agent passes from the data server is -30081. Unfortunately, this is a general communication error that DB2 Universal Database uses to report various communication problems, not just for an insufficient buffer size.

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 DB2 Universal Database transaction log for transactions to be sent to the Replication Server.

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

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

Warning! In the event that both LTM locator values are zero, there is a remote possibility of data loss. Two specific conditions could cause this data loss:

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

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

Replication Agent behavior

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

- Marking tables immediately after going to *Replicating* state, when the value of the LTM locator is 0 (zero)

- Forcing applications off the primary database with the DB2 FORCE APPLICATION command

Marking tables immediately after resume when LTM locator is zero

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

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

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

Forcing applications off the database

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

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

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

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. 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.

Note Replication will fail if a database object name is delivered to the primary Replication Server in a format different from that specified in the replication definition.

Sybase Replication Agent version 12.6 gives you some control over the way it treats the character case of database object names when it sends LTL to the primary Replication Server. You have three options:

- asis – 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.

You specify the character case option you want by setting the value of the ltl_character_case configuration parameter. The parameter values are asis, lower, and upper. The default value of the ltl_character_case parameter is asis.

In the case of the IBM DB2 Universal Database server, database object names are stored in all uppercase.

Format of origin queue ID

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

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

Table 1-1: 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 DB2 Universal Database datatypes to Sybase datatypes.

Table 1-2: DB2 Universal Database to Sybase default datatype mapping

DB2 UDB datatype	DB2 UDB length/range	Sybase datatype	Sybase length/range	Notes
BIGINT	-9,223,372,036, 854,775,808 to 9,223,372,036, 854,775,807.	decimal	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	

DB2 UDB datatype	DB2 UDB length/range	Sybase datatype	Sybase length/range	Notes
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	text	2GB	
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	The 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 or char	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	varchar	32K	
LONG VARCHAR FOR BIT DATA	32,700 bytes, binary data	varbinary	32K	
LONG VARGRAPHIC	16,350 characters, double-byte character data	univarchar or varchar	32K	
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)	

DB2 UDB datatype	DB2 UDB length/range	Sybase datatype	Sybase length/range	Notes
TIMESTAMP	0001-01-01-00.00.00.000000 to 9999-12-31-24.00.00.000000	char or datetime	32K (char)	If the pdb_convert_datetime parameter is false, TIMESTAMP values are sent as char datatype strings. If the pdb_convert_datetime parameter is true, TIMESTAMP values are converted to datetime values.
VARCHAR	32,672 bytes	varchar	32K	
VARCHAR FOR BIT DATA	32,672 bytes, binary data	varbinary	32K	
VARGRAPHIC	16,336 characters, double-byte character data	univarchar or varchar	32K	

For each datatype in Table 1-2, 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.

DB2 Universal Database datatype restrictions

If you use a version of Replication Server prior to version 12.5, the following size restrictions are imposed on DB2 datatypes:

- DB2 VARCHAR, VARCHAR FOR BIT DATA, VARGRAPHIC, and the LONG variants of those datatypes that contain more than 255 bytes will be truncated to 255 bytes.
- DB2 GRAPHIC, LONG VARGRAPHIC, and VARGRAPHIC multibyte character datatypes will be replicated as char or varchar single-byte datatypes.

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

Replication Agent for UDB transaction log

The Replication Agent for UDB uses the native database transaction log maintained by the DB2 Universal Database server to capture transactions in the primary database for replication.

Note The Replication Agent for UDB does not create stored procedures and triggers in the primary database.

The Replication Agent for UDB creates tables in the primary database for its system information. The Replication Agent system tables are created when the pdb_xlog command is invoked with the create 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-12_6\inst_name\scripts\xlog* directory.

The *create.sql* script creates Replication Agent transaction log components referred to as the *transaction log base objects*. The transaction log base objects must be created before any tables can be marked for replication in the primary database.

Note This section describes the schema and details of the Replication Agent transaction log base objects for a primary database that resides in the DB2 Universal Database server. See the Sybase Replication Agent *Administration Guide* for more general information about the Replication Agent transaction log.

Transaction log components

There are several tables used for the Replication Agent transaction log in the primary database. All tables of the Replication Agent transaction log contain at least one index, and some contain more than one index.

Transaction log object names

There are two variables in the Replication Agent transaction log table 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 transaction log table 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 transaction log tables in the primary database. If you change the value of pdb_xlog_prefix after you create the Replication Agent transaction log, the Replication Agent instance will not be able to find the transaction log tables that use the old prefix.

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

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

Transaction log base tables

Table 1-3 lists the Replication Agent system tables that are considered Replication Agent transaction log base objects. No permissions are granted on these tables when they are created.

Table 1-3: Replication Agent transaction log base tables

Table	Database name
transaction log 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*

Marker shadow tables

Table 1-4 lists the marker shadow tables that are considered Replication Agent transaction log base objects.

Table 1-4: Replication Agent marker shadow tables

Procedure/Table	Database name
rs_marker shadow table	*prefix*markersh_*xxx*
rs_dump shadow table	*prefix*dumpsh_*xxx*

Getting actual names of the transaction log objects

The Replication Agent instance generates the names of its transaction log tables. If you want to find out the actual names of transaction log tables, you must use the pdb_xlog command.

❖ **To find out the names of transaction log base tables**

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

  ```
  pdb_xlog
  ```

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

Marked objects table

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

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

Administering the transaction log

The Replication Agent for UDB does not support backing up or restoring transaction logs or truncating transaction logs. All DB2 Universal Database logs are maintained through the data server.

Archiving the transaction log

Before you archive any DB2 Universal Database transaction logs in a primary database, you must make sure that the Replication Agent has finished processing all replicated operations in the log using the following procedure.

❖ **To determine whether operations in the log have been processed by the Replication Agent**

1 With the Replication Agent in *Replicating* state, use the ra_locator command to retrieve the current value of the LTM locator from the primary Replication Server.

```
ra_locator update
```

The current value of the LTM locator from the primary Replication Server contains the DB2 Universal Database log sequence number of the last transaction operation to be replicated successfully.

Characters 4-19 of the origin queue ID are the 16-character (8-byte) operation sequence number. Characters 8-19 (6 bytes) of the operation sequence number are the DB2 log sequence number.

2 Use the DB2 db2flsn command, with the log sequence number identified in the origin queue ID, to identify which log file contains that log sequence number.

You can archive any log files up to the log file name returned by the db2flsn command.

Replication Agent for UDB setup test scripts

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

The Replication Agent test scripts are located in the *scripts* subdirectory under the Replication Agent base directory. For example: *RAX-12_6\scripts*.

The Replication Agent test scripts perform the following tasks:

1 Create a primary table.

2 Create a replicate table that corresponds to the primary table.

3 Create the primary data server connection in the primary Replication Server.

4 Create the replication definition in the primary Replication Server.

5 Test the replication definition.

6 Create the subscription in the replicate Replication Server.

7 Test the subscription.

8 Create the Replication Agent transaction log tables in the primary database.

9 Modify the primary table.

10 Clean up and remove the replication test environment created by the other scripts.

Before you begin

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

- Installed an IBM DB2 Universal Database server
- Installed a data server to act as a replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database (created a database connection for the replicate database in the RRS)

- Installed the Sybase Replication Agent software, created a Replication Agent for UDB instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, lower
  - ra_config rs_source_ds, rax
  - ra_config rs_source_db, test

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

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

Create the primary table

Use your database access tool (such as Command Line Processor) to log in to the DB2 Universal Database server and execute the *udb_create_test_primary_table.sql* script to create the primary table in the primary database.

Create the replicate table

Use isql or another query processor to log in to the replicate data server and execute the *ase_create_test_replicate_table.sql* script to create the replicate table in the replicate database.

Create the Replication Server connection for Replication Agent

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_connection.sql* script to create the Replication Server connection to the primary database.

Create the replication definition

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_repdef.sql* script to create a test replication definition.

Test the replication definition

Use isql or another query processor to log in to the RSSD of the primary Replication Server and execute the *rssd_helprep_test_repdef.sql* script to test the replication definition.

Create the subscription

There are two scripts provided for this step:

- *rs_create_test_sub.sql* – designed for use with Replication Server version 11.5 or later.
- *rs_create_test_sub_for_11.0.x.sql* – designed for use with Replication Server version 11.0.x

❖ **To create the test subscription**

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

1 Edit the appropriate script file (*rs_create_test_sub.sql* or *rs_create_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to ase.test.

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

Note If you are using the *rs_create_test_sub_for_11.0.x.sql* script, the script must be executed twice because, even though the PRS and RRS are the same Replication Server, there is not enough time between the command executions in the script to allow Replication Server to complete all its tasks before the next script command is executed. For this reason, you may wish to execute the different commands in this script separately.

Test the subscription

Use isql or another query processor to access the RSSD of the replicate Replication Server and execute the *rssd_helpsub_test_sub.sql* script to test the subscription.

Create the Replication Agent transaction log

If you have not already created the Replication Agent transaction log in the primary database, create the transaction log now.

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

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

2 Use the following command to create the transaction log:

```
pdb_xlog create
```

See the Sybase Replication Agent *Administration Guide* for more information about creating the Replication Agent transaction log.

Mark a primary table for replication

Mark the test primary table (rax_test) that was created in the primary database in the DB2 Universal Database server by the *udb_create_test_primary_table.sql* script.

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

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

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

```
pdb_setreptable rax_test, mark
```

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

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

Start replication

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

❖ **To start test replication**

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

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

```
resume
```

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

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

Execute the test transaction script in the primary database

Use your DB2 Universal Database access tool (such as Command Line Processor) to log in to the DB2 Universal Database server and execute the *udb_primary_test_transactions.sql* script to generate transactions in the primary table.

Check results of replication

Use your DB2 Universal Database access tool (such as Command Line Processor) to log in to the DB2 Universal Database server and execute the *udb_select_test_primary.sql* script to view the changes in the test primary table.

Use isql or another query processor to log in to the replicate database and execute the *ase_select_test_replicate.sql* script to verify that the transactions generated in the primary database were replicated to the test replicate table in the replicate database.

Clean up the test environment

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

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

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

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

```
quiesce
```

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

```
pdb_xlog remove, force
```

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

- *rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql* (to drop the test subscription)

Edit the appropriate script file (*rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used in the *rs_create_test_sub.sql* script. These values are initially set to ase.test.

- *rs_drop_test_repdef.sql* (to drop the test replication definition)
- *rs_drop_test_connection.sql* (to drop the test connection to the primary database)

5 Log in to the replicate data server and execute the *ase_drop_test_replicate_table.sql* script to drop the test replicate table.

6 Use your DB2 Universal Database access tool (such as Command Line Processor) to log in to the DB2 Universal Database server and execute the *udb_drop_test_primary_table.sql* script to drop the test primary table.

CHAPTER 2

Replication Agent for Informix

The term “Replication Agent for Informix” refers to an instance of the Sybase Replication Agent version 12.6 software configured for a primary database that resides in an Informix Dynamic Server data server.

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

Note For information on the basic functionality of Sybase Replication Agent version 12.6, refer to the Sybase Replication Agent *Administration Guide* and *Reference Manual*.

Informix-specific issues

This section describes general issues and considerations that are specific to using Sybase Replication Agent version 12.6 with the Informix data server.

Note DDL replication is not supported for Informix.

The following topics are included in this section:

- Replication Agent connectivity
- Transaction isolation
- Informix trigger limitations
- Maximum number of columns in a table

- Transaction commit order
- Change database command limitation
- Parameters to control statistics update
- Character case of database object names
- Stored procedure replication
- Format of origin queue ID
- Datatype compatibility

Replication Agent connectivity

Connectivity between the Replication Agent for Informix and the Informix data server occurs through the Informix JDBC driver.

The Informix JDBC driver must be installed on the Replication Agent host machine, and the full path name of this driver must be included in the user's CLASSPATH environment variable before starting a Replication Agent instance.

For more information on configuring the Informix JDBC driver, see the Sybase Replication Agent *Installation Guide*.

Transaction isolation

The default Informix database transaction isolation must be Committed Read. It cannot be Uncommitted Read or Transactions not supported.

This is ANSI standard behavior that guarantees that *user2* will see the data only after *user1* has committed the changes. If a change is in progress, *user2* will be locked out until the change is either committed or rolled back. This is only possible if the database is logged.

Informix trigger limitations

There are three types of trigger limitations in Replication Agent when using an Informix primary database:

- Selective update triggers are not allowed.

- Disabled triggers are not allowed.
- Reentrant triggers should be avoided.

Selective triggers

No table marked for replication can have a selective update trigger on it. (A selective update trigger is an update trigger that references specific columns.)

In the Informix Dynamic Server, a table may have multiple update triggers, each of which specifies a subset of the columns. However, each column may be referenced by only one update trigger.

Replication Agent requires an update trigger for all columns (an entire row), so if a selective update trigger already exists on the table, Replication Agent cannot create its update trigger, and therefore it cannot replicate transactions against that table.

Note Dynamic Server 2000 allows users to create select triggers. If the select trigger does not modify data in the triggering table, then the selective select trigger is allowed on marked tables.

Disabled triggers

No table marked for replication can have a disabled insert, update, or delete trigger on it.

Even though Replication Agent can modify a disabled trigger, as long as the trigger is disabled, it does not execute when the corresponding operation occurs; therefore, replication of that operation does not take place.

Reentrant triggers

Dynamic Server 2000 allows users to create reentrant triggers. A reentrant trigger is one where the triggering event and triggered action both act on the same table.

Note Reentrant triggers may cause problems with replication. Therefore, Sybase recommends that you avoid using reentrant triggers on tables marked for replication.

Maximum number of columns in a table

A table marked for replication can have no more than 4 columns less than the maximum number allowed by the Informix Dynamic Server instance.

For example, if the database instance allows up to 232 columns in a table, any table marked for replication can have up to 228 columns. Replication Agent adds 4 additional columns to a transaction log shadow table, which must not exceed the maximum number of columns allowed by the database instance.

Transaction commit order

The Replication Agent transaction log is implemented using database objects (tables, stored procedures, and triggers) in the primary Informix database. Because of this and the way the Informix database handles transactions internally, there is some risk that transactions may not be applied to a replicate table in the same order they were applied to the primary table.

For example, if transaction *B* commits during the time lapse between the last data change operation and the commit for transaction *A*, Sybase Replication Agent presents the transactions to the primary Replication Server with the order *AB*, when the actual commit order was *BA*.

To avoid this situation, you can do one of the following:

- Ensure that commits are performed immediately after the last data-changing operation in a transaction.
- Perform a dummy data-altering operation to a marked table just before committing a transaction.

Change database command limitation

Sybase Replication Agent provides a command that allows you to specify the current database for executing SQL commands (pdb_set_sql_database) and a command that allows you to view a list of the available databases on the data server Replication Agent is connected to (pdb_get_databases).

Because of limitations in the functionality of the Informix JDBC driver, both of these Replication Agent commands behave as if the Informix data server has only one database:

- The pdb_set_sql_database command returns success, but the current database does not change.

- The pdb_get_databases command returns only the name of the current database.

Parameters to control statistics update

When Sybase Replication Agent truncates its transaction log in the primary database, it updates its internal statistics counters. The statistics update process entails querying transaction log objects in the primary database. Because these query operations consume system resources, they can have an adverse affect on primary database performance if they occur frequently in systems with very large or numerous marked tables, or large transaction volume.

To reduce the impact of log truncation statistics update queries, the Replication Agent for Informix provides two parameters that allow you to control the level of the statistics update queries and how often they run:

- pdb_update_stats_count – defines the cumulative number of rows that must be truncated from the log before running the statistics update queries.
- pdb_update_stats_type – defines the level of queries to run (low, medium, or high).

Note These configuration parameters are supported only by the Replication Agent for Informix. They have no effect with a Replication Agent for another type of data server.

To set or change the pdb_update_stats_count and pdb_update_stats_type configuration parameters, use the ra_config command. See the Sybase Replication Agent *Administration Guide* and *Reference Manual* for more information.

The following sections describe these configuration parameters in detail.

pdb_update_stats_count

Specifies the number of rows that must be truncated from the Replication Agent transaction log before running statistics update queries. This number is cumulative, that is, it can include rows truncated in more than one truncation.

Default	10000
Value	An integer from 0 to 2,147,483,647.
Comments	• A value of 0 (zero) turns off statistics update queries.

pdb_update_stats_type

Determines and defines the level of statistics update queries to run.

Default: medium

Values:

low – generates and updates statistical data regarding table, row, and page count statistics.

medium – generates the same statistics as the low option, and also generates statistics about the distribution of data values for each specified column.

high – generates the same statistics as the low option, and also generates statistics about the distribution of data values for each specified column.

Comments:

- A value of 0 (zero) turns off statistics update queries.

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. 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.

Note Replication will fail if a database object name is delivered to the primary Replication Server in a format different from that specified in the replication definition.

Sybase Replication Agent version 12.6 gives you some control over the way it treats the character case of database object names when it sends LTL to the primary Replication Server. You have three options:

- asis – 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.

You specify the character case option you want by setting the value of the ltl_character_case configuration parameter. The parameter values are asis, lower, and upper. The default value of the ltl_character_case parameter is asis.

In the case of the Informix data server, database object names are stored in all lowercase.

Stored procedure replication

There are two issues related to stored procedure replication from a Dynamic Server 2000 primary database:

- Replicating user-defined routines (UDRs) from a Dynamic Server 2000 primary database
- UDR syntax

Replicating UDRs

The following limitations apply to replicating user-defined routines (UDRs) from a primary database in Dynamic Server 2000:

- UDRs written in an external language, such as C or Java, cannot be replicated because the Replication Agent cannot access external UDR source. To be replicated, a UDR must be written in SPL (stored procedure language).
- When searching the primary database for the UDR named in a pdb_setrepproc *proc*, mark command, the Replication Agent searches for a UDR with the specific name of *proc* or with the actual name of *proc* (and if *owner* is specified, *owner.proc*), using it in the where clause. If no UDR is found, or if more than one UDR is found, the specified UDR cannot be marked for replication as a stored procedure.

  If the specific-name search succeeds and no replicate name was supplied in the pdb_setrepproc command, the replicate name will be the actual UDR name.

UDR syntax

To mark a UDR for replication, the Replication Agent must be able to distinguish between the routine header and the routine body.

In versions of the Informix Dynamic Server prior to Dynamic Server 2000 (version 9.2), stored procedure syntax required a semicolon at the end of the returning clause, or the header ended with the close parenthesis following the parameter list.

In Dynamic Server 2000, the semicolon at the end of the header is optional. Without the semicolon, or without a begin label, the Replication Agent cannot distinguish the routine header from the routine body.

If the routine has any header syntax elements following the parameter list, such as the specific name, the semicolon must be used at the end of all the header options, or a begin label must be used to designate the beginning of the statement block. Otherwise, Replication Agent cannot mark the UDR for replication.

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 Informix with Informix Dynamic Server version 7.x and 2000.

Table 2-1: Replication Agent for Informix origin queue ID for Dynamic Server version 7.x

Character	Bytes	Description
0-3	2	Database generation ID
4-19	8	Transaction max sequence
20-27	4	Operationid HI
28-35	4	Operationid LO
36-39	2	Operation type (begin = 0, data/LOB = 1, commit/rollback = 7FFF)
40-55	8	Transaction ID
56-63	4	LOB sequence ID

Table 2-2 illustrates the format of the origin queue ID for the Replication Agent for Informix with Informix Dynamic Server 2000 (version 9.2).

Table 2-2: Replication Agent for Informix origin queue ID for Dynamic Server 2000 version 9.2

Character	Bytes	Description
0-3	2	Database generation ID
4-19	8	Transaction max sequence
20-35	8	Operationid
36-39	2	Operation type (begin = 0, data/LOB = 1, commit/rollback = 7FFF)
40-55	8	Transaction ID
56-63	4	LOB sequence ID

Datatype compatibility

Replication Agent for Informix processes transactions and stored procedure invocations 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 Informix.

The following table describes the default conversions of Informix datatypes to Sybase datatypes.

Table 2-3: Informix to Sybase default datatype mapping

Informix datatype	Informix length/range	Sybase datatype	Sybase length/range	Notes
byte	2^{31} bytes, binary data	image	2GB	
blob	4 terabytes ($4*2^{40}$), binary data	image	2GB	
boolean	1 byte	char(1)	1 byte	Values are replicated as literal characters t or f.
char(n)	32,767 bytes	char, varchar	32K	
clob	4 terabytes ($4*2^{40}$), text data	text	2GB	

Informix datatype	Informix length/range	Sybase datatype	Sybase length/range	Notes
date	Number of days since 12/31/1899	date or datetime	01/01/1753 to 12/31/9999	No special effort is required to format the value for either Replication Server or Adaptive Server. If the Informix value falls outside the Replication Server supported range, Replication Server will not accept it. Options for handling this situation include: • Using char(10) as the datatype in the replication definition. • Modifying the triggers and stored procedures that support the shadow table to normalize values outside Replication Server limits.
datetime *largest_qualifier* to *smallest_qualifier*	YEAR: 1 - 9999 MONTH: 1 - 12 DAY: 1 - 31 HOUR: 0 - 23 MINUTE: 0 - 59 SECOND: 0 - 59 FRACTION: 0 - 99999	char(n) or datetime	If datetime is used, 01/01/1753 to 12/31/9999	If char is used, the value of *n* depends on the date elements defined for the column. For example, datetime year to fraction(5) requires *n* to be 25, datetime month to day requires *n* to be 5. If datetime is used, set the value of the pdb_convert_datetime parameter to true before marking the table, then deal with the range issue as described for the Informix date datatype.

Informix datatype	Informix length/range	Sybase datatype	Sybase length/range	Notes
decimal(p) or dec(p)	10^{-130} to 10^{124}, up to 32 significant digits	decimal or float	When decimal is used, 10^{-38} to 10^{38}, 38 significant digits. float precision and range corresponds to a C double datatype, approximately 16 significant digits.	Using the decimal datatype allows precision at the expense of range. Using the float datatype allows range (to the extent that the Replication Server platform supports range) at the expense of precision. However, extremely small values get truncated at log time to 16 digits to the right of the decimal. For example, with the primary column datatype of decimal(32), the number 1.23456789012345678901234567 89012e-10 is logged as 0.0000000001234567. Extremely large values retain precision.
decimal(p,s) or dec(p,s) or numeric(p,s)	Range without error is 0 to 10^{p-s} - 10^{-s}, up to 32 significant digits	decimal	10^{-38} to 10^{38}, 38 significant digits.	
float(n) or double precision	Corresponds to a C double datatype on the given system, approximately 16 significant digits.	float	Corresponds to a C double datatype on the given system, approximately 16 significant digits.	Errors can be introduced if the Informix platform and Replication Server platform handle the C double datatype differently. Extremely large values may lose significant digits at log time. For example, with primary column datatype of float, the number 1.234567890123456e125 is logged as 1.234568e+125. However, extremely small values retain all digits.)
integer or int	-2,147,483,647 to 2,147,483,647.	int	-2,147,483,648 to 2,147,483,647.	
int8	-9,223,372,036, 854,775,807 to 9,223,372,036, 854,775,807	decimal	10^{-38} to 10^{38}, 38 significant digits.	

Informix datatype	Informix length/range	Sybase datatype	Sybase length/range	Notes
interval *largest_qualifier*(n) to *smallest_qualifier*(n)	1 <= *n* <= 9 for *largest_qualifier*. If the type of *smallest_qualifier* is fraction, 1 <= *n* <= 5. (*n* on *smallest_qualifier* only valid if *smallest_qualifier* is a fraction.)	char(n)	32K	The value of the char *n* depends on the interval elements defined for the column. For example, interval year(5) to month requires *n* to be 9, interval day(9) to fraction(5) requires *n* to be 25. By modifying the script that builds the shadow table and associated primary table triggers and stored procedures, you can break the interval value into separate integer values. Or you can use a function string to do this breakdown for each replicate.
lvarchar	4 GB	text	2GB	
money(p,s)	Range without error is 0 to 10^{p-s} - 10^{-s}, up to 32 significant digits	money or small-money	The money range is -922,337,203, 685,477.5808 to 922,337,203, 685,477.5807. The smallmoney range is -214,748.3648 to 214,748.3647.	Both money and smallmoney datatypes are accurate to one ten-thousandth of a monetary unit. This results in truncation if the primary column allows accuracy beyond that level.
nchar(n)	Up to 32,767 bytes of multi-byte character data	unichar or char	32K	
nvarchar(m,r)	Up to 255 bytes of multibyte character data	univarchar or varchar	32K	
real	See smallfloat.			

Informix datatype	Informix length/range	Sybase datatype	Sybase length/range	Notes

serial	-2,147,483,647 to 2,147,483,647.	identity or integer	-2,147,483,648 to 2,147,483,647.	If identity is used, the following command is applied to the replicated table before an insert command:
				`set identity_insert table_name on`
				The following command is applied to the replicated table after an insert command:
				`set identity_insert table_name off`
				Note Columns with the identity datatype are never updated by the update command.

serial8	-9,223,372,036, 854,775,807 to 9,223,372,036, 854,775,807	identity or numeric	1 to 10^{38} - 1	If identity is used, the following command is applied to the replicated table before an insert command:
				`set identity_insert table_name on`
				The following command is applied to the replicated table after an insert command:
				`set identity_insert table_name off`
				Note Columns with the identity datatype are never updated by the update command.

smallfloat	Corresponds to a C float datatype, approximately 8 significant digits.	real	Corresponds to a C float datatype, approximately 6 significant digits.	Errors may be introduced if the Informix platform and Replication Server platform handle the C float datatype differently.
smallint	-32,767 to 32,767	smallint	-32,768 to 32,767	
text	2GB, text data.	text	2GB	
varchar(m [,r])	255 bytes	char or varchar	32K	

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

- Character and graphic datatypes – maximum number of bytes.
- Numeric and temporal datatypes – range from smallest to largest values.

Informix datatype restrictions

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

- Informix blob, clob, and lvarchar datatypes that contain more than 2GB will be truncated to 2GB.
- Informix char and varchar datatypes that contain more than 255 bytes will be truncated to 255 bytes.

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

Replication Agent for Informix transaction log

Sybase Replication Agent uses its own proprietary transaction log to capture and record transactions in the primary database for replication. The Replication Agent transaction log consists of a set of shadow tables, stored procedures, and triggers that are created in the primary database.

The Replication Agent transaction log is created by invoking the pdb_xlog command with the create keyword. When you invoke this command, Replication Agent generates a SQL script that is run in the primary database. This script (stored in the *create.sql* file in the *RAX-12_6\inst_name\scripts\xlog* directory) creates the Replication Agent transaction log components referred to as the *transaction log base objects*. The transaction log base objects must be created before any objects can be marked for replication in the primary database.

Note This section describes the schema and details of the Replication Agent transaction log for an Informix database. For more general information about the Replication Agent transaction log, see the Sybase Replication Agent *Administration Guide*.

Transaction log components

There are three types of Informix database objects used for transaction log components:

- Tables
- Stored procedures
- Triggers

There are several tables used by the Replication Agent transaction log, both as base objects and as shadow tables for marked primary tables in the primary database. All table components of the transaction log contain at least one index, and some contain more than one index.

Stored procedures are used by the Replication Agent transaction log to capture transactions for replication and to convert the datetime data type.

Triggers are used to capture the insert, update, and delete operations in marked primary tables.

Transaction log 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 transaction log component 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 transaction log objects in the primary database. If you change the value of pdb_xlog_prefix after you create the transaction log, Replication Agent will not be able to find the transaction log objects that use the old prefix.

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

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

Transaction log base tables

Table 2-4 lists the tables that are considered Replication Agent transaction log base objects. No permissions are granted on these tables when they are created.

Table 2-4: Replication Agent transaction log base tables

Table	Database name
transaction log system table	*prefix*xlog_system_
marked objects table	*prefix*marked_objs_*xxx*
transaction log table	*prefix*tran_log_*xxx*
LOB columns table	*prefix*blob_columns_*xxx*
exception holding table	*prefix*exception_*xxx*
Log Reader work table	*prefix*lr_reptran_*xxx*
Log Admin work table	*prefix*rawork_*xxx*
proc active table	*prefix*procactive_*xxx*

Transaction log stored procedures

Table 2-5 lists the stored procedures that are considered Replication Agent transaction log base objects. Execute permission is granted to public when these stored procedures are created (except as noted).

Note The transaction log stored procedures listed in Table 2-5 have no effect when executed outside the context of replication.

Table 2-5: Replication Agent transaction log stored procedures

Procedure	Database name
datetime conversion	*prefix*convert_dt_*xxx*
transaction information capture	*prefix*get_tx_info_*xxx*
transaction log row	*prefix*txlog_row_*xxx*
Log Reader scanning	*prefix*bld_reptran_*xxx* (no permission granted)
rollover handling	*prefix*rollover_*xxx* (no permission granted)

Marker procedures and shadow tables

Table 2-6 lists the marker procedures and marker shadow tables that are considered Replication Agent transaction log base objects. Execute permission is granted to public when the marker procedures are created.

Table 2-6: Replication Agent marker procedures and shadow tables

Procedure/Table	Database name
transaction log-marker procedure	rs_marker*xxx*
transaction log-marker shadow table	*prefix*markersh_*xxx*
dump marker procedure	rs_dump*xxx*
dump marker shadow table	*prefix*dumpsh_*xxx*

Transaction log objects for each marked table

Table 2-7 lists the Replication Agent transaction log objects that are created for each primary table that is marked for replication. These objects are created only when a table is marked for replication. These objects are not considered transaction log base objects and are not displayed by the pdb_xlog command as the other objects are.

No permission is granted on these procedures.

Table 2-7: Replication Agent transaction log objects for each marked table

Object	Database name
shadow table	*prefix*sh_*xxx*
shadow row procedure	*prefix*srp_*xxx*
LOB shadow table	*prefix*bsh_*xxx*
LOB shadow row procedure	*prefix*bsrp_*xxx*
insert trigger	*prefix*instrg_*xxx* (or existing-trigger name)
update trigger	*prefix*updtrg_*xxx* (or existing-trigger name)
delete trigger	*prefix*deltrg_*xxx* (or existing-trigger name)

Note The marked objects table records the mapping between marked objects and their corresponding shadow tables and shadow row procedures. The pdb_setreptable and pdb_setrepproc commands return the names of the shadow tables and shadow row procedures.

Reserved names

There are two type of reserved names in the Replication Agent transaction log:

- Column names in Replication Agent shadow tables
- Global variables in Replication Agent stored procedures

Shadow table column names

When a primary table is marked for replication, Replication Agent creates a shadow table to record the replicated operations. The shadow table has all the columns of the primary table, plus four columns that the Replication Agent creates for its use:

- ra_tran_id_
- ra_opid_hi
- ra_opid_lo
- ra_img_type_

Note These column names are fixed, not generated; therefore, the "ra_" portion of the column name does not change when the value of the pdb_xlog_prefix parameter changes.

If a primary table has a column with the same name as one of the Replication Agent shadow table columns, the table marking procedure fails. Replication Agent flags the offending column in the table-marking script and the pdb_setreptable command returns an error.

In this event, you must change the name of the conflicting column in the primary table. After you change the name of the conflicting column, you can modify the table-marking script and run it manually to mark the primary table.

Note If you want to use the primary table's original column name in the replicate table, you can create a function string in the replicate Replication Server to map the new column name in the primary table to the original column name in the replicate table.

Global variables

The following global variables are used in Replication Agent transaction log stored procedures:

- ra_txid_
- ra_opid_hi_
- ra_opid_lo_

- ra_tstamp_

Note These variable names are fixed, not generated; therefore, the "ra_" portion of the variable name does not change when the value of the pdb_xlog_prefix parameter changes.

You cannot use these names as column names in a primary table or as global variables in a primary stored procedure in the primary database.

Marked table triggers

If a trigger exists on a primary table when that table is marked for replication, the existing trigger is modified to add a *for-each-row* section (if the trigger does not already contain a *for-each-row* section) to support Sybase replication.

The Sybase replication action in a trigger begins with a comment that serves as a marker so that, when the table is unmarked, the Sybase replication action can be removed from the trigger. If the resulting *for-each-row* section is empty, it will also be removed. If removing the *for-each-row* section leaves the trigger empty, the trigger will be dropped.

Note Do not remove or alter the Sybase replication comments in triggers. If you need to modify a trigger that contains the Sybase replication action, you must keep the Sybase replication action at the end of the trigger (the last action in the trigger).

A "selective update" trigger is an update trigger that acts based on operations only in specified columns.

If a "selective update" trigger exists on a table, Replication Agent cannot modify the trigger and changes to that table cannot be replicated. For more information on the "selective trigger" limitation, see "Selective triggers" on page 27.

Getting actual names of the transaction log objects

Because Sybase Replication Agent generates the names of its transaction log objects using its own algorithms, there is not any obvious correlation between the name of a primary object and the names of the transaction log objects (such as shadow tables) that record replicated operations for the primary object. If you want to find out the names of transaction log objects associated with a primary object, use the following procedure.

❖ **To determine the names of transaction log objects associated with a primary object**

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

```
pdb_xlog
```

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

2 At the Replication Agent administration port, invoke the pdb_setrepproc command with no keywords:

```
pdb_setrepproc
```

The pdb_setrepproc command returns a list of all stored procedures marked for replication, including all the Replication Agent transaction log objects associated with each marked procedure.

3 At the Replication Agent administration port, invoke the pdb_setreptable command with no keywords:

```
pdb_setreptable
```

The pdb_setreptable command returns a list of all tables marked for replication, including all the Replication Agent transaction log objects associated with each marked table.

Marked objects table

One of the Replication Agent transaction log base objects is the **marked objects table**. The marked objects table contains an entry for each marked object in the primary database (both tables and stored procedures). Each marked object entry contains the following information:

- Name of the marked primary object
- Primary object's replicated name (if any)
- Type of the primary object
- "Replication enabled" flag for the primary object
- Name of the shadow table for the primary object
- Name of the operation-image procedure
- Name of the LOB shadow table (if the primary object is a LOB column)
- Name of the LOB image procedure (if the primary object is a LOB column)

- Owner of the primary object
- “Send owner” flag

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

Sybase Replication Agent does not support backing up and restoring the transaction log automatically.

Sybase recommends that you use the database backup utilities provided with your Informix Dynamic Server software to periodically back up the transaction log.

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

Truncating the transaction log

Sybase 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.

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.

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.

If you want to truncate the transaction log at a specific time, you can use a scheduler utility to execute the pdb_truncate_xlog command automatically.

Replication Agent for Informix setup test scripts

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

The Replication Agent test scripts are located in the *scripts* subdirectory under the Replication Agent base directory. For example: *RAX-12_6\scripts*.

The Replication Agent test scripts perform the following tasks:

1 Create a primary table.

2 Create a replicate table that corresponds to the primary table.

3 Create the primary data server connection in the primary Replication Server.

4 Create the replication definition in the primary Replication Server.

5 Test the replication definition.

6 Create the subscription in the replicate Replication Server.

7 Test the subscription.

8 Create the Replication Agent transaction log in the primary database.

9 Modify the primary table.

10 Clean up and remove the replication test environment created by the other scripts.

Before you begin

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

- Installed an Informix data server
- Installed a data server to act as a replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database
- Installed the Replication Agent software, created a Replication Agent instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, lower
  - ra_config rs_source_ds, rax
  - ra_config rs_source_db, test

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

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

Create the primary table

Use your Informix database access tool (such as dbaccess) to log in to the Informix data server and execute the *informix_create_test_primary_table.sql* script to create the primary table in the primary database.

Create the replicate table

Use isql or another query processor to log in to the replicate data server and execute the *ase_create_test_replicate_table.sql* script to create the replicate table in the replicate database.

Create the Replication Server connection for Replication Agent

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_connection.sql* script to create the Replication Server connection to the primary database.

Create the replication definition

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_repdef.sql* script to create a test replication definition.

Test the replication definition

Use isql or another query processor to log in to the RSSD of the primary Replication Server and execute the *rssd_helprep_test_repdef.sql* script to test the replication definition.

Create the subscription

There are two scripts provided for this step:

- *rs_create_test_sub.sql* – designed for use with Replication Server version 11.5 or later.
- *rs_create_test_sub_for_11.0.x.sql* – designed for use with Replication Server version 11.0.x.

❖ **To create the test subscription**

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

1 Edit the appropriate script file (*rs_create_test_sub.sql* or *rs_create_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to ase.test.

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

Note If you are using the *rs_create_test_sub_for_11.0.x.sql* script, the script must be executed twice because, even though the PRS and RRS are the same Replication Server, there is not enough time between the command executions in the script to allow Replication Server to complete all its tasks before the next script command is executed. For this reason, you may wish to execute the different commands in this script separately.

Test the subscription

Use isql or another query processor to access the RSSD of the replicate Replication Server and execute the *rssd_helpsub_test_sub.sql* script to test the subscription.

Create the Replication Agent transaction log

If you have not already created the Replication Agent transaction log in the primary database, create the transaction log now.

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

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

2 Use the following command to create the transaction log:

```
pdb_xlog create
```

See the Sybase Replication Agent *Administration Guide* for more information about creating the Replication Agent transaction log.

Mark a primary table for replication

Mark the test primary table (rax_test) that was created in the Informix database by the *informix_create_test_primary_table.sql* script.

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

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

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

```
pdb_setreptable rax_test, mark
```

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

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

Start replication

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

❖ **To start test replication**

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

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

```
resume
```

See the Sybase Replication Agent *Administration Guide* for more information about starting replication.

Execute the test transaction script in Informix

Use your Informix database access tool (such as dbaccess) to log in to the Informix data server and execute the *informix_primary_test_transactions.sql* script to generate transactions in the primary table in the primary database.

Check results of replication

Use your Informix database access tool (such as dbaccess) to log in to the Informix data server and execute the *informix_select_test_primary.sql* script to view the changes in the test primary table in the primary database.

Use isql or another query processor to log in to the replicate database and execute the *ase_select_test_replicate.sql* script to verify that the transactions generated in the primary database were replicated to the test replicate table in the replicate database.

Clean up the test environment

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

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

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

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

```
quiesce
```

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

```
pdb_xlog remove, force
```

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

- *rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql* (to drop the test subscription)

Edit the appropriate script file (*rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used in the *rs_create_test_sub.sql* script. These values are initially set to ase.test.

- *rs_drop_test_repdef.sql* (to drop the test replication definition)
- *rs_drop_test_connection.sql* (to drop the test connection to the primary database)

5 Log in to the replicate data server and execute the *ase_drop_test_replicate_table.sql* script to drop the test replicate table.

6 Use your Informix database access tool (such as dbaccess) to log in to the Informix data server and execute the *informix_drop_test_primary_table.sql* script to drop the test primary table.

CHAPTER 3

Replication Agent for Microsoft SQL Server

The term “Replication Agent for Microsoft SQL Server” refers to an instance of the Sybase Replication Agent version 12.6 software installed and configured for a primary database that resides in a Microsoft SQL Server data server.

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

Note For information on the basic functionality of Sybase Replication Agent version 12.6, refer to the Sybase Replication Agent *Administration Guide* and *Reference Manual*.

Microsoft SQL Server-specific issues

This section describes general issues and considerations that are specific to using Sybase Replication Agent version 12.6 with the Microsoft SQL Server data server.

In this chapter, the term “Windows” refers to all supported Microsoft Windows platforms. See the Sybase Replication Agent version 12.6 release bulletin for more information on which Microsoft Windows platforms are supported.

Note DDL replication is not supported for Microsoft SQL server.

The following topics are included in this section:

- Replication Agent communications
- Replication Agent permissions
- Maximum number of columns in a table
- @@IDENTITY system variable
- Length of owner names
- Microsoft isql tool
- Character case of database object names
- Format of origin queue ID
- Datatype compatibility

Replication Agent communications

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

Replication Agent for Microsoft SQL Server connects to Microsoft SQL Server using the JDBC driver provided with the Sybase Replication Agent software.

Replication Agent for Microsoft SQL Server uses Sybase jConnect for JDBC to communicate with the primary Replication Server and its RSSD.

While replicating transactions and function invocations, Replication Agent maintains connections with both the primary database and the primary Replication Server. In addition, Replication Agent occasionally connects to the RSSD of the primary Replication Server to retrieve replication definition data.

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

Replication Agent permissions

Replication Agent for Microsoft SQL Server must create database objects (tables, triggers, and procedures) in the primary database for its transaction log.

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

- Create table
- Create trigger
- Create procedure

Maximum number of columns in a table

A table marked for replication can have no more than three columns less than the maximum number allowed by the Microsoft SQL Server database.

For example, if the database allows up to 232 columns in a table, any table marked for replication can have up to 229 columns. Replication Agent adds three additional columns to a transaction log shadow table, which must not exceed the maximum number of columns allowed by the database.

@@IDENTITY system variable

Microsoft SQL Server has a system variable named *@@IDENTITY*, which stores the most recent value of the identity column after an insert operation. When a Replication Agent trigger executes to capture an insert operation in a marked table, that trigger performs an insert operation in one of the transaction log shadow tables. As a result of the trigger execution, the value of the *@@IDENTITY* variable after an insert operation in a marked table is actually the value after the trigger's insert in the shadow table. Since the Replication Agent shadow table does not have an identity column, the value of the *@@IDENTITY* variable is set to NULL.

If you have a SQL Server application that uses the *@@IDENTITY* variable, use the following procedure to work around this problem:

❖ **To work around the *@@IDENTITY* variable problem**

1 Create a table in the primary database that has a single column defined as integer identity.

  This column will be used to store the original value of the *@@IDENTITY* variable.

2 Modify the Replication Agent insert trigger (*prefix*instrg_*xxx*) as follows:

- Create a local variable in the Replication Agent insert trigger (*prefix*instrg_*xxx*) to hold the value of the *@@IDENTITY* variable.
- Immediately save the value of the *@@IDENTITY* variable in the trigger's local variable before performing any replication operations.
- After all replication operations are complete, use the value saved in the trigger's local variable to insert a new record in the identity column of the new table created for this purpose. (You have to temporarily turn on the Identity_Insert option to insert an explicit value in the identity column, then turn it off after the insert.)

Note To execute a trigger that contains the Identity_Insert command, the Replication Agent user ID must have sa or table owner permissions.

After the original value of the *@@IDENTITY* variable is inserted in the identity column in the new table, the value of the *@@IDENTITY* variable is restored.

Length of owner names

The maximum length of owner names is increased to 128 characters in Microsoft SQL Server 2000. Replication Server 12.6 supports user (table or procedure owner) names up to 30 bytes.

Microsoft isql tool

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

Although the name of the Microsoft isql tool is the same as the Sybase tool called isql, the Sybase and Microsoft tools are not compatible. For example, you cannot use the Sybase isql tool to access the Microsoft SQL Server data server, and you cannot use the Microsoft isql tool to access the Replication Agent administration port.

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

Character case of database object names

Database object names must be delivered to the primary Replication Server in the same format as they are specified in replication definitions. 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.

Note Replication will fail if a database object name is delivered to the primary Replication Server in a format different from that specified in the replication definition.

Sybase Replication Agent version 12.6 gives you some control over the way it treats the character case of database object names when it sends LTL to the primary Replication Server. You have three options:

- asis – 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.

You specify the character case option you want by setting the value of the ltl_character_case configuration parameter. The parameter values are asis, lower, and upper. The default value of the ltl_character_case parameter is asis.

In the case of Microsoft SQL Server, database object names are stored in the same case as entered (upper case and/or lower case). 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 3-1 illustrates the format of the origin queue ID for the Replication Agent for Microsoft SQL Server.

Table 3-1: Replication Agent for Microsoft SQL Server origin queue ID

Character	Bytes	Description
0-3	2	Database generation ID
4-19	8	Transaction max sequence
20-35	8	Operation sequence
36-43	2	Operation type (begin = 0, data/LOB = 1, commit/rollback = 7FFF)
44-59	8	Transaction ID
60-63	4	LOB sequence ID

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.

The following table describes the default conversion of Microsoft SQL Server datatypes to Sybase Replication Server datatypes.

Table 3-2: Microsoft SQL Server to Replication Server default datatype mapping

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

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
money	Monetary from -2^{63} to $2^{63} - 1$	money		
smallmoney	Monetary from -214,748.3648 to 214,748.3647	smallmoney		
float	Floating precision from -1.79E + 308 to 1.79E + 308.			Results in Sybase are machine dependent.
real	Floating precision from -3.40E + 38 to 3.40E + 38			Results in Sybase are machine dependent.
datetime	Date and time from 01/01/1753 to 12/31/9999	datetime		
smalldatetime	Date and time from 01/01/1900 to 06/06/2079	datetime		
timestamp	Database-wide unique number	timestamp		To retain the actual value assigned in SQL Server, replicate to the varbinary(8) datatype.
uniqueidentifier	Globally unique identifier	varbinary		No Sybase equivalent. Map to binary(38) or varbinary(38) datatype.
char	Fixed length up to 8000 characters	char	32K	
varchar	Variable length up to 8000 characters	varchar	32K	
text	Variable length up to $2^{31} - 1$ characters	text	2GB	
nchar	Fixed length Unicode up to 4000 characters	unichar or char	32K	Actual maximum length is @@ncharsize * number of characters.
nvarchar	Variable length Unicode up to 4000 characters	univarchar or varchar	32K	Actual maximum length is @@ncharsize * number of characters.
ntext	Variable length Unicode up to $2^{30} - 1$ characters	text	2GB	

Microsoft SQL Server datatype	Microsoft SQL Server length/range	Sybase datatype	Sybase length/range	Notes
binary	Fixed length up to 8000 bytes	binary	32K	
varbinary	Variable length up to 8000 bytes	varbinary	32K	
image	Variable length up to 2^{31} - 1 bytes	image	2GB	
sql_variant	Any datatype except text, ntext, timestamp, and sql_variant, up to 8000 bytes	varbinary	32K	

Microsoft SQL Server datatype restrictions

Replication Server and Replication Agent impose the following constraints on Microsoft SQL Server datatypes:

- If you use a version of Replication Server prior to version 12, you must set the value of the Replication Agent use_rssd configuration parameter to true to avoid problems with the Microsoft SQL Server smallint and tinyint datatypes.
- If you use a version of Replication Server prior to version 12.5, the following size restrictions are imposed on Microsoft SQL Server datatypes:
  - Microsoft SQL Server char and varchar datatypes that contain more than 255 bytes will be truncated to 255 bytes.

- Microsoft SQL Server nchar and nvarchar multibyte character datatypes will be replicated as char or varchar single-byte datatypes.

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

Replicating *ntext* datatypes

Because Replication Agent 12.6 does not support the Replication Server 15.0 UNITEXT datatype, the Replication Agent's encoding of the character stream affects the byte order of data when it is applied at the replicate database. You can set the lr_ntext_byte_order property to specify the byte order that Replication Agent uses on ntext data to either BigEndian or LittleEndian.

For example, if the replicate database is on UNIX, set the property to big; if the replicate database is on Windows, set the property to little. The default value of lr_ntext_byte_order is big.

Objects with delimited identifiers

When marking an object that uses delimited identifiers, the marking script uses left and right brackets instead of double quotes to delimit the identifiers. This allows you to mark delimited names without requiring that the SQL Server database have the QUOTED_IDENTIFIER option turned on. However, the pdb_setreptable command itself still requires using double quotes around the identifiers.

For example:

```
pdb_setreptable "My Table", mark
```

appears in the marking script as `[owner].[My Table]`

Replication Agent for Microsoft SQL Server transaction log

Replication Agent uses its own proprietary transaction log to capture and record transactions in the primary database for replication. The Replication Agent transaction log consists of a set of shadow tables, stored procedures, and triggers that are created in the primary database.

The Replication Agent transaction log is created by invoking the pdb_xlog command with the create keyword. When you invoke this command, Replication Agent generates a SQL script that is run in the primary database. This script (stored in the *create.sql* file in the *RAX-12_6\inst_name\scripts\xlog* directory) creates the Replication Agent transaction log components referred to as the *transaction log base objects*. The transaction log base objects must be created before any objects can be marked for replication in the primary database.

Note This section describes the schema and details of the Replication Agent transaction log for a Microsoft SQL Server database. For more general information about the Replication Agent transaction log, see the Sybase Replication Agent *Administration Guide*.

Transaction log components

There are three types of Microsoft SQL Server database objects used for transaction log components:

- Tables
- Stored procedures
- Triggers

There are several tables used by the Replication Agent transaction log, both as base objects and as shadow tables for marked primary tables in the primary database. All table components of the transaction log contain at least one index, and some contain more than one index.

Stored procedures are used by the Replication Agent transaction log to capture transactions for replication.

Triggers are used to capture the insert, update, and delete operations in marked primary tables.

Transaction log 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.

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

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

Transaction log base tables

Table 3-3 lists the tables that are considered Replication Agent transaction log base objects. No permissions are granted on these tables when they are created.

Table 3-3: Replication Agent transaction log base tables

Description	Database name
Transaction log system table	*prefix*xlog_system_
Marked objects table	*prefix*marked_objs_*xxx*
Transaction log table	*prefix*tran_log_*xxx*
LOB columns table	*prefix*blob_column_*xxx*
Exception holding table	*prefix*exception_*xxx*
Transaction sequence counter table	*prefix*tran_seq_*xxx*
Procedure sequence counter table	*prefix*proc_seq_*xxx*
Log Reader work table	*prefix*lr_reptran_*xxx*
Log Admin work table	*prefix*rawork_*xxx*
Transaction active table	*prefix*tranactive_*xxx*
Procedure active table	*prefix*procactive_*xxx*

Transaction log stored procedures

Table 3-4 lists the stored procedures that are considered Replication Agent transaction log base objects. Execute permission is granted to public when these stored procedures are created.

Note The transaction log stored procedures listed in Table 3-4 have no effect when executed outside the context of replication.

Table 3-4: Replication Agent transaction log stored procedures

Procedure	Database name
Transaction information capture	*prefix*get_tx_info_*xxx*
Log truncation	*prefix*truncate_logs_*xxx*

Marker procedures and shadow tables

Table 3-5 lists the marker procedures and marker shadow tables that are considered Replication Agent transaction log base objects. No permissions are granted when these procedures and tables are created.

Table 3-5: Replication Agent marker procedures and shadow tables

Procedure/Table	Database name
Transaction log-marker procedure	rs_marker*xxx*
Transaction log-marker shadow table	*prefix*markersh_*xxx*
Dump marker procedure	rs_dump*xxx*
Dump marker shadow table	*prefix*dumpsh_*xxx*

Transaction log objects for each marked table

Table 3-6 lists the Replication Agent transaction log objects that are created for each primary table that is marked for replication. These objects are created only when a table is marked for replication. These objects are not considered transaction log base objects.

Note The transaction log stored procedures listed in Table 3-6 have no effect when executed outside the context of replication. See "Marked objects table" on page 69.

Table 3-6: Replication Agent transaction log objects for each marked table

Marked table object	Database name
Shadow table	*prefix*sh_*xxx*
Shadow row procedure	*prefix*srp_*xxx* (public execute permission)
LOB shadow table	*prefix*bsh_*xxx*
LOB shadow row procedure	*prefix*bsrp_*xxx* (public execute permission)
Insert trigger	*prefix*instrg_*xxx* (or existing-trigger name)
Update trigger	*prefix*updtrg_*xxx* (or existing-trigger name)
Delete trigger	*prefix*deltrg_*xxx* (or existing-trigger name)

Note The marked objects table contains the mapping between marked objects and their corresponding shadow tables and shadow row procedures.

Reserved names

When a primary table is marked for replication, Replication Agent creates a shadow table to record the replicated operations. The shadow table has all the columns of the primary table, plus three columns that the Replication Agent creates for its use:

- ra_tran_id_
- ra_opid_
- ra_img_type_

Note These column names are fixed, not generated; therefore, the "ra_" portion of the column name does not change when the value of the pdb_xlog_prefix parameter changes.

If a primary table has a column with the same name as one of the Replication Agent shadow table columns, the table marking procedure fails. Replication Agent flags the conflicting column in the table-marking script and the pdb_setreptable command returns an error.

In this event, you must change the name of the conflicting column in the primary table in order to replicate changes to that table. After you change the name of the conflicting column, you can modify the table-marking script and run it manually to mark the primary table.

Note If you want to use the primary table's original column name in the replicate table, you can create a function string in the replicate Replication Server to map the new column name in the primary table to a different column name in the replicate table.

Marked table triggers

If a trigger exists on a primary table when that table is marked for replication, the existing trigger is modified to support Sybase replication.

The Sybase replication action in a trigger begins and ends with comments that serve as markers so that, when the table is unmarked, the Sybase replication action can be removed from the trigger.

Note Do not remove or alter the Sybase replication comments in triggers.

Microsoft SQL Server 7.0 and Microsoft SQL Server 2000 allow multiple triggers of each type to be created on a table. If more than one trigger of a given type exists on a table that you mark for replication, the first trigger is the one modified for Sybase replication.

Getting actual names of the transaction log objects

Because Sybase Replication Agent generates the names of its transaction log objects using its own algorithms, there is not any obvious correlation between the name of a primary object and the names of the transaction log objects (such as shadow tables) that record replicated operations for the primary object. If you want to find out the names of transaction log objects associated with a primary object, use the following procedure.

❖ **To determine the names of transaction log objects associated with a primary object**

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

```
pdb_xlog
```

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

2 At the Replication Agent administration port, invoke the pdb_setrepproc command with no keywords:

```
pdb_setrepproc
```

The pdb_setrepproc command returns a list of all stored procedures marked for replication, including all the Replication Agent transaction log objects associated with each marked procedure.

3 At the Replication Agent administration port, invoke the pdb_setreptable command with no keywords:

```
pdb_setreptable
```

The pdb_setreptable command returns a list of all tables marked for replication, including all the Replication Agent transaction log objects associated with each marked table.

Marked objects table

One of the Replication Agent transaction log base objects is the **marked objects table**. The marked objects table contains an entry for each marked object in the primary database (both tables and stored procedures). Each marked object entry contains the following information:

- Name of the marked primary object
- Primary object's replicated name (if any)
- Type of the primary object

- "Replication enabled" flag for the primary object
- Name of the shadow table for the primary object
- Name of the operation-image procedure
- Name of the LOB shadow table (if the primary object is a LOB column)
- Name of the LOB image procedure (if the primary object is a LOB column)
- Owner of the primary object
- "Send owner" flag

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

Sybase Replication Agent does not support backing up and restoring the transaction log automatically.

Sybase recommends that you use the database backup utilities provided with your Microsoft SQL Server software to periodically back up the transaction log.

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

Truncating the transaction log

Sybase 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.

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.

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.

If you want to truncate the transaction log at a specific time, you can use a scheduler utility to execute the pdb_truncate_xlog command automatically.

Replication Agent for Microsoft SQL Server setup test scripts

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

The Replication Agent test scripts are located in the *scripts* subdirectory under the Replication Agent base directory. For example: *RAX-12_6\scripts*.

The Replication Agent test scripts perform the following tasks:

1 Create a primary table.

2 Create a replicate table that corresponds to the primary table.

3 Create the primary data server connection in the primary Replication Server.

4 Create the replication definition in the primary Replication Server.

5 Test the replication definition.

6 Create the subscription in the replicate Replication Server.

7 Test the subscription.

8 Create the Replication Agent transaction log in the primary database.

9 Modify the primary table.

10 Clean up and remove the replication test environment created by the other scripts.

Before you begin

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

- Installed a Microsoft SQL Server data server
- Installed a data server to act as a replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database
- Installed the Replication Agent software, created a Replication Agent instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, lower
  - ra_config rs_source_ds, rax
  - ra_config rs_source_db, test

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

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

A word about the isql tool

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

Although the name of the Microsoft isql tool is the same as the Sybase tool called isql, the Sybase and Microsoft tools are not compatible. For example, you cannot use the Sybase isql tool to access the Microsoft SQL Server data server, and you cannot use the Microsoft isql tool to access the Replication Agent administration port.

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

Create the primary table

Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the Microsoft SQL Server data server and execute the *mssql_create_test_primary_table.sql* script to create the primary table in the primary database.

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

Create the replicate table

Use isql or another query processor to log in to the replicate data server and execute the *ase_create_test_replicate_table.sql* script to create the replicate table in the replicate database.

Create the Replication Server connection for Replication Agent

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_connection.sql* script to create the Replication Server connection to the primary database.

Create the replication definition

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_repdef.sql* script to create a test replication definition.

Test the replication definition

Use isql or another query processor to log in to the RSSD of the primary Replication Server and execute the *rssd_helprep_test_repdef.sql* script to test the replication definition.

Create the subscription

There are two scripts provided for this step:

- *rs_create_test_sub.sql* – designed for use with Replication Server version 11.5 or later.
- *rs_create_test_sub_for_11.0.x.sql* – designed for use with Replication Server version 11.0.x.

❖ **To create the test subscription**

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

---

1 Edit the appropriate script file (*rs_create_test_sub.sql* or *rs_create_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to ase.test.

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

Note If you are using the *rs_create_test_sub_for_11.0.x.sql* script, the script must be executed twice because, even though the PRS and RRS are the same Replication Server, there is not enough time between the command executions in the script to allow Replication Server to complete all its tasks before the next script command is executed. For this reason, you may wish to execute the commands in this script separately.

Test the subscription

Use isql or another query processor to access the RSSD of the replicate Replication Server and execute the *rssd_helpsub_test_sub.sql* script to test the subscription.

Create the Replication Agent transaction log

If you have not already created the Replication Agent transaction log in the primary database, create the transaction log now.

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

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

2 Use the following command to create the transaction log:

```
pdb_xlog create
```

See the Sybase Replication Agent *Administration Guide* for more information about creating the Replication Agent transaction log.

Mark a primary table for replication

Mark the test primary table (rax_test) that was created in the Microsoft SQL Server database by the *mssql_create_test_primary_table.sql* script.

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

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

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

```
pdb_setreptable rax_test, mark
```

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

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

Start replication

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

❖ **To start test replication**

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

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

```
resume
```

See the Sybase Replication Agent *Administration Guide* for more information about starting replication.

Execute the test transaction script in Microsoft SQL Server

Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the Microsoft SQL Server data server and execute the *mssql_primary_test_transactions.sql* script to generate transactions in the primary table in the primary database.

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

Check results of replication

Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the Microsoft SQL Server data server and execute the *mssql_select_test_primary.sql* script to view the changes in the test primary table in the primary database.

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

Use isql or another query processor to log in to the replicate database and execute the *ase_select_test_replicate.sql* script to verify that the transactions generated in the primary database were replicated to the test replicate table in the replicate database.

Clean up the test environment

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

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

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

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

```
quiesce
```

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

```
pdb_xlog remove, force
```

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

- *rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql* (to drop the test subscription)

Edit the appropriate script file (*rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used in the *rs_create_test_sub.sql* script. These values are initially set to ase.test.

- *rs_drop_test_repdef.sql* (to drop the test replication definition)
- *rs_drop_test_connection.sql* (to drop the test connection to the primary database)

5 Log in to the replicate data server and execute the *ase_drop_test_replicate_table.sql* script to drop the test replicate table.

6 Use your Microsoft SQL Server database access tool (such as Microsoft isql) to log in to the Microsoft SQL Server data server and execute the *mssql_drop_test_primary_table.sql* script to drop the test primary table.

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

CHAPTER 4

Replication Agent for Oracle

The term “Replication Agent for Oracle” refers to an instance of the Sybase Replication Agent version 12.6 software installed and configured for a primary database that resides in an Oracle data server.

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

Note For information on the basic functionality of Sybase Replication Agent version 12.6, refer to the Sybase Replication Agent *Administration Guide* and *Reference Manual*.

Oracle-specific issues

This section describes general issues and considerations that are specific to using Sybase Replication Agent version 12.6 with the Oracle data server.

The following topics are included in this section:

- Replication Agent connectivity
- Replication Agent permissions
- Redo log setup
- Setting ddl_username and ddl_password
- Character case of database object names

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

Replication Agent connectivity

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

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

The TNS Listener Service must be installed and running on the primary database so the Replication Agent instance can connect to it. See the Oracle Networking document for more information.

Replication Agent permissions

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

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

- drop public synonym – required to drop created synonyms.
- select on SYS.OBJ$ – required to process procedure DDL commands.
- select on SYS.LOB$ – required to support LOB replication.
- select on SYS.COLLECTION$ – required to support table replication.
- select on SYS.COL$ – required to support table replication.

In addition, the user who starts the Replication Agent for Oracle instance must have read access to the Oracle *redo* logs.

Redo log setup

Note The Replication Agent for Oracle *must* be installed on a machine where it can directly access the Oracle redo logs.

Replication Agent for Oracle requires that automatic archiving of Oracle redo logs be disabled. Archiving is performed manually by the Replication Agent as the data in the redo logs ia replicated. The following tasks should not be performed until the Replication Agent is ready to be initialized, so that archiving of redo logs will continue to take place.

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

- Redo log archiving must be enabled. To enable redo log archiving, execute the following command:

  ```
  alter database ARCHIVELOG;
  ```

  To verify that log archiving is enabled, execute the following query:

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

  If ARCHIVELOG is returned, then log archiving is enabled.

- 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 Oracle command:

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

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

You can change the LOG_ARCHIVE_START parameter by manually editing the server's startup parameter file, or with the following Oracle command:

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

To check the setting of the LOG_ARCHIVE_START parameter, execute the following query:

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

If FALSE is returned, then the value in the server parameter file has been correctly modified to prevent automatic archiving when you re-start the Oracle server.

For more information about the LOG_ARCHIVE_START parameter, or the ALTER SYSTEM commands, see the Oracle database reference guide.

After Replication Agent for Oracle is initialized, automatic archiving must never be enabled, even temporarily. If automatic archiving is re-enabled, or manual archiving is performed, causing a redo log file not yet processed by the Replication Agent to be overwritten, then the data in the lost redo log file will not be replicated. To recover from this situation requires that the replicate be rematerialized.

- In Oracle release 9.2 and later, supplemental logging of primary key data and index columns must be enabled. To enable supplemental logging, execute the following Oracle command:

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

  To verify that supplemental logging of primary key information is enabled, execute the following query:

  ```
  select SUPPLEMENTAL_LOG_DATA_PK from v$database;
  ```

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

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;
```

Setting *ddl_username* and *ddl_password*

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

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

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

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

DDL commands and objects filtered from replication

The following DDL commands are not replicated:

- alter database
- create database link
- drop database link
- alter session
- create snapshot
- create snapshot log
- alter snapshot
- alter snapshot log
- drop snapshot
- drop snapshot/log
- alter rollback segment
- create rollback segment
- drop rollback segment

alter system switch log
create control file
create pfile from spfile
create schema authorization
create spfile from pfile
explain
lock table
rename
set constraints
set role
set transaction
analyze
audit
no audit
create tablespace
alter tablespace
drop tablespace

The following objects are not replicated:

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

Character case of database object names

Database object names must be delivered to the primary Replication Server in the same format as they are specified in replication definitions. 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.

Note Replication will fail if a database object name is delivered to the primary Replication Server in a format different from that specified in the replication definition.

Sybase Replication Agent version 12.6 gives you some control over the way it treats the character case of database object names when it sends LTL to the primary Replication Server. You have three options:

- asis – 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.

You specify the character case option you want by setting the value of the ltl_character_case configuration parameter. The parameter values are asis (the default), lower, and upper.

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

See the following examples using the asis option:

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

See the following examples using the upper option:

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

Format of origin queue ID

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

Table 4-1 illustrates the format of the origin queue ID for the Replication Agent for Oracle.

Table 4-1: Replication Agent for Oracle origin queue ID

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

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 4-2 describes the conversion of Oracle datatypes to Sybase datatypes.

Table 4-2: Oracle to Sybase datatype mapping

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
CHAR	255 bytes	char	32K	
DATE	8 bytes, fixed-length, default format: DD-MON-YY	datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD. The default value replicated is: YYYYMMDD. If pdb_convert_datetime is true, the value replicated is YYYYMMDD HH:MM:SS.sss
TIMESTAMP(n)	21-31 bytes, variable-length, default format: DD-MON-YY hh.mm.ss.ffffff AM	datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD.

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
TIMESTAMP(n) WITH [LOCAL] TIME ZONE	Variable-length, default format: DD-MON-YY hh.mm.ss.ffffff AM {+|-}hh:mm	varchar(100)		
INTERVAL YEAR(n) TO MONTH	Variable-length	varchar(25)		
INTERVAL DAY(n) TO SECOND(n)	Variable-length	varchar()		
LONG	2GB, variable-length character data	text		
LONG RAW	2GB, variable-length binary data	image		
BLOB	4GB, variable-length binary large object	image		
CLOB	4GB, variable-length character large object	text		
NCHAR	255 bytes, multi-byte characters	unichar or char	32K	
NCLOB	4GB, variable-length multi-byte character large object	text		
NVARCHAR2	2000 bytes, variable-length, multi-byte character data	univarchar or varchar	32K	
BFILE	4GB, locator points to large binary file	image		
MLSLABEL	5 bytes, variable-length binary OS label			Not supported.

Oracle datatype	Oracle length/range	Sybase datatype	Sybase length/range	Notes
NUMBER (p,s)	21 bytes, variable-length numeric data	float, int, real, number, or decimal	float is 4 or 8 bytes. int is 4 bytes. real is 4 bytes. number and decimal are 2 to 17 bytes.	The float datatype can convert to scientific notation if the range is exceeded. Integers (int) are truncated if they exceed the Replication Server range of 2,147,483,647 to -2,147,483,648 or $1x10^{-130}$ to $9.99x10^{25}$. The number and decimal datatypes are truncated if they exceed the range of -10^{38} to $10^{38}-1$. Oracle precision ranges from 1 to 38 digits. Default precision is 18 digits. Oracle scale ranges from -84 to 127. Default scale is 0.
RAW	2000 bytes, variable-length binary data	binary or varbinary	32K	
ROWID	6 bytes, binary data representing row addresses	char	32K	
UDD object type	variable length character data	user-defined Replication Server datatype	32K	See “Oracle user-defined types” on page 93
VARCHAR2	2000 bytes, variable-length character data	varchar	32K	

Oracle datatype restrictions

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

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

- In the *integer* representation:

    - The corresponding Sybase int datatype has a smaller absolute maximum value.

      The Oracle NUMBER absolute maximum value is 38 digits of precision, between 9.9×10^{125} and 1×10^{-130}. The Sybase int value is between $2^{31} - 1$ and -2^{31} (2,147,483,647 and -2,147,483,648), inclusive.

    - Oracle NUMBER values greater than the Sybase int maximum are rejected by Replication Server.

- In the *floating point* representation:

    - The precision of the floating point representation has the same range limitation as the integer representation.

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

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

- When a TIMESTAMP WITH TIME ZONE datatype is replicated, the time zone information is used to resolve the timestamp value to the ‘local’ time zone and then the resolved value is replicated. The time zone information itself is not replicated.

- As an 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’ timezone is -6:00, the value replicated will be ‘01-JAN-05 11:00:00.000000’. The timestamp value is adjusted for the difference between the recorded timezone of -8:00 and the local time zone of -6:00, and the adjusted value is replicated.

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

- Oracle BLOB, CLOB, NCLOB, and BFILE datatypes that contain more than 2GB will be truncated to 2GB.

- Oracle CHAR, RAW, ROWID, and VARCHAR2 datatypes that contain more than 255 bytes will be truncated to 255 bytes.

- Oracle NCHAR and NVARCHAR2 multibyte character datatypes are replicated as char or varchar single-byte datatypes.

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

The following Oracle datatypes are not supported:

- Oracle REF Type
- Oracle VARRAY Type
- Oracle NESTED TABLE Type
- Oracle Supplied types

See Also

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

Oracle large object (LOB) support

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

- character
  - LONG
  - CLOB
  - NCLOB
- binary
  - LONG RAW
  - BLOB
  - BFILE

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

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

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

Special handling for *off row* LOBS

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

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

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

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

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

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

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

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

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

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

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

- missing accompanying change to the same row:

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

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

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

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

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

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

Oracle user-defined types

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

Replication Agent for Oracle version 12.6 supports replication of user-defined object types. Object types are abstractions of real-world entities, such as purchase orders, that application programs deal with. An object type is a schema object with three kinds of components:

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

Replicating UDDs

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

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

❖ **To create a datatype definition in Replication Server**

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

1 Log into the RSSD.

2 Add a row to the rs_datatype table using the following example as a guide:

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

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

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

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

go
Delimiter Prefix   Translated                  Value Delimiter Postfix

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

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

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

Example

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

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

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

Note The ltl_character_case must be in upper case for this example.

Replication Agent objects in the Oracle primary database

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

Sybase 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-12_6\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 can *not* be run manually to initialize the primary database. This is also true for the procedure marking and unmarking scripts that are generated when you use pdb_setrepproc. Scripts are no longer generated when marking and unmarking tables with pdb_setreptable.

Replication Agent object names

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

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

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

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

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

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

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

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

```
pdb_xlog
```

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

Table objects

Table 4-3 lists the tables that are considered Replication Agent objects.

Table 4-3: Replication Agent transaction log base tables

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

Procedure objects

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

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

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

Sequences

Table 4-5 lists the Oracle sequences that are considered Replication Agent base objects.

Table 4-5: Replication Agent sequences

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

Marked procedures

Table 4-6 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 4-6: Replication Agent objects for each marked procedure

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

Archiving the Oracle redo log

To ensure that data in the Oracle redo logs is not archived and overwritten before it is processed for replication, Replication Agent for Oracle requires that automatic archiving by Oracle is disabled. The Oracle archiving process is handled by the following Replication Agent log truncation processing.

Sybase 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.

Transaction log truncation

Each time Replication Agent performs log truncation (either scheduled or manual) it issues 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.

If you want to truncate the transaction log at a specific time, you can use a scheduler utility to execute the pdb_truncate_xlog command automatically.

Replication Agent for Oracle setup test scripts

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

The Replication Agent test scripts are located in the *scripts* subdirectory under the Replication Agent base directory. For example: *RAX-12_6\scripts*.

The Replication Agent test scripts perform the following tasks:

1 Create a primary table.

2 Create a replicate table that corresponds to the primary table.

3 Create the primary data server connection in the primary Replication Server.

4 Create the replication definition in the primary Replication Server.

5 Test the replication definition.

6 Create the subscription in the replicate Replication Server.

7 Test the subscription.

8 Create the Replication Agent transaction log in the primary database.

9 Modify the primary table.

10 Clean up and remove the replication test environment created by the other scripts.

Before you begin

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

- Installed an Oracle data server
- Installed a data server to act as a replicate data server
- Created the replicate database in the replicate data server
- Installed primary and replicate Replication Servers (PRS and RRS)

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

- Added the replicate database as an RRS-managed database
- Installed the Replication Agent software, created a Replication Agent instance, and used the ra_config command to set the following configuration parameters:
  - ra_config ltl_character_case, lower
  - ra_config rs_source_ds, rax
  - ra_config rs_source_db, test

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

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

Create the primary table

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_create_test_primary_table.sql* script to create the primary table in the primary database.

Create the replicate table

Use isql or another query processor to log in to the replicate data server and execute the *ase_create_test_replicate_table.sql* script to create the replicate table in the replicate database.

Create the Replication Server connection for Replication Agent

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_connection.sql* script to create the Replication Server connection to the primary database.

Create the replication definition

Use isql or another query processor to log in to the primary Replication Server and execute the *rs_create_test_repdef.sql* script to create a test replication definition.

Test the replication definition

Use isql or another query processor to log in to the RSSD of the primary Replication Server and execute the *rssd_helprep_test_repdef.sql* script to test the replication definition.

Create the subscription

There are two scripts provided for this step:

- *rs_create_test_sub.sql* – designed for use with Replication Server version 11.5 or later.

- *rs_create_test_sub_for_11.0.x.sql* – designed for use with Replication Server version 11.0.x.

❖ **To create the test subscription**

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

1 Edit the appropriate script file (*rs_create_test_sub.sql* or *rs_create_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used to create the connection to the replicate data server and replicate database. These values are initially set to ase.test.

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

Note If you are using the *rs_create_test_sub_for_11.0.x.sql* script, the script must be executed twice because, even though the PRS and RRS are the same Replication Server, there is not enough time between the command executions in the script to allow Replication Server to complete all its tasks before the next script command is executed. For this reason, you may want to execute the different commands in this script separately.

Test the subscription

Use isql or another query processor to access the RSSD of the replicate Replication Server and execute the *rssd_helpsub_test_sub.sql* script to test the subscription.

Create the Replication Agent transaction log

If you have not already created the Replication Agent transaction log in the primary database, create the transaction log now.

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

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

2 Use the following command to create the transaction log:

```
pdb_xlog init
```

See the Sybase Replication Agent *Administration Guide* for more information about creating the Replication Agent transaction log.

Mark a primary table for replication

Mark the test primary table that was created in the Oracle database by the *oracle_create_test_primary_table.sql* script (rax_test).

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

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

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

```
pdb_setreptable rax_test, mark
```

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

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

Start replication

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

❖ **To start test replication**

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

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

```
resume
```

See the Sybase Replication Agent *Administration Guide* for more information about starting replication.

Execute the test transaction script in Oracle

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_primary_test_transactions.sql* script to generate transactions in the primary table in the primary database.

Check results of replication

Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_select_test_primary.sql* script to view the changes in the test primary table in the primary database.

Use isql or another query processor to log in to the replicate database and execute the *ase_select_test_replicate.sql* script to verify that the transactions generated in the primary database were replicated to the test replicate table in the replicate database.

Clean up the test environment

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

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

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

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

```
quiesce
```

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

```
pdb_xlog remove, force
```

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

- *rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql* (to drop the test subscription)

Edit the appropriate script file (*rs_drop_test_sub.sql* or *rs_drop_test_sub_for_11.0.x.sql*) so that the values for *RDS.RDB* on the with replicate at clause for each command match the *RDS.RDB* values that you used in the *rs_create_test_sub.sql* script. These values are initially set to ase.test.

- *rs_drop_test_repdef.sql* (to drop the test replication definition)
- *rs_drop_test_connection.sql* (to drop the test connection to the primary database)

5 *ase_drop_test_replicate_table.sql*Log in to the replicate data server and execute the script to drop the test replicate table.

6 Use your Oracle database access tool (such as sqlplus) to log in to the Oracle data server and execute the *oracle_drop_test_primary_table.sql* script to drop the test primary table.

APPENDIX A

Migration in Replication Agent

This appendix describes how to upgrade and downgrade to different versions of Sybase Replication Agent.

Migrating from version 12.5 to 12.6

Sybase Replication Agent 12.6 requires a JDBC version 2.0 or 3.0 driver to connect with the primary data server. Previous versions of Sybase Replication Agent used JDBC version 1.2 drivers. You must make sure that the CLASSPATH environment variable on the Replication Agent host machine points only to the correct JDBC 2.0 or 3.0 driver for your primary data server.

Note If the CLASSPATH variable points to both a JDBC 2.0 driver and a JDBC 1.2 driver, Sybase Replication Agent may return errors when it attempts to connect to the primary data server.

If you are migrating from Sybase Replication Agent version 12.5 to 12.6 on Oracle, see “Migrating from Replication Agent for Oracle 12.5 to 12.6” on page 110.

If you are migrating from Sybase Replication Agent version 12.5 to 12.6, and you are not changing the primary database version, refer to “Migrating from version 12.5 to 12.6 for non-Oracle instances” on page 108.

If you are using Sybase Replication Agent and you migrate the primary database from Microsoft SQL Server 7.0 to SQL Server 2000 (8.0), refer to “Migrating from SQL Server 7.0 to SQL Server 2000” on page 119.

Migrating from version 12.5 to 12.6 for non-Oracle instances

Migration from Sybase Replication Agent version 12.5 to 12.6 is a simple, three-step process:

1 Install the Sybase Replication Agent version 12.6 software.

2 Copy the existing Replication Agent version 12.5 instance directories to the version 12.6 directory.

3 Set rs_charset.

Using this process, new Replication Agent version 12.6 instances will have the same configuration, including instance names and administrative user IDs and passwords, as existing version 12.5 instances. The following procedure describes the details of this process.

❖ **To migrate from Replication Agent version 12.5 to 12.6**

1 Install the Sybase Replication Agent version 12.6 software, as described in "Installing the Sybase Replication Agent software" in the Sybase Replication Agent *Installation Guide*.

Note If you are installing the Replication Agent for Oracle version 12.6 software on a machine with an installation of the previous version, it must be installed in a different *SYBASE* directory.

After you complete the version 12.6 installation procedure, return to this section to continue the migration procedure.

2 Quiesce and shut down all Replication Agent version 12.5 instances.

3 For each existing Replication Agent version 12.5 instance, copy the instance directory and subdirectories to the Replication Agent version 12.6 directory structure.

- On Windows NT or Windows 2000, use Explorer to cut and paste the version 12.5 instance directory and all its contents into the version 12.6 base directory (*%SYBASE%\RAX-12_6*).
- On UNIX, use the following command at the operating system prompt:

```
cp -R $SYBASE/rax-12_5/inst_name $SYBASE/RAX-12_6
```

where *inst_name* is the name of the directory for an existing Replication Agent version 12.5 instance.

4 Edit the configuration file for each Replication Agent version 12.6 instance to change the log directory path reference to point to the new directory location.

The configuration file named *inst_name.cfg*, is located in the instance directory, *%SYBASE%\RAX-12_6\inst_name*, where *%SYBASE%* is the installation directory and *inst_name* is the name of the Replication Agent instance. In the following examples, the installation directory is *sybase* and the instance name is *repagent*.

Windows example:

Before:

```
log_directory=C:\\sybase\\rax-12_5\\repagent\\log
```

After:

```
log_directory=C:\\sybase\\RAX-12_6\\repagent\\log
```

Note When editing the configuration file on Windows 2000, you must retain the double backslash characters.

UNIX example:

Before:

```
log_directory=/sybase/rax-12_5/repagent/log
```

After:

```
log_directory=/sybase/RAX-12_6/repagent/log
```

Save each configuration file after you edit it.

5 Start up each new Replication Agent version 12.6 instance.

See "Starting the Replication Agent instance from the command line" in the Sybase Replication Agent *Installation Guide* for more information.

6 Set rs_charset to match the Replication Server character set.

7 Update the version of the Replication Agent objects in the database. Invoke the pdb_xlog command with no parameters.

Replication Agent checks the version of the Replication Agent objects in the database for compatibility, and makes any updates necessary to complete the migration process.

8 Invoke the resume command to resume replication.

9 Uninstall the Sybase Replication Agent version 12.5 software.

Migrating from Replication Agent for Oracle 12.5 to 12.6

❖ **To migrate from Replication Agent for Oracle 12.5 to 12.6**

1. Back up the existing Replication Agent for Oracle instance directory, containing the following configuration file: *$SYBASE/rax-12_5/<instance>/<instance>.cfg*.
2. Install the Replication Agent for Oracle 12.6 version software on a machine where it can directly read the Oracle redo logs.

   Note If you are installing the Replication Agent for Oracle version 12.6 software on a machine with an installation of the previous version, it must be installed in a different *SYBASE* directory.

3. Create a Replication Agent for Oracle version 12.6 instance with a different name and port number than the Replication Agent for Oracle version 12.5 instance. The port and port+1 must be unique on the machine.

   Do *not* start the instance.

4. Update the appropriate interfaces file (*interfaces* for UNIX, *sql.ini* for Windows) with the new instance name and new port number so that the migration script can ISQL in to the new Replication Agent for Oracle version 12.6 instance.
5. Update the Replication Agent for Oracle version 12.6 instance's configuration file by running the generation script at *$SYBASE/RAX-12_6/bin/gen_RAO_migrate_with_parms.ksh*:

   ```
   cd $SYBASE/RAX-12_6/bin
   ./gen_RAO_migrate_with_parms.ksh mySrcRao myuid
   mypwd /workdir/path/mySrcRao.cfg
   ../myTgtRao/myTgtRao.cfg
   ```

   where:

   - *mySrcRao* is the name of the *interfaces* or *sql.ini* file entry for the Replication Agent for Oracle version 12.5.
   - *myuid* is the user ID for logging in to the Replication Agent for Oracle version 12.5 instance.
   - *mypwd* is the password for logging in to the Replication Agent for Oracle version 12.5 instance. If there is no password, then use two double-quotes with nothing in between ("").

- */workdir* is the path name of a directory to use as a work area and where the *<src_instance>_migrate_<date>.cmds* migration file will be created.
- */path/mySrcRao.cfg* is the full path name of the Replication Agent for Oracle version 12.5 instance configuration file.
- *../myTgtRao/myTgtRao.cfg* is the path name of the Replication Agent for Oracle version 12.6 instance configuration file that was created in step 3.

The generation script copies to the Replication Agent for Oracle version 12.6 configuration file or creates parameter initialization commands in the migration file for most of the parameters in the Replication Agent for Oracle version 12.5 configuration file. The generated migration script is a file *(/<workdir>/<src_instance>_migrate_<date>.cmds).* It contains Replication Agent commands that you will later run against the Replication Agent for Oracle version 12.6 instance to initialize the primary database, to initialize the Replication Agent for Oracle version 12.6 (including incrementing the database generation ID), and to re-mark all the tables, procedures, and LOB columns that were marked in the Replication Agent for Oracle version 12.5 instance.

If the Replication Agent for Oracle version 12.5 and the Replication Agent for Oracle version 12.6 instances are on different machines and both configuration files cannot be accessed at the same time, copy the Replication Agent for Oracle version 12.5 configuration file to a location on the Replication Agent for Oracle version 12.6 instance's machine where it can be read by the generation script.

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

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

If the Replication Agent for Oracle version 12.6 instance is on a Windows machine that does not have Korn shell available, copy the generation script and the Replication Agent for Oracle version 12.6 instance's configuration file to a UNIX machine from which you can log in to the Replication Agent for Oracle version 12.5 instance. This copy of the Replication Agent for Oracle version 12.6 instance's configuration file is updated by the generation script. After it is updated, copy the configuration file back to the Replication Agent for Oracle version 12.6 instance directory.

Note After the migration script is generated, do *not* mark, unmark, enable, or disable any of the tables, LOB columns, or procedures, nor modify any parameters in the Replication Agent for Oracle version 12.5 instance. If you do, these changes will not be applied to the Replication Agent for Oracle version 12.6 instance.

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

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

7 If necessary, install the Oracle 9.2.0.5 JDBC driver for JDK 1.4 on the same machine where you installed Replication Agent for Oracle version 12.6, and add the JDBC driver's path to your CLASSPATH environment variable on this machine.

   Note No other Oracle drivers are allowed in the CLASSPATH.

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

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

10 Set rs_charset to match the Replication Server character set.

11 Test the primary database connection:

```
test_connection PDS
```

12 At the primary Oracle database, grant the Replication Agent for Oracle version 12.6 primary Oracle user (the user specified by the pds_username configuration parameter) the additional required privileges. See “Replication Agent permissions” on page 80.

13 Prevent users (other than the Replication Agent for Oracle version 12.6 user) from any further access to the primary database.

14 In the Replication Agent for Oracle version 12.5 instance, verify 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 the following statistics are zero:

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

   b Once they are all zero, note the `Last QID Sent` from the last set of statistics.

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

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

15 Quiesce the Replication Agent for Oracle version 12.5 instance.

16 In the Replication Agent for Oracle version 12.5 instance, remove the XLog:

```
pdb_xlog remove, force
```

17 Shut down the Replication Agent for Oracle version 12.5 instance.

18 In the primary Oracle database:

   a Enable supplemental logging of primary key data.

   b Enable archiving of redo logs.

   c Disable “auto” archiving of redo logs.

   d Alter system switch log file.

e Alter system archive log all.

See Chapter 4, “Replication Agent for Oracle” for details.

19 Run the migration script that was generated in step 5 against the Replication Agent for Oracle version 12.6 instance. For example:

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

This script initializes the primary database, initializes Replication Agent for Oracle version 12.6 (including incrementing the database generation ID), and re-marks all the tables, procedures, and LOB columns that were marked in the Replication Agent for Oracle version 12.5 instance.

20 Allow users access to the primary database.

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

```
rs_zeroltm source_ds, source_db
```

where:

- *source_ds* matches the Replication Agent for Oracle version 12.6 instance values for *rs_source_ds.*
- *source_db* matches the Replication Agent for Oracle version 12.6 instance values for *rs_source_db.*

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

22 In the Replication Agent for Oracle version 12.6 instance, resume replication.

23 Sybase recommends that you change the administration user ID and password in the Replication Agent for Oracle version 12.6 instance from the default values to the same values you used in the Replication Agent for Oracle version 12.5 instance.

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

25 Update the *interfaces* or *sql.ini* file entries if you want the Replication Agent for Oracle version 12.5 instance name associated with the Replication Agent for Oracle version 12.6 instance machine and port number.

Downgrading from Replication Agent for Oracle 12.6

This procedure assumes you have followed the migration instructions above and are using a Replication Agent for Oracle version 12.6 instance. If the Replication Agent for Oracle version 12.5 instance no longer exists, create one using the Replication Agent for Oracle version 12.5 ra_admin or administrator command, then follow the procedure described below, using the *$SYBASE/RAX-12_6/bin/gen_RAO_migrate_with_parms.ksh* script described above, *instead of* the *$SYBASE/RAX-12_6/bin/gen_RAO_migrate.ksh* script described below.

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

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

❖ **To downgrade from Replication Agent for Oracle version 12.6**

1 Generate the downgrade script by running the *$SYBASE/RAX-12_6/bin/gen_RAO_migrate.ksh* file:

```
cd $SYBASE/RAX-12_6/bin
./gen_RAO_migrate.ksh mySrcRao myuid mypwd
/workdir
```

where:

- *mySrcRao* is the name of the *interfaces* or *sql.ini* file entry for the Replication Agent for Oracle version 12.6.
- *myuid* is the user ID for logging in to the Replication Agent for Oracle version 12.6 instance.
- *mypwd* is the password for logging in to the Replication Agent for Oracle version 12.6 instance. If there is no password, then use two double-quotes with nothing in between (“”).
- *workdir* is the path name of a directory to use as a work area and where the <*src_instance*>_*migrate*_<*date*>.*cmds* file will be created.

The script generates a file *(/<workdir>/<src_instance>_migrate_<date>.cmds)* of Replication Agent commands that you will later run against the Replication Agent for Oracle version 12.5 instance to initialize the primary database, to initialize the Replication Agent for Oracle version 12.5 (including incrementing the database generation id), and to re-mark all the tables, procedures, and LOB columns that were marked in the Replication Agent for Oracle version 12.6 instance.

The script does not modify any of the Replication Agent for Oracle version 12.5 parameters except for pdb_auto_run_scripts, pdb_dflt_column_repl, and pdb_convert_datetime, which will all be set to the same values as configured in the Replication Agent for Oracle version 12.6 instance.

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

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

After the migration script is generated, do *not* mark, unmark, enable, or disable any of the tables, LOB columns, or procedures, nor modify any parameters in the Replication Agent for Oracle version 12.6 instance. If you do, these changes will not be applied to the Replication Agent for Oracle version 12.5 instance.

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

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

3 Be sure that the appropriate Oracle JDBC driver is in your CLASSPATH. The one required for Replication Agent for Oracle version 12.6 will not work for older versions of Replication Agent for Oracle.

Note No other Oracle drivers are allowed in the CLASSPATH.

4 Start and log in to the Replication Agent for Oracle version 12.5 instance.

5 Test the primary database connection:

```
test_connection PDS
```

6 Prevent users (other than the Replication Agent for Oracle version 12.6 user) from any further access to the primary database.

7 Verify that the Replication Agent for Oracle version 12.6 instance is in REPLICATING state and allow replication to finish. To verify that replication has completed:

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

      • Input queue size

      • Output queue size

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

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

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

8 Quiesce the Replication Agent for Oracle version 12.6 instance.

9 In the Replication Agent for Oracle version 12.6 instance, de-initialize the Replication Agent:

```
pdb_xlog remove, force
```

10 Shut down the Replication Agent for Oracle version 12.6 instance.

11 Run the script that was generated in Step 1 above against the Replication Agent for Oracle version 12.5 instance. For example:

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

   where:

   • *myTgtRAO* is the name of the path to the Replication Agent for Oracle version 12.5 instance.

   • *workdir* is the path name of the directory that was used as a work area and where the *<src_instance>_migrate_<date>.cmds* file was created.

- *mySrcRao* is the name of the *interfaces* or *sql.ini* file entry for the Replication Agent for Oracle version 12.6.

The script initializes the primary database, the Replication Agent (including incrementing the database generation id), and re-marks all the tables, procedures, and LOB columns that were marked in the Replication Agent for Oracle version 12.6 instance.

12 Allow users access to the primary database.

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

```
rs_zeroltm source_ds, source_db
```

where *source_ds* and *source_db* match the Replication Agent for Oracle version 12.5 instance's values for the *rs_source_ds* and *rs_source_db* parameters.

Note The *rs_source_ds* and *rs_source_db* values were migrated from Replication Agent for Oracle version 12.6 and should *not* be changed.

14 In the Replication Agent for Oracle version 12.5 instance, resume replication.

15 Log out of the Replication Agent for Oracle version 12.5 instance.

16 If you created a new *interfaces* or *sql.ini* entry when you upgraded to Replication Agent for Oracle version 12.6, update entry so the Replication Agent for Oracle version 12.5 instance name is again associated with the old Replication Agent for Oracle version 12.5 instance machine and port number.

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

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

Migrating from SQL Server 7.0 to SQL Server 2000

Use the following migration procedure if you are using Sybase Replication Agent and you migrate the primary database from Microsoft SQL Server 7.0 to SQL Server 2000 (8.0).

Note If you are migrating from Sybase Replication Agent version 12.5 to 12.6, but you are *not* migrating from one version of SQL Server to another, use the migration procedure described in "Migrating from version 12.5 to 12.6 for non-Oracle instances" on page 108.

❖ **To migrate from SQL Server 7.0 to SQL Server 2000 with Replication Agent 12.6**

1. Quiesce the replication system and stop all transaction activity on the primary database.
2. Unmark all primary objects (tables and stored procedures) in the primary database.
3. Remove all Replication Agent transaction log objects from the primary database:

   ```
   pdb_xlog remove
   ```

4. Uninstall the earlier version of the Sybase Replication Agent software using the uninstall utility provided with the software.
5. Install the Sybase Replication Agent 12.6 software using the instructions in "Installing the Sybase Replication Agent software" in the Sybase Replication Agent *Installation Guide*.
6. Follow the procedures in Chapter 2 and Chapter 3 of the Sybase Replication Agent *Administration Guide* to set up the Replication Agent 12.6 software with your replication system.

Glossary

This glossary describes Replication Server—Heterogeneous Replication Options terms used in this book.

Adaptive Server

The brand name for Sybase relational database management system (RDBMS) software products.

- *Adaptive Server Enterprise* manages multiple, large relational databases for high-volume online transaction processing (OLTP) systems and client applications.
- *Adaptive Server IQ* manages multiple, large relational databases with special indexing algorithms to support high-speed, high-volume business intelligence, decision support, and reporting client applications.
- *Adaptive Server Anywhere* manages relational databases with a small DBMS footprint, which is ideal for embedded applications and mobile device applications.

See also **DBMS** and **RDBMS**.

atomic materialization

A materialization method that copies subscription data from a primary database to a standby database in a single, atomic operation. No changes to primary data are allowed until the subscription data is captured at the primary database. See also **bulk materialization** and **nonatomic materialization**.

BCP utility

A bulk copy transfer utility that provides the ability to load multiple rows of data into a table in a target database. See also **bulk copy**.

bulk copy

An Open Client interface for the high-speed transfer of data between a database table and program variables. It provides an alternative to using SQL insert and select commands to transfer data.

bulk materialization

A materialization method whereby subscription data in a standby database is initialized outside of the replication system. You can use bulk materialization for subscriptions to table replication definitions or function replication definitions. See also **atomic materialization** and **nonatomic materialization**.

client — In client/server systems, the part of the system that sends requests to servers and processes the results of those requests. See also **client application**.

client application — Software that is responsible for the user interface, including menus, data entry screens, and report formats. See also **client**.

commit — An instruction to the DBMS to make permanent the changes requested in a transaction. See also **transaction**. Contrast with **rollback**.

data client — A client application that provides access to data by connecting to a data server. See also **client**, **client application**, and **data server**.

data distribution — A method of locating (or placing) discrete parts of a single set of data in multiple systems or at multiple sites. Data distribution is distinct from data replication, although a data replication system can be used to implement or support data distribution. Contrast with **data replication**.

data replication — The process of copying data to remote locations, and then keeping the replicated data synchronized with the primary data. Data replication is distinct from data distribution. Replicated data is stored copies of data at one or more remote sites throughout a system, and it is not necessarily distributed data. Contrast with **data distribution**. See also **disk replication** and **transaction replication**.

data server — A server that provides the functionality necessary to maintain the physical representation of a table in a database. Data servers are usually database servers, but they can also be any data repository with the interface and functionality a data client requires. See also **client**, **client application**, and **data client**.

database — A collection of data with a specific structure (or schema) for accepting, storing, and providing data for users. See also **data server**, **DBMS**, and **RDBMS**.

database connection — A connection that allows Replication Server to manage the database and distribute transactions to the database. Each database in a replication system can have only one database connection in Replication Server. See also **Replication Server** and **route**.

datatype — A keyword that identifies the characteristics of stored information on a computer. Some common datatypes are: char, int, smallint, date, time, numeric, and float. Different data servers support different datatypes.

DBMS An abbreviation for *database management system*. A DBMS is a computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The DBMS can include the user interface for using the database, or it can be a stand-alone data server system. Compare with **RDBMS**.

disaster recovery A method or process used to restore the critical business functions interrupted by a catastrophic event. A disaster recovery (or business continuity) plan defines the resources and procedures required for an organization to recover from a disaster, based on specified recovery objectives.

failback A procedure that restores the normal user and client access to a primary database, after a failover procedure switched access from the primary database to a standby database. See also **failover**.

failover A procedure that switches user and client access from a primary database to a standby database, particularly in the event of a failure that interrupts operations at the primary database, or access to the primary database. Failover is an important fault-tolerance feature for systems that require high availability. See also **failback**.

function A Replication Server object that represents a data server operation such as insert, delete, or begin transaction. Replication Server distributes operations to standby databases as functions. See also **function string**.

function string A string that Replication Server uses to map a function and its parameters to a data server API. Function strings allow Replication Server to support heterogeneous replication, in which the primary and standby databases are different types, with different SQL extensions and different command features. See also **function**.

gateway Connectivity software that allows two or more computer systems with different network architectures to communicate.

inbound queue A stable queue managed by Replication Server to spool messages received from a Sybase 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, Sybase Replication Agent, and Replication Server. See also **Open Client** and **Open Server**.

Java An object-oriented programming language developed by Sun Microsystems. A platform-independent, "write once, run anywhere" programming language.

Java VM The Java Virtual Machine. The Java VM (or JVM) is the part of the Java Runtime Environment (JRE) that is responsible for interpreting Java byte codes. See also **Java** and **JRE**.

JDBC An abbreviation for *Java Database Connectivity*. JDBC is the standard communication protocol for connectivity between Java clients and data servers. See also **data server** and **Java**.

JRE An abbreviation for *Java Runtime Environment*. The JRE consists of the Java Virtual Machine (Java VM or JVM), the Java Core Classes, and supporting files. The JRE must be installed on a machine to run Java applications, such as the Sybase Replication Agent. See also **Java VM**.

LAN An abbreviation for "local area network." A local area network is a computer network located on the user's premises and covering a limited geographical area (usually a single site). Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary can be subject to some form of regulation. Contrast with **WAN**.

latency In transaction replication, the time it takes to replicate a transaction from a primary database to a standby database. Specifically, latency is the time elapsed between committing an original transaction in the primary database and committing the replicated transaction in the standby database.

In disk replication, latency is the time elapsed between a disk write operation that changes a block or page on a primary device and the disk write operation that changes the replicated block or page on a mirror (or standby) device.

See also **disk replication** and **transaction replication**.

LOB An abbreviation for *large object*. A LOB is a type of data element that is associated with a column that contains extremely large quantities of data.

Log Reader An internal component of the Sybase 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 Sybase Replication Agent that interacts with Replication Server to forward transactions for distribution to a standby database. See also **Log Reader** and **Log Transfer Manager**.

Log Transfer Manager — An internal component of the Sybase Replication Agent that interacts with the other Sybase Replication Agent internal components to control and coordinate Sybase Replication Agent operations. See also **Log Reader** and **Log Transfer Interface**.

Maintenance User — A special user login name in the standby database that Replication Server uses to apply replicated transactions to the database. See also **Replication Server**.

materialization — The process of copying the data from a primary database to a standby database, initializing the standby database so that the Replication Server—Heterogeneous Replication Options system can begin replicating transactions. See also **atomic materialization**, **bulk materialization**, and **non-atomic materialization**.

nonatomic materialization — A materialization method that copies subscription data without a lock on the primary database. Changes to primary data are allowed during data transfer, which may cause temporary inconsistencies between the primary and standby databases. Contrast with **atomic materialization**. See also **bulk materialization**.

ODBC — An abbreviation for *Open Database Connectivity*. ODBC is an industry standard communication protocol for clients connecting to data servers. See also **JDBC**.

Open Client — A Sybase product that provides customer applications, third-party products, and other Sybase products with the interfaces needed to communicate with Open Server applications. See also **Open Server**.

Open Client application — An application that uses Sybase Open Client libraries to implement Open Client communication protocols. See also **Open Client** and **Open Server**.

Open Server — A Sybase product that provides the tools and interfaces required to create a custom server. See also **Open Client**.

Open Server application — A server application that uses Sybase Open Server libraries to implement Open Server communication protocols. See also **Open Client** and **Open Server**.

outbound queue — A stable queue managed by Replication Server to spool messages to a standby database. See also **inbound queue** and **stable queue**.

primary data — The version of a set of data that is the source used for replication. Primary data is stored and managed by the primary database. See also **Sybase Replication Agent**, **primary database**, and **Replication Server**.

primary database — The database that contains the data to be replicated to another database (the standby database) through a replication system. The primary database is the database that is the source of replicated data in a replication system. Sometimes called the *active database*. Contrast with **standby database**. See also **primary data**.

primary key — The column or columns whose data uniquely identify each row in a table.

primary site — The location or facility at which primary data servers and primary databases are deployed to support normal business operations. Sometimes called the *active site* or *main site*. See also **primary database** and **standby site**.

primary table — A table used as a source for replication. Primary tables are defined in the primary database schema. See also **primary data** and **primary database**.

primary transaction — A transaction that is committed in the primary database and recorded in the primary database transaction log. See also **primary database**, **replicated transaction**, and **transaction log**.

quiesce — To cause a system to go into a state in which further data changes are not allowed. See also **quiescent**.

quiescent — In a replication system, a state in which all updates have been propagated to their destinations. Some Sybase 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*. RCL is the command language used to manage Replication Server.

RDBMS — An abbreviation for *relational database management system*. An RDBMS is an application that manages and controls relational databases. Compare with **DBMS**. See also **relational database**.

relational database A collection of data in which data is viewed as being stored in tables, which consist of columns (data items) and rows (units of information). Relational databases can be accessed by SQL requests. See also **SQL**.

replicated data A set of data that is replicated from a primary database to a standby database by a replication system. See also **primary database**, **replication system**, and **standby database**.

replicated transaction A primary transaction that is replicated from a primary database to a standby database by a transaction replication system. See also **primary database**, **primary transaction**, **standby database**, and **transaction replication**.

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

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

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

RSSD An abbreviation for *Replication Server System Database*. The RSSD manages replication system information for a Replication Server. See also **Replication Server**.

replication system A data processing system that replicates data from one location to another. Data can be replicated between separate systems at a single site, or from one or more local systems to one or more remote systems. See also **disk replication** and **transaction replication**.

rollback An instruction to a database to back out of the changes requested in a unit of work (called a transaction). Contrast with **commit**. See also **transaction**.

SQL An abbreviation for *Structured Query Language*. SQL is a non-procedural programming language used to process data in a relational database. ANSI SQL is an industry standard. See also **transaction**.

stable queue A disk device-based, store-and-forward queue managed by Replication Server. Messages written into the stable queue remain there until they can be delivered to the appropriate process or standby database. Replication Server provides a stable queue for both incoming messages (the inbound queue) and outgoing messages (the outbound queue). See also **database connection**, **Replication Server**, and **route**.

standby data The data managed by a standby database, which is the destination (or target) of a replication system. See also **data replication** and **standby database**.

standby database A database that contains data replicated from another database (the primary database) through a replication system. The standby database is the database that receives replicated data in a replication system. Sometimes called the *replicate database*. Contrast with **primary database**. See also **standby data**.

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

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

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

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

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

transaction replication A data replication method that copies data-changing operations from a primary database transaction log to a standby database. See also **data replication** and **disk replication**.

transactional consistency A condition in which all transactions in the primary database are applied in the standby database, in the same order that they were applied in the primary database.

WAN An abbreviation for “wide area network.” A wide area network is a system of local-area networks (LANs) connected together with data communication lines. Contrast with **LAN**.

Index

F

I

J

L

M

N

O

P

R

S

T

U

V

W