SYBASE®

Heterogeneous Replication Guide

Replication Server®

15.1

DOCUMENT ID: DC36924-01-1510-01

LAST REVISED: May 2008

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

Contents

About This Book

Replication Server® maintains replicated data at multiple sites on a network. Organizations with geographically distant sites can use Replication Server to create distributed database applications with better performance and data availability than a centralized database system can provide.

This book introduces heterogeneous replication concepts, and it addresses the issues peculiar to heterogeneous replication with Sybase replication technology.

In this book, the term *heterogeneous replication* refers to replication between different types of data servers (for example, replicating from an Oracle database to a Sybase Adaptive Server® Enterprise database).

This book also addresses the issues involved with replication between non-Sybase data servers of the same type (for example, replicating from one Oracle database to another Oracle database).

Audience

The Replication Server *Heterogeneous Replication Guide* is to be used to plan, design, implement, or maintain a Sybase replication system with heterogeneous or non-Sybase data servers.

If you are new to Replication Server, refer to the Replication Server *Design Guide* for an introduction to basic data replication concepts and Sybase replication technology.

How to use this book

This book is divided into four parts:

- Part 1, “Introduction,” provides an overview of data replication concepts and Sybase replication technology, and it introduces the issues specific to replication systems with heterogeneous or non-Sybase data servers. It contains the following chapters:
  - Chapter 1, “Replication System Overview,” introduces replication system concepts, with a focus on heterogeneous replication using Sybase replication technology.
  - Chapter 2, “Replication Components,” introduces the Sybase software products that you can use to implement a replication system with heterogeneous or non-Sybase data servers.

    - Chapter 3, “Heterogeneous Replication Issues,” describes the issues and problems that are peculiar to heterogeneous replication and that must be addressed in a successful replication system.
- Part 2, “Component-Specific Issues,” describes the issues you must consider for each component of a heterogeneous replication system and explains how to address those issues. It contains the following chapters:
    - Chapter 4, “Data Server Issues,” describes the issues and considerations for specific non-Sybase data servers in a Sybase replication system.
    - Chapter 5, “Replication Server Issues,” describes the issues related to using Sybase Replication Server in a replication system with heterogeneous or non-Sybase data servers.
- Part 3, “Implementing Heterogeneous Replication,” describes how to set up and maintain a replication system using Replication Server with heterogeneous or non-Sybase data servers. It contains the following chapters:
    - Chapter 6, “Replication System Configuration Examples,” describes several configuration options for a Sybase replication system with heterogeneous or non-Sybase data servers, and explains the issues involved with each configuration.
    - Chapter 7, “Administering Heterogeneous Replication Systems,” describes administration tasks for replication systems with heterogeneous or non-Sybase data servers.
    - Chapter 8, “Troubleshooting Heterogeneous Replication Systems,” describes common problems and troubleshooting procedures for replication systems with heterogeneous or non-Sybase data servers.
- Part 4, “Appendixes,” contains supplemental information in the following appendixes:
    - Appendix A, “Datatype Translation and Mapping,” lists the class-level datatype translations for all non-Sybase data servers supported by Replication Server. It also lists Replication Server datatype names for non-Sybase datatypes.
    - Appendix B, “Materialization Issues,” describes the materialization issues that you need to consider when implementing a replication system with heterogeneous or non-Sybase data servers.

- Appendix C, “Heterogeneous Database Reconciliation,” describes the issues involved with reconciling data from different databases in a replication system with heterogeneous or non-Sybase data servers.
- Appendix D, “Replication with SQL Anywhere,” describes primary and replicate data server issues for Adaptive Server Anywhere in a Sybase replication system.

Related documents

The Replication Server documentation set consists of the following:

- The Replication Server *Release Bulletin* for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the Replication Server *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.
- *Installation Guide* for your platform – describes installation and upgrade procedures for all Replication Server and related products.
- *What’s New in Replication Server?* – describes the new features in Replication Server version 15.1 and the system changes added to support those features.
- *Administration Guide* – contains an introduction to replication systems. This manual 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 all 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 heterogeneous 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.
- *Reference Manual* – contains the syntax and detailed descriptions of Replication Server commands in the Replication Command Language (RCL); Replication Server system functions; Sybase Adaptive Server commands, system procedures, and stored procedures used with Replication Server; Replication Server executable programs; and Replication Server system tables.

- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Available only in print version.
- *Troubleshooting Guide* – contains information to aid in diagnosing and correcting problems in the replication system.
- Replication Server plug-in help, which contains information about using Sybase Central™ to manage Replication Server.
- Additional Replication Server Options documents that may be helpful:
  - *Overview Guide* – describes components used for replicating to and from ASE and non-Sybase databases.
  - *Release Bulletin* Replication Server Options Version 15.1 for Linux, Microsoft Windows, and UNIX – describes issues for the Replication Server Options.
  - *Release Bulletin* Replication Agent Version 15.1 for Linux, Microsoft Windows, and UNIX – describes features and issues for Replication Agent™.
  - *Installation Guide* Replication Agent Version 15.1 – describes how to install Replication Agent.
  - Replication Agent *Administration Guide* – describes how to extend the capabilities of Replication Server to replicate from non-Sybase primary data servers in a Sybase replication system.
  - Replication Agent *Primary Database Guide* – describes the specific issues related to the non-Sybase primary data servers in a Sybase replication system.
  - Replication Agent *Reference Manual* – contains the syntax and detailed descriptions of the Replication Agent commands and configuration parameters.
  - Enterprise Connect™ Data Access Option for Oracle *Server Administration and User's Guide* – describes how to configure ECDA Option for Oracle.
  - Enterprise Connect Data Access Options *User's Guide* for Access Services – describes how to configure a DirectConnect™ access service to extend the capabilities of Replication Server to replicate into non-Sybase data servers in a Sybase replication system.
  - Enterprise Connect Data Access and Mainframe Connect Server *Administration Guide* – describes how to use a Sybase DirectConnect server.

Other sources of information

Use the Getting Started CD, the SyBooks CD, and the 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.

❖ **Finding 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 timeframe and then click Go.
4. Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1. Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

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

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

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

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

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

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

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

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

3 Select a product.

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

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

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

Conventions

The following style conventions are used in this manual:

- In a sample screen display, commands that you should enter exactly as shown are in:

```
this font
```

- In a sample screen display, words that you should replace with the appropriate value for your installation are shown in:

  this font

- In the regular text of this document, the names of files and directories appear like this:

  /usr/u/sybase

- In the regular text of this document, the names of programs, utilities, procedures, and commands appear like this:

  bcp

The conventions for syntax statements in this manual are as follows:

Table 1: SQL syntax conventions

Key	Definition
command	Command names, command option names, utility names, utility flags, and other keywords.
variable	Variables, or words that stand for values that you fill in.
{ }	Curly braces indicate that you must choose at least one of the enclosed options. Do not include braces in your option.
[ ]	Brackets mean that choosing one or more of the enclosed options is optional. Do not include brackets in your option.
( )	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.

Accessibility features

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

Replication Server 15.1 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

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

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

For a Section 508 compliance statement for Replication Server 12.6, see Sybase Accessibility at http://www.sybase.com/detail_list?id=52484.

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.

PART 1

Introduction

Chapters in this part introduce some data replication concepts and the Sybase replication technology that supports replication systems with heterogeneous or non-Sybase data servers.

- Chapter 1, “Replication System Overview,” introduces basic replication system concepts, with a focus on heterogeneous replication using Sybase replication technology.
- Chapter 2, “Replication Components,” introduces the Sybase software products that enable you to implement a heterogeneous replication system using Sybase replication technology.
- Chapter 3, “Heterogeneous Replication Issues,” describes the issues and problems that are specific to heterogeneous replication and that must be addressed in a successful heterogeneous replication system.

CHAPTER 1

Replication System Overview

This chapter introduces some basic replication system concepts, with a focus on heterogeneous replication using Sybase replication technology.

Basic replication concepts

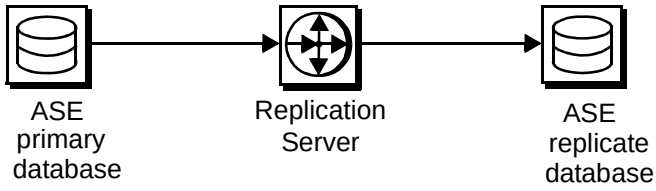
A basic Sybase replication system consists of three components:

- **Primary database** – a database in which original data-changing operations (or transactions) are performed and captured for replication
- **Replication Server** – a Sybase Open Client™ and Open Server™ product that receives transactions to be replicated from a primary database, and delivers them to a replicate database
- **Replicate database** – a database that receives replicated transactions from a Replication Server and applies those transactions to its own "copy" of the primary data

If both primary and replicate data servers are Adaptive Server Enterprise (ASE), you can implement a replication system with only these three components. Adaptive Server Enterprise includes all the features necessary to support a Sybase replication system, with no additional components other than the Replication Server.

Figure 1-1 illustrates a basic Sybase replication system, showing the flow of data between two Adaptive Servers and a Replication Server.

Figure 1-1: Basic Sybase replication system



For more information about basic Sybase replication system concepts and Replication Server features, see the first two chapters of the Replication Server *Administration Guide*.

Heterogeneous replication

The term *heterogeneous replication* refers to replicating data-changing operations between heterogeneous data servers (that is, data servers that are supplied by different vendors). For example:

- A replication system in which Adaptive Server Enterprise is either the primary or the replicate data server, and a non-Sybase data server (such as DB2 Universal Database) is the other data server.
- A replication system in which the primary and replicate data servers are different non-Sybase data servers (for example, Oracle is the primary data server and DB2 Universal Database is the replicate data server).

Note This book also addresses Sybase replication systems with no Adaptive Server, and just one type of non-Sybase data server (for example, a system in which both the primary and replicate data servers are Oracle). This is not a heterogeneous replication system because only one type of data server is used, but the issues that you must consider are the same as the heterogeneous replication issues.

Adaptive Server Enterprise is the data server that Sybase Replication Server was designed to work with. All of the data server elements required to support Replication Server (that is, a data-change capture mechanism in the primary database, and system tables and stored procedures in the replicate database) are

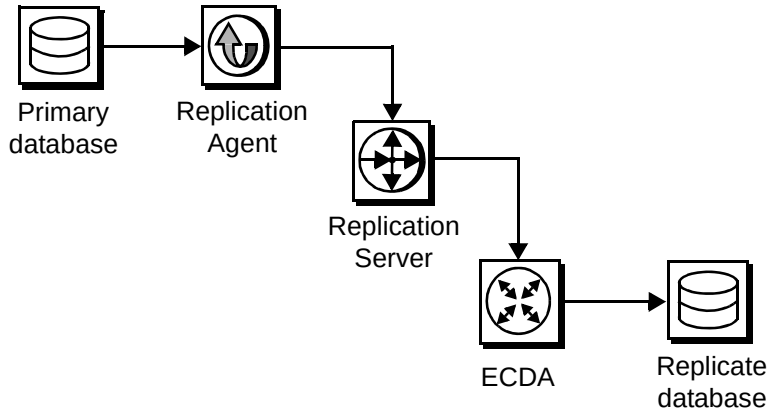
either built into Adaptive Server Enterprise or enabled by utilities that are provided with the Replication Server or Adaptive Server software.

Two additional components provided by Replication Server Options are required to implement a Sybase replication system with non-Sybase data servers:

- A Replication Agent™
- Enterprise Connect Data Access (ECDA)

Figure 1-2 illustrates a Sybase replication system with non-Sybase data servers, showing the flow of data between the data servers, through the Replication Agent, Replication Server, and Enterprise Connect Data Access database gateway.

Figure 1-2: Sybase replication system with non-Sybase data servers



Replication Agents support non-Sybase data servers by capturing the data-changing operations in the primary database and sending them to Replication Server for distribution.

ECDA database gateways support non-Sybase data servers by providing connectivity between Sybase Open Client and Open Server and either ODBC or the native protocol of the replicate data server, and by providing SQL transformation and other services.

Replication Server also includes features and utilities that support non-Sybase data servers. See “Replication Server heterogeneous datatype support” on page 121 for more details about Replication Server support for non-Sybase data servers.

Replication system components

This section describes the major components in a Sybase replication system. The following components are described by their function and role in the replication system:

- Primary data server
- Replication Agent
- Replication Server
- Database gateway
- Replicate data server

For a more complete description of the Sybase software products (Replication Server, Replication Agents, and Enterprise Connect Data Access gateways), see Chapter 2, “Replication Components.”

Primary data server

A **primary data server** is a server that manages one or more primary databases, which are the sources of the data-changing operations or transactions in a replication system.

Non-Sybase primary data servers are supported by Replication Agents, which access replication data in the primary database by reading the data server’s transaction log.

For more information about primary data servers, see “Primary data servers” on page 37.

Replication Agent

A **Replication Agent** transfers transaction information, which represents changes made to data, from a primary data server to a Replication Server, for distribution to other (replicate) databases.

For Sybase data servers (such as Adaptive Server Enterprise), an embedded Replication Agent is provided with the database management system software. The Replication Agent for Adaptive Server Enterprise is called RepAgent, and it is an Adaptive Server thread.

For non-Sybase data servers, Sybase provides two Replication Agent products:

- Replication Agent for DB2 UDB – provides primary data server support for DB2 Universal Database servers that run on IBM z/OS platforms
- Replication Agent – provides primary data server support for DB2 Universal Database, Microsoft SQL Server, and Oracle data servers that run on Linux, UNIX, or Microsoft Windows platforms

Replication Agents read the Primary Database transaction log. The primary Replication Server reconstructs the transaction and forwards it to replicate sites that have subscriptions for the data.

A Replication Agent is required for each database that contains primary data or for each database where replicated stored procedures are executed. A replicate database that contains only replicated data, and that has no replicated stored procedures, does not require a Replication Agent.

For more information about Replication Agents, see “Replication Agent” on page 15 and “Primary data servers” on page 37.

Replication Server

A **Replication Server** at each primary or replicate site coordinates data replication activities for local data servers and exchanges data with Replication Servers at other sites.

Replication Server performs the following major tasks:

- Receives data-changing operations from primary databases through a Replication Agent and distributes them to replicate database sites that have subscriptions for the data
- Receives data-changing operations from other Replication Servers and applies them to local replicate databases

- Provides guaranteed delivery of transactions to each replicate site

The information needed to accomplish these tasks is stored in Replication Server system tables. The system tables include descriptions of the replicated data and replication objects, such as replication definitions and subscriptions, security records for Replication Server users, routing information for other Replication Server sites, access methods for local databases, and other administrative information.

Replication Server system tables are stored in a database called the *Replication Server System Database* (RSSD). Each Replication Server has its own RSSD.

For more information about Replication Server, see "Replication Server" on page 7, and Chapter 5, "Replication Server Issues."

Database gateway

A **database gateway** allows clients using one communication protocol to connect with data servers that use a different protocol.

The Sybase Enterprise Connect Data Access product line consists of database gateway servers that allow clients using the Sybase Open Client and Open Server protocol (such as Replication Server) to connect with non-Sybase data servers, using either the data server's native communications protocol or the standard, ODBC protocol.

Sybase Enterprise Connect Data Access products also allow for the retrieval of metadata from non-Sybase replicate data servers.

For more information about Enterprise Connect Data Access database gateways and replicate data servers, see "Replicate data servers" on page 78.

Gatewayless connections

Sybase also provides the Mainframe Connect™ DB2 UDB Option for CICS product that allows "gatewayless" connections to DB2 Universal Database data servers on IBM z/OS platforms. In some environments, a gatewayless connection to the DB2 data server may be more efficient than a gateway server connection.

Replicate data server

A **replicate data server** manages a database that contains replicate data, which is data that is a "copy" of the data in a primary database.

Replication Server maintains the data in a replicate data server by logging in as a database user. In the case of non-Sybase data servers, Replication Server logs in to the replicate data server through a database gateway server.

Replication Server can treat any system as a data server if it supports a set of required data operations and transaction processing directives, either directly (such as Adaptive Server Enterprise) or indirectly (such as a Enterprise Connect Data Access database gateway server).

For more information about replicate data servers and Enterprise Connect Data Access database gateway servers, see “Replicate data servers” on page 78.

CHAPTER 2

Replication Components

This chapter describes the Sybase software products that allow you to implement a Sybase replication system with heterogeneous or non-Sybase data servers.

Sybase replication technology

Sybase replication technology solves several major problems that are not addressed by a traditional distributed database architecture:

- Corporate consolidation – Sybase replication technology allows you to maintain a corporate overview of distributed operations that is close to real time, even when distributed business units run on a variety of hardware and DBMS platforms.
- Decentralization – Sybase replication technology allows you to locate data where it is needed, making distributed business units much less vulnerable to central computer or network downtime while reducing overall communication costs. It also allows bidirectional data-sharing with a safe approach for replicating remote updates.
- High availability – Replication systems based on Sybase replication technology remain robust, despite typical hardware, software, and network failures, delivering applications with very high uptime at a reasonable cost.

- Live decision support – Replication systems based on Sybase replication technology can replicate an OLTP database, allowing analysts to run complex decision-support queries on data that is within seconds of real time, without affecting OLTP system performance.
- Disaster recovery – Sybase replication technology allows you to maintain a near-real-time "warm standby" database, to which applications can switch with virtually no downtime if the primary site fails.

Sybase replication products

Sybase offers product lines that specifically support replication systems with heterogeneous or non-Sybase data servers, based on Sybase replication technology:

- Replication Server, which is the centerpiece of Sybase's advanced replication technology and incorporates several features specifically to support non-Sybase data servers in a Sybase replication system.
- Replication Server Options that consist of a Replication Agent and an Enterprise Connect Data Access (ECDA):
  - Replication Agents support Replication Server by providing a way to obtain replication data from non-Sybase primary databases. Replication Agents provide this support for DB2 Universal Database, Microsoft SQL Server, and Oracle data servers.
  - ECDA database gateways support Replication Server by providing access to a variety of non-Sybase databases, allowing them to function as replicate databases in a Sybase replication system.

Replication Server

Replication Server can access data locally instead of from remote, centralized databases. Compared to a centralized data system, a replication system improves system performance and data availability, and reduces communication overhead. Replication Server provides a cost-effective, fault-tolerant system for replicating data.

Because Replication Server replicates transactions—incremental changes instead of data copies—and stored procedure invocations, rather than the operations that result from execution of the stored procedures, it enables a high-performance distributed data environment while maintaining transactional integrity of replicated data across the system.

How Replication Server works

Replication Server works to distribute data over a network by:

- Providing application developers and system administrators with a flexible publish-and-subscribe model for marking data and stored procedures to be replicated
- Managing replicated transactions while retaining transaction integrity across the network

A Replication Server at each primary or replicate site coordinates the data replication activities for the local data servers and exchanges data with Replication Servers at other sites.

A Replication Server:

- Receives transactions from primary databases through Replication Agents and distributes them to sites with subscriptions for the data
- Receives transactions from other Replication Servers and applies them to local databases

Replication Server system tables store the information needed to accomplish these tasks. The system tables include descriptions of the replicated data and the following replication objects:

- Replication definitions and subscriptions
- Security records for Replication Server users
- Routing information for other sites
- Access methods for local databases
- Other administrative information

Replication Server system tables are stored in a database called the Replication Server System Database (RSSD).

To manage replication information in Replication Server, use Replication Command Language (RCL). You can execute RCL commands, which resemble SQL commands, on Replication Server using isql, the Sybase interactive SQL utility. For a complete reference for RCL, see the Replication Server *Reference Manual*.

Publish-and-subscribe model

Transactions that occur in a primary database are detected by a Replication Agent and transferred to the local Replication Server, which distributes the information across a network to Replication Servers at destination sites. In turn, these Replication Servers update the replicate database according to the requirements of the remote client.

The primary data is the source of the data that Replication Server replicates in other databases. You publish data at primary sites to which Replication Servers at other (replicate) sites subscribe. To do so, you first create a *replication definition* to designate the location of the primary data. The replication definition describes the structure of the table and names the database that contains the primary copy of the table. For easier management, you can collect replication definitions in *publications.*

Creating a replication definition or publication does not, by itself, cause Replication Server to replicate data. You must also create a *subscription* against the replication definition (or the publication) to instruct Replication Server to replicate the data in another database. A subscription resembles a SQL select statement: It can include a where clause to specify the rows of a table you want to replicate in the local database.

You can have multiple replication definitions for a primary table. Replicate tables can subscribe to different replication definitions to obtain different views of the data.

After you have created subscriptions to replication definitions or publications, Replication Server replicates transactions to databases with subscriptions for the data.

Replicated functions

With some data servers, Replication Server allows you replicate stored procedure invocations asynchronously between databases. By encapsulating many changes in a single replicated function, you can improve performance over normal data replication. Because they are not associated with table replication definitions, replicated functions can execute stored procedures that may or may not modify data directly.

Note Replication Server does not support stored procedure replication on all types of data servers. For more information about replicating stored procedures on a particular data server, refer to the appropriate Replication Agent documentation.

With replicated functions, you can execute a stored procedure in another database. A replicated function allows you to:

- Replicate the execution of a stored procedure to subscribing sites
- Improve performance by replicating only the name and parameters of the stored procedure rather than the actual database changes

Replication Server supports both *applied functions* and *request functions*:

- An *applied function* is replicated from a primary to a replicate database. You create subscriptions at replicate sites for the function replication definition and mark the stored procedure for replication in the primary database.
- A *request function* is replicated from a replicate to a primary database. There is no subscription for a request function. You mark the stored procedure for replication in the replicate database.

Transaction management

Replication Server depends on data servers to provide the transaction processing services needed to protect their stored data. To guarantee the integrity of distributed data, data servers must comply with such transaction-processing conventions as atomicity and consistency.

Data servers that store primary data provide most of the concurrency control needed for the distributed database system. If a transaction fails to update a table with primary data, Replication Server does not distribute the transaction to other sites. When a transaction does update primary data, Replication Server distributes the changes, and unless a failure occurs, the update succeeds at all sites that have subscribed to the data.

Replication Agent

Replication Agent products

Replication Agent products extend the capabilities of Replication Server by supporting non-Sybase data servers as primary data servers in a Sybase replication system.

Sybase offers the following Replication Agent products for non-Sybase databases:

- Replication Agent for DB2 UDB – provides primary data server support for a DB2 Universal Database server running on IBM z/OS platforms.
- Replication Agent – provides primary data server support for DB2 Universal Database, Microsoft SQL Server, and Oracle data servers running on Linux, UNIX, and Microsoft Windows platforms.

Replication Agent for DB2 UDB

Replication Agent for DB2 UDB fits into a replication system as follows:

- The primary data server is DB2 Universal Database, which runs as a subsystem in IBM z/OS. The transaction logs are DB2 logs.

- Replication Agent for DB2 UDB provides a log extract called Sybase Log Extract, which reads DB2 logs and retrieves the relevant DB2 active and archive log entries for the tables marked for replication.
- LTM for MVS receives the data marked for replication from Sybase Log Extract and transfers that data to Replication Server using the TCP/IP communications protocol.

The DB2 data server logs any changes to rows in DB2 tables as they occur. The information written to the transaction log includes copies of the data before and after the changes. In DB2, these records are known as "undo" and "redo" records. Control records are written for commits and aborts. These records are translated to commit and rollback operations.

The DB2 log consists of a series of data sets, which Sybase Log Extract uses to identify DB2 data changes. Because DB2 writes change records to the active log as they occur, Sybase Log Extract can process the log records immediately after they are entered.

Replication Agent

Replication Agent is the component that captures data-changing transactions in DB2 Universal Database, Microsoft SQL Server, or Oracle primary databases on Linux, UNIX, and Microsoft Windows platforms by reading the database transaction log.

Replication Agent is implemented in the Java programming language. During Replication Agent installation a Java Runtime Environment (JRE) is installed on the computer that will act as the Replication Agent host machine.

Replication Agent uses the Java Database Connectivity (JDBC) protocol for all of its communications. It uses a single instance of the Sybase JDBC driver (jConnect™ for JDBC™) to manage all of its connections to Open Client and Open Server applications, including the primary Replication Server. In the case of the primary data server, Replication Agent connects to the primary database using the appropriate JDBC driver for that database.

How Replication Agent works

A Replication Agent is a Replication Server client that retrieves information from the transaction log for a primary database and formats it for the primary Replication Server.

After the desired primary tables and stored procedures are marked in the Replication Agent, it detects changes to primary data and, using Log Transfer Language (LTL), which is a subset of Replication Control Language (RCL), sends changes in primary data to the primary Replication Server.

A Replication Agent performs the following steps:

1 Logs in to the Replication Server.

2 Sends a connect source command to identify the session as a log transfer source and to specify the database for which transaction information will be transferred.

3 Retrieves the name of the Maintenance User for the database from the Replication Server.

4 Requests the secondary truncation point for the database from the Replication Server.

   This request returns a value, called the *origin queue ID*, that the Replication Agent uses to find the location in the transaction log, where it will begin transferring transaction operations. The Replication Server has already received operations up to this location.

5 Retrieves records from the transaction log, beginning at the record following the secondary truncation point, and formats the information into LTL commands.

ECDA database gateways

The Enterprise Connect Data Access (ECDA) products are Open Server-based software gateways that support DB-Library and CT-Library application program interfaces, and Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) protocols. They serve as fundamental building blocks for database middleware applications that allow you to access mainframe and LAN-based non-Sybase data sources.

ECDA Options

There are two options available for ECDA:

- ECDA Option for ODBC
- ECDA Option for Oracle

ECDA Option for ODBC

ECDA Option for ODBC provides Replication Server with an Open client interface to IBM DB2 UDB, Microsoft SQL Server, and ODBC-accessible databases.

Note The ODBC driver for ECDA Option for ODBC (the back-end driver connecting to the target) is not provided by Sybase; you must obtain, install, and configure it.

ECDA Option for ODBC provides access to non-Sybase data sources, using the ODBC back-end (server-side) driver that you obtain for your target database, such as IBM DB2 or Microsoft SQL Server. Following the vendor's instructions, install the ODBC driver on the same server as ECDA Option for ODBC, then configure ECDA Option for ODBC to use that ODBC driver for access to your database.

Note Be sure to verify that your ODBC driver is compatible with Sybase driver manager software or that it contains a driver manager.

Because ODBC drivers have varying degrees of functionality, it is important that when working with non-Sybase-provided, third-party ODBC drivers, you carefully integrate and test them to be sure they meet your needs.

ECDA Option for Oracle

ECDA Option for Oracle provides Replication Server with an Open Client interface to Oracle databases. To Replication Server, ECDA Option for Oracle appears as an Open Server application that understands Oracle SQL.

ECDA products provide the following middleware services:

- Access services that provide access to non-Sybase data sources
- Administrative services (through DirectConnect Manager) that provide server-side system management

How ECDA works

All Sybase ECDA options provide basic connectivity to non-Sybase data services. In particular, they provide access management, copy management, and remote systems management.

Each ECDA option consists of a DirectConnect server and one or more access service libraries. The server provides the framework in which the service libraries operate. From the server, each access service library accesses data from a particular target database, such as DB2 Universal Database, Microsoft SQL Server, or Oracle.

Each access service library contains one or more access services that are specific sets of configuration properties. An access service transfers data between Replication Server and the target databases.

The DirectConnect server listens for, validates, and accepts incoming client connections, such as language events or remote procedure calls (RPCs). These events are routed to the target data source (replicate database) through access services, which provide target-specific connectivity features, including datatype conversion, network connectivity, and SQL transformation.

CHAPTER 3

Heterogeneous Replication Issues

This chapter describes the issues and problems that are specific to heterogeneous replication and that must be addressed in a successful heterogeneous replication system.

Overview of heterogeneous replication issues

The biggest challenge in implementing a successful heterogeneous replication system is accommodating the unique characteristics of data servers that are supplied by different vendors. Regardless of the type or brand of a data server, there are issues that are specific to the data server's role in the replication system. And when a single data server acts as both a primary data server and a replicate data server (bidirectional replication), there are still more issues to consider.

Primary database issues

The following primary database issues must be addressed in a successful heterogeneous replication system:

- The requirements of the Replication Agent and the intrusions and impacts of the Replication Agent on the data server. For example, some Replication Agents create and use database objects in the primary database to support replication.

- The access and permissions required in the data server for other replication system components. Both the primary Replication Server and the Replication Agent for a database must have user IDs and passwords defined in the database with appropriate permissions to access primary database objects.
- The connectivity required to support communications between the data server and other replication system components. Replication Agents use either the native communication protocol of the data server, ODBC protocols, or JDBC protocols to communicate with the primary database. Replication Server may require a database gateway to communicate with a data server.
- The specific limitations on replication from the particular data server. For example, some Replication Agents restrict the configuration options of some data servers. Replication Server may impose size limitations on some native datatypes in some databases.
- How replication definitions stored in the RSSD are used by the Replication Agent for the particular data server. For example, both Replication Server and Replication Agents are case sensitive in identifying database object names, but some databases are not.
- The datatype conversions that may be required when replicating transactions from one particular data server to another type of data server. For example, almost every type of data server has a unique way of representing temporal data. The TIMESTAMP datatype in one database may need to be "translated" to be stored as a datetime datatype in another database.
- The replication system management issues specific to the particular data server. For example, different data servers allow different system management options.

For more information about specific primary database issues for specific databases, see "Primary data servers" on page 37.

Replicate database issues

The following replicate database issues must be addressed in a successful heterogeneous replication system:

- The requirements of the ECDA database gateway for the particular database server. The DirectConnect access services must be configured to work with the replicate database server and Replication Server.

- The access and permissions required in the data server for the replication system to apply transactions to the replicate database. Both the replicate Replication Server and the ECDA gateway for a database must have user IDs and passwords defined in the database, with appropriate permissions to access replicate database objects.
- The connectivity required to support communication between the replicate data server and other replication system components. ECDA gateways use either the native communication protocol of a data server, or standard ODBC or JDBC protocols to communicate with a replicate database. Replication Server generally requires a database gateway to communicate with a non-Sybase data server.

  Note In the case of DB2 Universal Database on IBM z/OS, Replication Server can use Mainframe Connect DirectConnect for z/OS Option to connect directly to the mainframe in a "gatewayless" system. This eliminates the need for a database gateway. Replication through a gatewayless connection requires a TCP/IP connection to the mainframe. For more information about DB2 for IBM z/OS gatewayless connections, see "IBM DB2 Universal Database replicate data servers" on page 79, and the Mainframe Connect Server Option for IBM IMS and MVS *Installation and Administration Guide*.

- The limitations on replication into the particular data server. For example, Replication Server imposes limitations on some native datatypes in some databases.
- The intrusion and impact of the database objects required to support Replication Server operations. Replication Server requires two tables and may require some stored procedures to manage a replicate database.
- The replication system management issues specific to the particular data server. For example, different data servers allow different system management options.

For more information about specific replicate database issues for specific databases, see "Replicate data servers" on page 78.

Setting character sets

In a heterogeneous replication system, in which the primary and replicate data servers are different types, the data servers might not support the same character sets. In that case, replication system components must perform at least one character set conversion (from the primary data server's character set to the replicate data server's character set).

Even in a homogeneous replication system, in which both primary and replicate data servers are the same type, character set conversions might be required if replication system components reside on more than one type of platform.

Character set problems can produce data inconsistencies between the primary database and the replicate database. To avoid character set problems, you must either:

- Use the same character set on all servers and platforms in the replication system, or
- Use compatible character sets on all servers and platforms in the replication system, and configure replication system components to perform the appropriate character set conversions.

By default, the Java Virtual Machine (JVM) under which a Replication Agent instance runs, finds your system's default character set. The type of character data that Replication Agent can handle is determined by the character set, also known as the encoding. Unless you want to override the default character set that the JVM finds on your system, you do *not* have to explicitly set the character set-related environment variable.

To support overriding the default character set, all of the executable scripts (or batch files) in the Replication Agent */bin* directory and the Replication Agent instance *RUN* scripts refer to an environment variable named RA_JAVA_DFLT_CHARSET. You can set this environment variable to use the character set you want. However, the character set you specify must be the character set configured on the primary database. For a list of valid Java character sets, see Supported Encodings on the Internationalization page under Documentation for the J2SE 5.0 JDK at http://java.sun.com/javase/technologies/core/basic/intl/.

Note If you are using Replication Server to replicate a number of different character sets, you must configure it for UTF8.

You can override the system default character set by either:

- Setting the value of a system variable named RA_JAVA_DFLT_CHARSET in your environment and using the ra utility to start the Replication Agent instance, or
- Setting the value of the RA_JAVA_DFLT_CHARSET variable in the Replication Agent instance RUN script and using the RUN script to start the Replication Agent instance.

If you start a Replication Agent instance by invoking the ra utility, you can override the value of the RA_JAVA_DFLT_CHARSET system variable in your environment to specify the character set. All instances you start with the *ra* utility will use the same character set.

If you start a Replication Agent instance by invoking the instance *RUN* script (or batch file), you can edit the instance *RUN* script to specify the default value of RA_JAVA_DFLT_CHARSET and specify the character set you want to use. You can configure each instance with a different character set.

For more information on setting and overriding the default character set, see Chapter 2, "Setup and Configuration" in the Replication Agent *Administration Guide*.

Heterogeneous replication limitations

This section describes some limitations of a heterogeneous replication system based on Sybase replication technology:

- Stored procedure replication
- Owner-qualified object names
- Large object replication
- Setup for replicate databases
- Replication Server support for encrypted columns
- Subscription materialization
- Replication Server rs_dump command
- Replication Server rs_marker command
- Replication Server rs_dumptran command
- Replication Server rs_subcmp utility

Stored procedure replication

Stored procedure replication allows the execution call of a stored procedure to be replicated, including the parameter values passed as arguments to the primary stored procedure call.

The availability of stored procedure replication depends on the capabilities of the primary and replicate databases, as well as support from the associated Replication Agent and ECDA database gateway. Refer to the documentation for the specific Replication Agent and ECDA components to determine if stored procedure replication is available for your databases.

Owner-qualified object names

Access to replicate tables and stored procedures in a non-Sybase database often requires that the reference to the replicate table or stored procedure be owner-qualified.

For example, suppose the Replication Server Maintenance User defined to apply transactions to an Oracle replicate database is orauser. A replicate insert command to table table1 may fail with a "table not found" error if the owner of table1 is bob. When attempting to find table1, Oracle looks for orauser.table1, not bob.table1. To properly identify the replicate table to be updated, you can:

- Create an alias at the Oracle replicate database that refers to the correct replicate table. For example, you could create a synonym object in Oracle named table1, which refers to the fully qualified name of "bob.table1."
- When creating the replication definition, use the with replicate table named [table_owner.['table_name']] clause. To satisfy the example, the clause would be:

```
with replicate table named bob.table1
```

Owner qualifying with multiple replicate databases

The problem becomes a little more complicated when table1 is to be replicated to more than one replicate database (for example, Oracle replicate table bob.table1). The option of using the with replicate table named clause in the replication definition supports only one replicate table name.

To work around this issue, you must create multiple replication definitions, one for each unique replicate table name required. Then, each subscription must refer to the correct replication definition. Also, each replication definition must use the with replicate table named clause.

Large object replication

Large object (LOB) datatypes (such as BLOB, CLOB, IMAGE, and TEXT) provide support for the longest streams of character and binary data in a single column. Their size poses unique challenges, both as primary and replicate data.

Primary database LOB replication issues

At the primary database, the impact of LOB datatypes is on the transaction logging function. For Replication Agents, the log resources must be adequate to support retention of the changes in LOB data, only after images of LOB data are logged.

Replicate database LOB replication issues

Adaptive Server Enterprise uses a text pointer to identify the location of text and image column data. The text pointer is passed to system functions that perform the actual updates to data in these large columns. The same technique is used internally in Replication Server to apply LOB datatypes. Replication Server obtains a text pointer, and data server function calls are made to apply the data to replicate databases.

Replication Server default function strings are designed for an Adaptive Server replicate database. Replication Server executes an rs_textptr_init or rs_get_textptr function string, followed by one or more rs_writetext function strings to apply the LOB data to a replicate database. The default function strings supplied in Replication Server for most non-Sybase data servers do not support LOB replication.

When a non-Sybase database is the replicate database, the database gateway used to communicate with the replicate database must be able to emulate the Adaptive Server text pointer processing. The ECDA Option for ODBC and the Mainframe Connect DirectConnect for z/OS Option gateways provide this feature.

ECDA Option for ODBC

ECDA option for ODBC provides support for LOB replication into Microsoft SQL Server databases. For more information, see “Microsoft SQL Server replicate data servers” on page 91.

In the case of a replicate database in DB2 for IBM z/OS, you can use the Mainframe Connect DirectConnect for z/OS option to provide a "gatewayless" connection to the replicate database, and use modified rs_get_textptr and rs_writetext function strings to support LOB replication into DB2. For more information, see "IBM DB2 Universal Database replicate data servers" on page 79.

You might be able to remove the dependency on text pointers from the Replication Server (or the ECDA database gateway) by modifying the Replication Server text pointer function strings and creating a stored procedure in the replicate database.

Remote procedure call (RPC) method

Each text (LOB) column to be processed by Replication Server has a unique rs_writetext function string created to issue a writetext function call to the replicate database. You can modify the rs_writetext function string to replace the default writetext function call with a remote procedure call (RPC) to a stored procedure you create in the replicate database.

Replicating LOB datatypes to DB2 Universal Database on Linux, UNIX, and Windows platforms requires the RPC method, with a separate stored procedure in the replicate database for each LOB column.

Replicating LOB datatypes to DB2 for IBM z/OS requires the writetext method, in which the Mainframe Connect DB2 UDB Option for CICS and IMS language handler facilitates replication of LOB datatypes to DB2. For more information about the rs_writetext function string, see the Replication Server *Reference Manual*.

Implement an RPC workaround

To implement an RPC workaround, you must create one or more stored procedures in the replicate database (depending on the types of primary keys processed), and you must create one Replication Server function string for each text or image (LOB) column to be replicated.

The stored procedure must perform the following processing:

- The stored procedure must be able to identify the row where the text data should be applied. One way to accomplish this is to have the primary key values for the table passed as parameters in the RPC.
- The stored procedure must be able to append text data to the data in the column. Replication Server executes the RPC repeatedly, passing 16,384 bytes (16K) of data at each invocation until all the data is sent.

- The stored procedure must handle both insert and update operations. During insert operations, the Replication Server inserts all other (non-LOB) column data into a row, leaving the LOB columns null. Then, Replication Server invokes the RPC to insert LOB data. During update operations, the Replication Server updates all other (non-LOB) column data, but it does not update the LOB column—any existing data remains. The stored procedure must be able to replace the existing LOB data on an update operation, and also to append data from the multiple RPC invocations.

The ECDA gateway for the replicate database must support RPC processing. For more information about RPC handling, refer to the appropriate ECDA documentation.

For more information about Replication Server processing of text and image columns and the rs_writetext system function, see the Replication Server *Administration Guide* and the Replication Server *Reference Manual*.

Setup for replicate databases

Replication Server provides a utility named rs_init, which sets up an Adaptive Server database as a replicate database as follows:

- Creates the Replication Server database connection
- Creates the required tables and stored procedures in the replicate database
- Defines the Replication Server Maintenance User ID

The rs_init utility is not provided for non-Sybase databases. Instead, Sybase provides an example script for each step in setting up a non-Sybase database as a replicate database. These sample scripts are designed for each supported database. You must modify them and apply them manually.

For more information on setting up replicate databases, see “Replicate data servers” on page 78.

Replication Server support for encrypted columns

To enable replication of encrypted columns to an Adaptive Server table, Replication Server provides an rs_set_ciphertext function string that you must alter each time that you make a database connection to a non-ASE database. The following describes the function string and its usage:

Function string	rs_set_ciphertext
Description	Enables replication of encrypted columns to an Adaptive Server table.
Example	Alters rs_set_ciphertext for non-ASE databases which does not support “set ciphertext on.”

```
alter function string rs_set_ciphertext
for some_function_string_class
output language
''
```

Usage

- rs_set_ciphertext is called after rs_usedb for any user database connection.Replication Server will not call this function string for Replication Server connections and RSSD connections.
- rs_set_ciphertext issues “set ciphertext on” for the rs_default_function_class and the rs_sqlserver_function_class. For all other function classes, rs_set_ciphertext is set to null (an empty string).
- Encrypted columns come to Replication Server in varbinary, encrypted form. For materialization and de-materialization, Replication Server needs either to “set ciphertext on” for the database connection, or to call the ASE ciphertext function.
- Replication Server always sets the ciphertext property on, regardless of whether there is an encrypted column to be replicated or the target database accepts ciphertext property.
- Replicated encrypted columns are not searchable nor translatable. Replication Server will not decrypt data and users cannot replicate encrypted columns based on their contents
- Replication Server does not support encrypting text, unitext, and image columns.
- Do not specify encrypted columns as searchable columns. Replication Server does not know if a varbinary column is ciphertext or plain binary and will not prevent an encrypted column being a search column.
- Do not map encrypted columns to other than varbinary datatypes. Replication Server does not know if a column is encrypted or not and will not prevent ciphertext from being converted to other datatypes.
- In case of failure, Replication Server will continue running and will not report back to the user. This is for backward compatibility with older versions of ASE, which do not support “set ciphertext on.”

For more information about the rs_set_ciphertext command, its use, and function-string modifications, see Chapter 4, Replication Server System Functions, in the Replication Server *Reference Manual*.

Subscription materialization

Materialization is the process of creating and activating subscriptions, and copying data from the primary database to the replicate database, thereby initializing the replicate database.

Before you can replicate data from a primary database, you must set up and populate each replicate database so that it is in a state consistent with that of the primary database. There are two types of subscription materialization supported by Replication Server:

- Bulk materialization – manually creating and activating a subscription and populating a replicate database using data unload and load utilities outside the control of the replication system.
- Automatic materialization – creating a subscription and populating a replicate database using Replication Server commands.

Because of the limitations of reading a transaction log (a complete picture of all table data is not available), Replication Agents do not support automatic materialization.

Bulk materialization methods are supported, with varying complexity based on the specific Replication Agent capabilities.

Refer to the Replication Server *Administration Guide* for a general discussion of subscription materialization, and refer to the appropriate Replication Agent documentation for details regarding a particular Replication Agent and its materialization support.

Replication Server rs_dump command

The Replication Server rs_dump command is typically used to coordinate database dump activities across a replication system. When a replicate connection receives an rs_dump transaction, Replication Server executes the rs_dump function string for that connection. You can customize the rs_dump function string to execute whatever commands are required.

For heterogeneous replication, some Replication Agents provide a method to invoke the rs_dump command from a non-Sybase primary database. Refer to the appropriate Replication Agent documentation to determine if rs_dump execution from the primary database is supported.

For replicate databases, no default function string for rs_dump is provided.

For more information about the rs_dump command, its use, and function-string modifications, see the Replication Server *Reference Manual*.

Replication Server rs_marker command

The Replication Server rs_marker command is a primary database transaction log marker mechanism that is used to assist with the materialization process. An rs_marker execution is used to pass activate subscription and validate subscription commands to a primary Replication Server. Most Replication Agents support an rs_marker invocation to assist with materialization.

For more information about rs_marker usage, see the Replication Server *Reference Manual*. For more information about the use and availability of rs_marker for a particular database, refer to the appropriate Replication Agent documentation.

Replication Server rs_dumptran command

The Replication Server rs_dumptran command is typically used to coordinate database transaction dump activities across a replication system. When a replicate connection receives an rs_dumptran transaction, the Replication Server executes the rs_dumptran function string for that connection. You can customize the rs_dumptran function string to execute whatever commands are required.

Support for rs_dumptran is not provided for non-Sybase primary databases.

For replicate databases, no default function string for rs_dumptran is provided.

For more information about the rs_dumptran command, its use, and function-string modifications, see the Replication Server *Reference Manual*.

Replication Server rs_subcmp utility

Replication Server provides an rs_subcmp executable program that you can use to compare primary and replicate tables, optionally reconciling any differences found.

For non-Sybase database support, you may use rs_subcmp providing you have connectivity to the primary and replicate databases. You must also develop custom SELECT commands for the primary and replicate databases that will provide comparable outputs for both. Additional options are to buy third-party tools that provide such functionality, or build your own application.

For more information about comparing and reconciling databases in a heterogeneous replication system, see Appendix C, “Heterogeneous Database Reconciliation.”

PART 2

Component-Specific Issues

Chapters in this part describe component-specific issues and considerations peculiar to a replication system with heterogeneous or non-Sybase data servers.

- Chapter 4, “Data Server Issues,” describes the issues and problems that are peculiar to specific data servers in a heterogeneous replication system.
- Chapter 5, “Replication Server Issues,” describes the issues specific to Sybase Replication Server in a heterogeneous replication system.

CHAPTER 4

Data Server Issues

This chapter describes the issues specific to non-Sybase primary and replicate data servers in a Sybase replication system.

Primary data servers

In addition to Adaptive Server Enterprise, Sybase replication technology actively supports transaction replication from the following relational database servers:

- IBM DB2 Universal Database
- Microsoft SQL Server
- Oracle

To find out about the most current, supported versions of these data servers, refer to the documentation for the Replication Agent that supports a particular non-Sybase data server.

General issues for non-Sybase primary data servers

There are several issues related to non-Sybase primary data servers in a Sybase replication system. A successful replication system must address all of the issues for each primary data server.

Non-Sybase primary data server issues include:

- The requirements of the Replication Agent, including any limitations, intrusions, and impacts on the data server's operation

- The access and permissions necessary for the Replication Agent to obtain transactions from the primary database
- The connectivity requirements to support communications between the primary data server and other replication system components
- The limitations imposed on the replication system by the non-Sybase data server
- The datatype conversions (or translations) that may be required to replicate transactions from one type of data server to another
- The replication system management issues specific to the non-Sybase data server

The following sections describe the specific primary data server issues for each actively supported type of non-Sybase data server.

IBM DB2 Universal Database primary data servers

Sybase provides two different Replication Agent software products to support primary databases in the IBM DB2 Universal Database:

- **Replication Agent for DB2 UDB** – supports replication from DB2 Universal Database on IBM z/OS (mainframe) platforms.
- **Replication Agent for UDB** – supports replication from the IBM DB2 Universal Database on UNIX, Linux, and Microsoft Windows platforms.

In the following sections, the IBM DB2 Universal Database is treated separately for the two platform types:

- IBM DB2 Universal Database primary data servers on IBM z/OS
- IBM DB2 Universal Database primary data servers on UNIX, Linux, or Windows

IBM DB2 Universal Database primary data servers on IBM z/OS

This section describes the primary database issues and considerations specific to the IBM DB2 Universal Database server on a IBM z/OS platform in a Sybase replication system.

Replication Agent for DB2 UDB

As a primary data server, the IBM DB2 Universal Database interacts with the Replication Agent for DB2 UDB.

The Replication Agent identifies and transfers information about data-changing operations or transactions from an IBM DB2 Universal Database primary database to a primary Replication Server.

The Replication Agent interacts with the primary Replication Server and with the RSSD of the primary Replication Server, if so configured.

DB2 for z/OS system management issues

Replication Manager (RM) does not manage or monitor the Replication Agent for DB2 UDB.

Replication intrusions and impacts in the IBM DB2 Universal Database for z/OS

The performance and operation of IBM DB2 Universal Database primary data servers in a Sybase replication system might be affected as follows:

- The IBM DB2 Universal Database transaction log is affected in the following ways:
  - Replication requires a *before* and *after* image of each row that is changed. When you mark a primary table for replication, the table is altered with the DATA CAPTURE CHANGES clause. As the number of tables marked for replication increases, so does the DASD space requirement for the IBM DB2 Universal Database active log data sets.
  - Using Replication Agent for DB2 UDB increases the amount of data stored in IBM DB2 Universal Database logs. The size of the increase depends on the number, type, and size of the primary tables, and the types of transactions replicated. For example, update transactions require both *before* and *after* images, and they include all of the columns in a row, even if those columns do not change. For more detailed information, see the Replication Agent for DB2 UDB documentation.
- Two Replication Agent system tables are created in the primary IBM DB2 Universal Database when the Replication Agent is installed:
  - LTMOBJECTS contains a row for each primary table marked for replication. Its size depends on the number of tables marked for replication.

- LTMLASTCOMMIT stores information about the most recent replicated transaction successfully committed in the replicate database. The size changes according to the number of primaries that are replicated into the IBM DB2 Universal Database, when it is a replicate database.

Note Unless the primary IBM DB2 Universal Database also serves as a replicate database (in bidirectional replication), you can remove the LTMLASTCOMMIT table from the primary database.

- Each Replication Agent for DB2 UDB started task can process only one IBM DB2 Universal Database log. Replicating transactions from multiple IBM DB2 Universal Database subsystems (or from multiple data-sharing members in a data-sharing environment) requires one Replication Agent-started task for each IBM DB2 Universal Database subsystem (or data-sharing member).
- The following IBM DB2 Universal Database utilities may jeopardize replication integrity if they are executed:
  - LOAD LOG NO
  - RECOVER
  - REORG with RECOVER

Note The IBM DB2 Universal Database LOAD LOG YES utility is supported in Replication Agent for DB2 UDB.

DB2 UDB primary database permissions

Two user IDs must be created in the IBM DB2 Universal Database to facilitate its operation as a primary data server:

- LTMADMIN user – a TSO user, optionally named LTMADMIN, to:
  - Install, start, and stop the Replication Agent for DB2 UDB
  - Manage the Replication Agent system tables on the IBM DB2 Universal Database

  The LTMADMIN user must have ALTER TABLE authority on any IBM DB2 Universal Database table to be marked for replication. This user ID issues an ALTER TABLE DATA CAPTURE CHANGES command on a primary table that is marked for replication.

The LTMADMIN user must have READ permission on the IBM DB2 Universal Database log files.

- Replication Server Maintenance User – the user ID specified in the Replication Server create connection command for the primary database.

  Any updates applied to the primary database by the Maintenance User are ignored for replication, unless the value of the LTM for MVS LTM_process_maint_uid_trans configuration parameter is Y.

Note LTM for MVS is a component of Replication Agent for DB2 UDB and the other component is the IBM DB2 Universal Database log reader.

Because the Maintenance User ID may be a user ID that can perform updates on the IBM DB2 Universal Database, it must conform to IBM z/OS user ID restrictions.

Primary data server connectivity

Replication Agent for DB2 UDB requires the following to connect to a primary IBM DB2 Universal Database data server in an IBM z/OS environment:

- A valid user ID (the LTADMIN user identified earlier) must be defined to IBM z/OS and granted permission to the correct IBM DB2 Universal Database packages. Replication Agent for DB2 UDB uses this user ID to log in to the IBM DB2 Universal Database. This user ID must have the following permissions:
  - CREATE authority for the Replication Agent for DB2 UDB system tables
  - ALTER TABLE authority on all tables marked for replication
  - READ authority on the IBM DB2 Universal Database log data sets
- Replication Agent for DB2 UDB jobs must have their JCL modified to execute with the correct accounting, user_id, IBM DB2 Universal Database logs, and IBM DB2 Universal Database subsystem libraries identified.

Replication Server connectivity

Replication Agent for DB2 UDB requires the following to connect to the primary Replication Server:

- To support a TCP/IP connection between the Replication Agent and the primary Replication Server, you must provide the IBM z/OS system with information about the Sybase servers and TCP/IP host addresses in your replication system. You do this by editing the SYGWHOST macro, located in the XCPHPING member in the *hlq*.JCL data set.

  The SYGWHOST macro does the following:

  - Serves as a directory containing network addresses and other information that controls how the Replication Agent connects to other servers
  - Configures the Open Client component of LTM for MVS to use the MVS TCP/IP protocol for a connection between the Replication Agent and the primary Replication Server

  Note You must create a SYGWHOST macro entry for each primary Replication Server that receives transactions from an IBM DB2 Universal Database subsystem in the IBM z/OS system, and also for any RSSDs that Replication Agents for DB2 UDB are configured to use. You can define any number of Replication Servers in the SYGWHOST macro.

- The values of the following LTM for MVS configuration parameters must be set as described so that the Replication Agent for DB2 UDB can connect to the primary Replication Server:
  - Communications_Protocol – must be set to IBMTCP for connectivity to the Replication Server.
  - RS – the name of the host machine on which the primary Replication Server resides.
  - RS_user – the user ID that the Replication Agent uses to log in to the primary Replication Server. This user ID must be defined and granted connect source permission in the Replication Server.

    Note The RS_user should not be identical to the Maintenance User ID specified in the Replication Server create connection command for the primary database.

  - RS_pw – the password for the user ID that the Replication Agent uses to log in to the primary Replication Server.

- RS_source_db – the logical database name of the primary database specified in the Replication Server database connection. Sybase recommends using the IBM DB2 Universal Database subsystem or group name.
- RS_source_ds – the MVS system that contains the IBM DB2 Universal Database primary database. This value can be arbitrary, as long as it matches the *data_server* value specified in the Replication Server create connection command for the primary database.

RSSD connectivity

Replication Agent for DB2 UDB requires the following to connect to the Replication Server System Database (RSSD) of the primary Replication Server:

- If you set the value of the LTM for MVS Use_repdef configuration parameter to Y, you must provide values in the SYGWHOST macro for the data server that contains the RSSD for the primary Replication Server.
- The values of the following LTM for MVS configuration parameters must be set as described so the Replication Agent can connect to the RSSD of the primary Replication Server:
  - RSSD_server – the name of the host machine for the data server that contains the RSSD.
  - RSSD_database – the database name of the RSSD.
  - RSSD_user – the user ID that the Replication Agent uses to log in to the RSSD. This user ID must be defined and granted select permission in the RSSD.
  - RSSD_pw – the password for the user ID that the Replication Agent uses to log in to the RSSD.

DB2 UDB primary database configuration issues

The Replication Agent for DB2 UDB is a mainframe z/OS application. It consist of two tasks that run simultaneously in a single z/OS address space:

- **Sybase Log Extract** – continuously scans the IBM DB2 Universal Database active and archive logs for data-changing operations on primary tables.

- **LTM for MVS** – receives replicated transactions from Sybase Log Extract, converts them to Log Transfer Language (LTL), and then sends them to the primary Replication Server.

One Replication Agent for DB2 UDB started task is required for each IBM DB2 Universal Database log from which transactions will be replicated.

All Replication Agent installation and configuration issues are described in the Replication Agent for DB2 UDB *Installation Guide*. However, the following items may need further attention in a heterogeneous replication system:

- All configuration parameter values in the LTM for MVS configuration file are case sensitive. Be careful when specifying the values of the RS_source_ds and RS_source_db parameters, as the Replication Server is also case sensitive. If the same case is not used in both Replication Agent and Replication Server parameters, the connection does not succeed.
- The LTM for MVS LTM_process_maint_uid_trans configuration parameter controls whether the Replication Agent sends transactions performed by the Maintenance User to the primary Replication Server. The Maintenance User ID is defined in the Replication Server create connection command for the primary database.

  In a bidirectional replication environment (replicating both into and out of the same IBM DB2 Universal Database region), the value of the LTM_process_maint_uid_trans parameter should be set to N. If it is not, transactions replicated to another site could return to be applied at the originating site, creating an endless loop.

Replication definitions for primary tables in DB2 for z/OS

The LTM for MVS Use_repdef configuration parameter controls whether the Replication Agent sends Log Transfer Language (LTL) that contains only the columns specified in a replication definition, or all of the columns in the IBM DB2 Universal Database primary table.

When the value of the Use_repdef parameter is set to N, the Replication Agent sends LTL with data for all of the columns in the IBM DB2 Universal Database primary table. When the value of the Use_repdef parameter is set to Y, the Replication Agent sends LTL with data for only the columns specified in the replication definition.

By sending data for only the columns needed for the replication definition, network traffic is reduced, which can improve performance.

If you set the value of Use_repdef to Y, you can use other LTM for MVS parameters, such as Minimal_cols and Suppress_col_names, to enhance Replication Agent performance. See the Replication Agent for DB2 UDB *Installation Guide* for more information.

DB2 for z/OS primary datatype translation issues

The LTM for MVS Date_in_char configuration parameter controls whether the Replication Agent sends values in IBM DB2 Universal Database DATE columns as default CHAR(10) character strings, or converts them to the Sybase datetime format.

In general, if the value of the Date_in_char parameter is set to N, all of the corresponding datatypes in replication definitions for DATE columns must specify the Sybase datetime datatype.

See the Replication Agent for DB2 UDB *User's and Troubleshooting Guide* for a complete description of the Date_in_char parameter. See the Replication Server *Administration Guide* for a complete description of UDDs and their use.

Note If you use any date- or time-related UDDs in a replication definition, Sybase recommends that you configure the Replication Agent to send data to the Replication Server in the format that is native to the primary database. Sybase prefers to *not* have the Replication Agent perform any datatype translations.

In general, the Replication Agent for DB2 UDB should not perform datatype translations. However, when all of the replicate data servers require the same translation, to save processing time, it is probably better to perform the translation once at the Replication Agent, rather than at each replicate database DSI.

IBM DB2 Universal Database primary data servers on UNIX, Linux, or Windows

This section describes the primary database issues and considerations specific to the IBM DB2 Universal Database server on a UNIX, Linux, or Windows platform in a Sybase replication system.

Replication Agent for UDB

As a primary data server, IBM DB2 Universal Database interacts with Replication Agent. An instance of the Replication Agent configured for the IBM DB2 Universal Database is referred to as a *Replication Agent for UDB*.

The Replication Agent for UDB identifies and transfers information about data-changing operations or transactions from an IBM DB2 Universal Database primary data server to a primary Replication Server.

Note A separate Replication Agent for UDB instance is required for each database from which transactions are replicated.

The Replication Agent interacts with the primary Replication Server and with the RSSD of the primary Replication Server, if so configured.

Note Replication Agent is a Java program. Some operating systems may require patches to support Java. Refer to the Replication Agent *Administration Guide* and the Replication Agent *Release Bulletin* for more information.

Replication intrusions and impacts on the DB2 Universal Database

The performance and operation of the IBM DB2 Universal Database primary data servers in a Sybase replication system might be affected by the transaction log in the following ways:

- You must set LOGARCHMETH1 configuration parameter to LOGRETAIN or DISK:<*path*>, where <*path*> is the directory to which the logs are archived (by using the pdb_xlog command). To determine the LOGARCHMETH1 setting, use the following UDB command:

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

- Replication requires a *before* and *after* image of each row that is changed. When you mark a primary table for replication, the Replication Agent for UDB sets the table's DATA CAPTURE option to DATA CAPTURE CHANGES. As the number of tables marked for replication increases, so does the space requirement for the IBM DB2 Universal Database transaction log.
- The primary database must have a user temporary system managed tablespace with a page size of at least 8KB.

Primary database limitations in the DB2 Universal Database

Replication Agent does not support stored procedure or DDL replication for IBM DB2 Universal Database. For more information, see the Replication Agent *Primary Database Guide*.

Replication Manager limitations

The Replication Manager plug-in cannot start, but can stop a Replication Agent instance in a primary IBM DB2 Universal Database data server.

See the Replication Agent *Administration Guide* for more information on starting and stopping a Replication Agent instance.

Scripts to set up single-table replication from DB2 Universal Database to ASE

The Replication Agent provides a set of sample scripts that you can use to set up simple, single-table replication from the IBM DB2 Universal Database to Adaptive Server. These scripts are located in the *$SYBASE/RAX-15_1/scripts* directory of the Replication Agent installation.

For more information about the sample scripts and their use, see the Replication Agent *Administration Guide*.

DB2 Universal Database system management issues

The Replication Agent provides a number of commands that return metadata information about the primary database (database names, table names, procedure names, column names, and so on). It does this by issuing specific JDBC calls designed to return this information or by querying the system tables directly.

DB2 Universal Database primary database permissions

The Replication Agent for UDB requires an IBM DB2 Universal Database login that has permission to access data and create new objects in the primary database.

The IBM DB2 Universal Database login must SYSADM or DBADM authority to access the primary database transaction log.

Replication Agent objects

This section describes objects that Replication Agent creates in the primary database to support replication. Replication Agent creates these objects when you initialize it using pdb_xlog init.

Replication Agent object names

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

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

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

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

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

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

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

Table objects

Table 4-1 lists the Replication Agent tables. No permissions are granted on these tables when they are created. All of these tables contain at least one index, and some contain more than one index.

Table 4-1: Replication Agent tables

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

Java procedure objects

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

- On Windows, the files are installed in *%DB2DIR%/SQLLIB/FUNCTION/jar/pds_username*. Here, *$DB2DIR* is the path to the UDB installation, and *pds_username* is the name of the primary database user specified by the pds_username Replication Agent configuration parameter.
- On UNIX, the files are installed in *$HOME/sqllib/function/jar/pds_username*. Here, *$HOME* is the home directory of the UDB instance owner and the pds_username is the name of the primary database user specified by the pds_username Replication Agent configuration parameter.

These jar files implement several Java procedures in the UDB primary database. Table 4-2 lists the Java procedures that are created during the Replication Agent initialization and used in log truncation.

Table 4-2: Java procedures for truncation

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

Getting actual names of the Replication objects

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

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

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

  ```
  pdb_xlog
  ```

  The pdb_xlog command returns a list of objects created by the Replication Agent in the primary database.

Primary data server connectivity

Replication Agent for UDB requires the following to connect to a primary IBM DB2 Universal Database data server:

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

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

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

  On a UNIX system, instead of using ODBC, simply catalog the node and the primary database in UDB. Then, use the database alias specified when cataloging the primary database to set the data source Replication agent configuration parameter.

- If the Replication Agent for UDB is installed on a UNIX or Linux machine, you must source the UDB *db2cshrc* (for C-shell) or the *db2profile* (for Bourne and Korn shells) script before starting the Replication Agent. These scripts are located in one of the following paths:

  - *$DB2DIR/cfg*

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

  - *$HOME/sqllib*

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

- For all Replication Agent platforms except HP Itanium, the library path environment variable must point to the location of the 32-bit UDB libraries, and not to the 64-bit libraries. The 32-bit versions of the libraries are located in the *<library_path>/sqllib/lib32* directory.

  The exact name of the library path environment variable depends on the operating system.

- Table 4-3 contains a description of the Replication Agent for DB2 UDB configuration parameters that must be set.

Table 4-3: Configuration parameters for Replication Agent for DB2 UDB

Configuration parameters	Description
pds_username	The IBM DB2 Universal Database user ID that the Replication Agent uses to log in to the primary database. This user ID must be granted permissions as described in "DB2 Universal Database primary database permissions" on page 47.
pds_password	The password for the IBM DB2 Universal Database user ID.
pds_host_name	The host name of the machine where the primary IBM DB2 Universal Database data server resides.
pds_port_number	The client socket port number where the IBM DB2 Universal Database data server listens for connections.
pds_database_name	The name of the primary database on the IBM DB2 Universal Database data server from which transactions will be replicated.
pds_datasource_name	The data source name or the primary database alias for the primary database in the IBM DB2 Universal Database data server.

Replication Server connectivity

Note Replication Agent uses TCP/IP and the Sybase JDBC driver (jConnect for JDBC, which is included in Replication Agent installation) to communicate with other Sybase servers. The Replication Agent does not rely on the Sybase interfaces file for connectivity information.

Table 4-4 contains a description of the Replication Agent configuration parameters that must be set to allow Replication Agent for UDB to connect to the primary Replication Server.

Table 4-4: Configuration parameters to connect to the primary Replication Server

Configuration parameters	Description
rs_host_name	The name of the host machine on which the primary Replication Server resides.
rs_charset	The character set that Replication Agent uses when creating LTL commands for Replication Server. It must match Replication Server's character set, defined by the RS_charset property in the Replication Server configuration file (*$SYBASE/REP 15_1/install/rssrvname.cfg*).
rs_port_number	The client socket port number where the primary Replication Server listens for connections.
rs_username	The user ID that the Replication Agent uses to log in to the primary Replication Server. This user ID must be defined and granted connect source permission in the Replication Server.
rs_password	The password for the user ID that the Replication Agent uses to log in to the primary Replication Server.
rs_source_ds	The host name of the machine where the primary IBM DB2 Universal Database server resides (as specified by the pds_host_name Replication Agent configuration parameter).
rs_source_db	The data source name or the primary database alias (as specified by the pds_datasource_name Replication Agent configuration parameter).

RSSD connectivity

Table 4-5 contains a description of the Replication Agent configuration parameters that must be set to allow Replication Agent for UDB to connect to the RSSD of the primary Replication Server.

Table 4-5: Configuration parameters to connect to RSSD

Configuration parameters	Description
rssd_host_name	The name of the host machine for the data server that contains the RSSD.
rssd_port_number	The client socket port number where the RSSD data server listens for connections.
rssd_username	The user ID that the Replication Agent uses to log in to the RSSD. This user ID must be defined and granted select permission in the RSSD.
rssd_password	The password for the user ID that the Replication Agent uses to log in to the RSSD.
rssd_database_name	The database name of the RSSD. When logging in to the data server specified in the rssd_host_name parameter, the Replication Agent invokes a use command for the database name specified in the rssd_database_name parameter.

DB2 Universal Database primary database configuration issues

All the installation issues and configuration parameter details for a primary IBM DB2 Universal Database data server are provided in the Replication Agent *Installation Guide*. The following are a few items that may need additional attention:

- When you install Replication Agent, a Java Runtime Environment (JRE) that is compatible with the Replication Agent for UDB is installed. For each operating system, you should download and install the most recent recommended patches specified by your operating system vendor for Java compatibility.
- All configuration parameter values in the Replication Agent configuration file are case sensitive. Be careful when specifying the values of the rs_source_ds and rs_source_db parameters, as Replication Server is also case sensitive. If the same case is not used in both Replication Agent and Replication Server parameters, no connection occurs.
- The Replication Agent filter_maint_userid configuration parameter controls whether the Replication Agent forwards transactions performed by the Maintenance User to the primary Replication Server. The Maintenance User name is defined in the Replication Server create connection command for the primary database.

In a bidirectional replication environment (replicating both into and out of the same database), the value of the filter_maint_userid parameter should be set to true. If it is not, transactions replicated to another site could return to be applied at the originating site, creating an endless loop.

- The Replication Agent ltl_character_case configuration parameter controls the character case in which the Replication Agent sends database object names to the Replication Server.

  For example, if a replication definition is created for all tables named testtab, the table name sent by the Replication Agent must be testtab, or no match occurs. Because Replication Server is case sensitive, a value of TESTTAB does not match the value of testtab.

  When you create replication definitions choose a default case (for example, create all replication definitions in either all uppercase or all lowercase), and change the value of the Replication Agent ltl_character_case parameter to match.

- In an IBM DB2 Universal Database, object names are stored in uppercase by default, if no case was forced when the object was created. That means the Replication Agent sends object names in uppercase to the primary Replication Server, unless configured to do otherwise.

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

Replication definitions for primary tables in DB2 Universal Database

The Replication Agent use_rssd configuration parameter controls whether the Replication Agent sends Log Transfer Language (LTL) that contains only the columns specified in a replication definition, or all of the columns in the primary table.

When the value of the use_rssd parameter is set to false, the Replication Agent sends LTL with data for all of the columns in the primary table. When the value of the use_rssd parameter is set to true, the Replication Agent sends LTL with data for only the columns specified in the replication definition for each primary table.

By sending data for only the columns specified in the replication definition, network traffic is reduced, which can improve performance.

In addition, column names and parameter names are removed from the LTL because the Replication Agent can send information in the order identified by the replication definition. The LTL minimal columns and structured tokens options are also available when the value of the use_rssd parameter is set to true. For more information, see the Replication Agent *Administration Guide*.

DB2 Universal Database primary datatype translation issues

The Replication Agent allows you to control how it sends the IBM DB2 Universal Database DATE, TIME, and TIMESTAMP column values to the Replication Server. There are two options:

- Send the value as a character string (the default)
- Send the value as a Sybase datetime value

The value of the Replication Agent pdb_convert_datetime configuration parameter determines how the Replication Agent handles temporal datatypes.

If you set the pdb_convert_datetime parameter to true, all corresponding datatypes in a replication definition for DATE, TIME, and TIMESTAMP columns are converted to the Sybase datetime datatype.

If you set the pdb_convert_datetime parameter to false, the datatype in a replication definition for a DATE, TIME, or TIMESTAMP column must be either:

- A Replication Server user-defined datatype (UDD), or
- A character datatype.

The character (char or varchar) datatype specified in a replication definition for for a DATE, TIME, or TIMESTAMP column must have sufficient length to accommodate the column's default display length.

See the Replication Agent *Administration Guide* for more information about the pdb_convert_datetime parameter and a complete list of datatype mapping for the IBM DB2 Universal Database datatypes. For a complete list of the IBM DB2 Universal Database datatype mapping, see the Replication Agent *Primary Database Guide*.

Microsoft SQL Server primary data servers

This section describes the primary database issues and considerations specific to the Microsoft SQL Server data server in a Sybase replication system.

Note Replication Agent for Microsoft SQL Server must be installed on Microsoft Windows.

Replication Agent for Microsoft SQL Server

As a primary data server, Microsoft SQL Server interacts with Replication Agent. The Replication Agent identifies and transfers information about data-changing operations or transactions from a Microsoft SQL Server primary database to a primary Replication Server.

Note A separate Replication Agent instance is required for each database from which transactions are replicated.

The Replication Agent interacts with the primary Replication Server and with the RSSD of the primary Replication Server, if so configured.

sybfilter driver

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

Microsoft SQL Server system management issues

The Replication Agent provides a number of commands that return metadata information about the primary database (such as database names, table names, procedure names, and column names). It does this by issuing specific JDBC calls designed to return this information or by querying the system tables directly.

Scripts to set up single-table replication from Microsoft SQL Server to ASE

The Replication Agent provides a set of sample scripts that you can use to set up simple, single-table replication from Microsoft SQL Server to Adaptive Server. These scripts can be found in the *$SYBASE/RAX-15_1/scripts* directory.

For more information about the sample scripts and their use, see the Replication Agent *Administration Guide*.

Replication Manager

The Replication Manager plug-in cannot start, but can stop a Replication Agent instance in a Microsoft SQL Server primary data server:

For more information on starting and stopping the Replication Agent instance, see the Replication Agent *Administration Guide*.

Replication Agent permissions

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

The user ID that the Replication Agent instance uses to log in to the Microsoft SQL Server must have access to the primary database. Table 4-6 lists the required permissions that are granted.

Table 4-6: Permissions required to log in to Microsoft SQL Server

Permissions	When required
create table	To create tables in the primary database.
create trigger	To create DDL triggers in the primary database.
create procedure	To create procedures in the primary database.
create role	To create the "ReplicationAdmin" role. Only the user with the "ReplicationAdmin" role can mark a table or procedure.
db_owner role	To allow Replication Agent to execute sp_repltrans and sp_repldone in the primary database. This role is also required for primary database initialization.
grant	To grant select permission on sys.sysschobs to the "ReplicationAdmin" role.
sysadmin role	For Microsoft SQL Server data server initialization and deinitialization using pdb_xlog init and pdb_xlog remove, respectively.

Primary data server connectivity

Replication Agent requires a JDBC driver to communicate with the primary database. JDBC drivers for Microsoft SQL Server databases are provided by the database vendors. If the JDBC driver for your database is not already installed, obtain the appropriate driver from the vendor's Web site.

Refer to the Replication Agent *Release Bulletin* for the version of the Microsoft SQL Server JDBC.

❖ **To set the CLASSPATH environment variable**

1 Install the JDBC driver on the host machine on which Replication Agent resides or where Replication Agent can access it.

2 Add the location of the JDBC driver to the CLASSPATH environment variable. For Microsoft Windows:

Go to Start | Settings | Control Panel | System | Environment, and add the following to the existing CLASSPATH environment variable, using the semicolon (;) as the path separator, or create the path in the User Variables panel:

```
drive:\path_name\driver
```

where:

- *drive* is the drive letter.
- *path_name* is the name of the path where you installed the JDBC driver.
- *driver* is the name of the JDBC driver. For Microsoft SQL Server, the name is *sqljdbc.jar.*

Click Apply, then OK.

Table 4-7 contains a description of the Replication Agent configuration parameters that must be set.

Table 4-7: Configuration parameters to connect to the primary data server

Configuration parameters	Description
pds_username	The Microsoft SQL Server user login that the Replication Agent uses to log in to the primary database. (This user login must be granted permissions as described in "Replication Agent permissions" on page 57.)
pds_password	The password for the user login that the Replication Agent uses to log in to the primary database.

Configuration parameters	Description
pds_dac_port_number	The dedicated administration connection port number that Replication Agent uses to connect to the primary database during the server-level initialization of the primary data server.
pds_host_name	The name of the host machine on which the Microsoft SQL Server data server resides.
pds_port_number	The client socket port number for the primary database gateway server.
pds_server_name	The server name of the Microsoft SQL Server data server.
pds_database_name	The name of the primary database on the Microsoft SQL Server data server, from which transactions will be replicated.

Replication Server connectivity

Table 4-8 lists the values of the Replication Agent configuration parameters that must be set to allow Replication Agent to connect to the primary Replication Server.

Note Replication Agent uses TCP/IP and the Sybase JDBC driver (jConnect for JDBC, which is included in Replication Agent installation) to communicate with other Sybase servers. The Replication Agent does not rely on the Sybase *interfaces* file for connectivity information.

Table 4-8: Configuration parameters to connect to the primary Replication Server

Configuration parameters	Description
rs_host_name	The name of the host machine on which the primary Replication Server resides.

Configuration parameters	Description
rs_charset	The character set that Replication Agent uses when creating LTL commands for Replication Server. It must match Replication Server's character set, defined by the RS_charset property in the Replication Server configuration file (*$SYBASE/REP-15_1/install/rssrvname.cfg)*. **Note** Setting this property to anything other than the character set of the primary Replication Server causes it to incorrectly do character set conversion of the LTL commands it receives from Replication Agent. Only if this value is different from the RA_JAVA_DFLT_CHARSET value (which should match the primary database's character set) will Replication Agent do character set conversion on the character data being replicated. Character set conversion slows performance.
rs_port_number	The port number Replication Agent uses to log in to Replication Server. Check with your System Administrator to determine which port numbers are available.
rs_username	The Replication Server client user ID that Replication Agent uses to log in to the primary Replication Server. This user ID must have connect source authority in the Replication Server. **Note** The value for the rs_username parameter must not be the same as the value for the pdb_maint_user parameter.
rs_password	The password for the user ID that the Replication Agent uses to log in to the primary Replication Server.
rs_source_db	The database name representing the primary database to which Replication Server connects. This value is specified in the Replication Server create connection command used to create the Replication Agent connection in the primary Replication Server. **Note** This name can be any name that helps you identify this as the connection to the primary database. For example, it can be the same name as the pds_database_name.

Configuration parameters	Description
rs_source_ds	The data server name representing the primary data server to which Replication Server connects. This value is specified in the Replication Server create connection command used to create the Replication Agent connection in the primary Replication Server. **Note** This name can be the name of the Replication Agent instance.

RSSD connectivity

Table 4-9 contains a description of the Replication Agent configuration parameters that must be set to allow the Replication Agent to connect to the RSSD of the primary Replication Server.

Table 4-9: Configuration parameters to connect to RSSD

Configuration parameters	Description
rssd_host_name	The name of the host machine for the data server that contains the RSSD.
rssd_port_number	The client socket port number where the RSSD data server listens for connections.
rssd_username	The user ID that the Replication Agent uses to log in to the RSSD. This user ID must be defined and granted select permission in the RSSD.
rssd_password	The password for the user ID that the Replication Agent uses to log in to the RSSD.
rssd_database_name	The database name of the RSSD. When logging in to the data server specified in the rssd_host_name parameter, the Replication Agent invokes a use for the database name specified in the rssd_database_name parameter.

Replication Agent objects in the Microsoft SQL Server primary database

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

Replication Agent creates objects in the Microsoft SQL Server primary database to assist with replication tasks. The Replication Agent objects are created by invoking the pdb_xlog command with the init keyword. The objects must be created before any primary database objects can be marked for replication.

Replication Agent object names

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

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

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

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

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

Table objects

Insert and delete permissions are granted to Public only on the DDL shadow table for the database name *prefix*ddl_trig_xxx. No permissions are granted on the other tables.

Procedure objects

Table 4-10 lists the procedure objects that are considered Replication Agent objects. The sp_SybSetLogforReplTable and sp_SybSetLogforReplProc procedures are created in the Microsoft SQL Server mssqlsystemresource system database. Although execute permission on these procedures is granted to Public, only the Replication Agent pds_username user is able to successfully execute the procedures because only the pds_username user is granted select permission on the sys.sysschobjs table. No permissions are granted on the other procedures when they are created.

Note The stored procedures listed in Table 4-10 have no effect when executed outside the context of replication.

Table 4-10: Replication Agent procedure objects

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

Marker objects

Table 4-11 lists the marker procedures and marker shadow tables that are considered Replication Agent objects. No permissions are granted when these procedures and tables are created.

Table 4-11: Replication Agent marker objects

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

Trigger objects

Table 4-12 lists Replication Agent trigger objects.

Table 4-12: Replication Agent trigger objects

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

Microsoft SQL Server primary database configuration issues

All the installation issues and configuration parameter details for a Microsoft SQL Server primary data server are provided in the Replication Agent *Installation Guide*. However, this section describes additional issues specific to heterogeneous replication.

rs_source_ds and *rs_source_db* configuration parameters

All configuration parameter values in the Replication Agent configuration file are case sensitive. Be careful when specifying the values of the rs_source_ds and rs_source_db parameters, as Replication Server is also case sensitive. If the same case is not used in both Replication Agent and Replication Server parameters, no connection occurs.

Filter_maint_userid configuration parameters

If you use a MSSQL login with sysadmin privilege as a maint_user, you should map the login to a user in the corresponding database, otherwise the Replication Agent can not filter the transaction performed by this maint_user correctly.

ltl_character_case configuration parameter

The Replication Agent ltl_character_case configuration parameter controls the character case in which the Replication Agent sends database object names to the primary Replication Server.

For example, if a replication definition is created for all tables named testtab, the table name sent by the Replication Agent must be testtab, or no match occurs. Because Replication Server is case sensitive, a value of TESTTAB does not match a value of testtab.

If you create replication definitions, choose a default case (for example, create all replication definitions in either all uppercase or all lowercase), and change the value of the Replication Agent ltl_character_case parameter to match.

The following is dependent on the collation you provided when you create the database. In a Microsoft SQL Server database, object names are stored in lowercase by default, if no case was forced when the object was created. That means the Replication Agent sends object names in lowercase to the primary Replication Server, unless configured to do otherwise.

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

Replication definitions for primary tables in Microsoft SQL Server

The Replication Agent use_rssd configuration parameter controls whether the Replication Agent sends Log Transfer Language (LTL) that contains only the columns specified in a replication definition or all of the columns in the primary table, as follows:

- When the value of the use_rssd parameter is set to false, the Replication Agent sends LTL with data for all of the columns in the primary table.
- When the value of the use_rssd parameter is set to true, the Replication Agent sends LTL with data for only the columns specified in the replication definition for each primary table.

By sending data for only the columns specified in the replication definition, network traffic is reduced, which can improve performance.

In addition, column names and parameter names are removed from the LTL because the Replication Agent can send information in the order identified by the replication definition. The LTL minimal columns and structured tokens options are also available when the value of the use_rssd parameter is set to true. For more information, see the Replication Agent *Administration Guide*.

Microsoft SQL Server primary datatype translation issues

The Microsoft SQL Server datetime datatype is compatible with the Sybase datetime datatype, so the Replication Agent pdb_convert_datetime parameter is not significant for a Microsoft SQL Server primary data server.

All Microsoft SQL Server datatypes are compatible with the corresponding Adaptive Server datatypes.

Oracle primary data servers

This section describes the primary database issues and considerations specific to the Oracle data server in a Sybase replication system.

As a primary data server, Oracle interacts with Replication Agent. The Replication Agent identifies and transfers information about data-changing operations or transactions from an Oracle primary data server to a primary Replication Server.

Note A separate Replication Agent instance is required for each Oracle instance from which transactions are replicated.

The Replication Agent interacts with the primary Replication Server and with the RSSD of the primary Replication Server, if so configured.

Note Replication Agent is a Java program. Some operating systems may require patches to support Java. Refer to the Replication Agent *Administration Guide* and the Replication Agent *Release Bulletin* for more information.

Replication intrusions and impacts in Oracle

The performance and operation of Oracle primary data servers in a Sybase replication system might be affected when the Replication Agent reads the Oracle redo and archive logs to retrieve transaction information. To provide and maintain the necessary information, the following adjustments to Oracle redo logs are required:

- Archiving of redo logs must be enabled.
- Supplemental logging of primary key and index data must be enabled.

Oracle system management issues

The Replication Agent provides a number of commands that return metadata information about the primary database (such as database names, table names, procedure names, and column names). It does this by issuing specific JDBC calls designed to return this information, or by querying the Oracle system tables directly.

Note Oracle does not support multiple databases within a single server instance as Adaptive Server Enterprise does.

Scripts to set up single-table replication from Oracle to ASE

The Replication Agent provides a set of sample scripts which can be used to set up a simple, single-table replication from Oracle to Adaptive Server. These scripts can be found in the *$SYBASE/RAX-15_1/scripts* directory of the Replication Agent installation.

For more information about the sample scripts and their use, see the Replication Agent *Administration Guide*.

Replication Agent for Oracle

This section describes replication definitions for primary tables in Oracle and Replication Manager limitations.

Replication definitions for primary tables in Oracle

The Replication Agent use_rssd configuration parameter controls whether the Replication Agent sends Log Transfer Language (LTL) that contains only the columns specified in a replication definition, or all of the columns in the primary table.

When the value of the use_rssd parameter is set to false, the Replication Agent sends LTL with data for all of the columns in the primary table. When the value of the use_rssd parameter is set to true, the Replication Agent sends LTL with data for only the columns specified in the replication definition for each primary table.

By sending data for only the columns specified in the replication definition, network traffic is reduced, which can improve performance.

In addition, column names and parameter names are removed from the LTL because the Replication Agent can send information in the order identified by the replication definition. The LTL minimal columns and structured tokens options are also available when the value of the use_rssd parameter is set to true. For more information, see the Replication Agent *Administration Guide*.

Replication Manager limitations

The Replication Manager plug-in cannot start, but can stop a Replication Agent instance in an Oracle primary data server:

See the Replication Agent *Administration Guide* for more information on starting and stopping the Replication Agent instance.

Oracle primary database permissions

The Replication Agent requires an Oracle login ID that has permission to access data and create new objects in the primary database. Table 4-13 contains the Oracle login ID that must have these required permissions.

Table 4-13: Oracle primary data required permissions

Permissions	Required to
create session	Connect to Oracle
select_catalog_role	Select from the DBA_* views
alter system	Perform redo log archive operations
execute on DBMS_FLASHBACK	Execute DBMS_FLASHBACK.get _system_change_number
alter any procedure	Start procedures for replication
create table	Create tables in the primary database
create procedure	Create rs_marker and rs_dump proc procedures
create public synonym	Create synonyms for the proc-active and shadow tables
create sequence	Support replication
drop public synonym	Drop created synonyms
select on SYS.OBJ$	Process procedure DDL commands
select on SYS.LOB$	Support LOB replication
select on SYS.COLLECTION$	Support table replication

Permissions	Required to
select on SYS.COL$	Support table replication
select on SYS.COLTYPE$	Support table replication
select on SYS.CON$	Support table replication
select on SYS.CDEF$	Support replication
select on SYS.IND$	Support index identification
select on SYS.USER$	Support replication
select on SYS.SEQ$	Support sequence replication

Note In addition to the required permissions, the user who starts the Replication Agent for Oracle instance must have read access to the Oracle redo and archive logs.

Primary data server connectivity

Replication Agent requires the following to connect to an Oracle primary data server:

- The JDBC driver must be installed and referenced in the CLASSPATH system variable of the Replication Agent host machine. Java uses the contents of the CLASSPATH system variable to identify the search locations for Java classes. For the Oracle JDBC driver, the full path and file name must be included in the CLASSPATH variable, for example, *drive:\<path_name>\ojdbc14.jar*.

  See the Replication Agent *Release Bulletin* for the version of the JDBC driver that is supported.

- For JDBC connectivity, the TNS Listener process for the Oracle primary data server must be running.

- Table 4-14 contains a description of the Replication Agent configuration parameters that must be set.

Table 4-14: Oracle Replication Agent configuration parameters

Configuration parameters	Description
pds_tns_filename	The fully-qualified file name identifying the Oracle *tnsnames.ora* file that contains connection properties for the primary Oracle data server. **Note** If both the pds_tns_filename and pds_tns_connection properties are set, the pds_host_name, pds_port_number, and pds_database_name configuration properties are not used.
pds_tns_connection	The Oracle connection name that identifies the primary database connection in the Oracle *tnsnames.ora* file. **Note** If both the pds_tns_filename and pds_tns_connection properties are set, the pds_host_name, pds_port_number, and pds_database_name configuration properties are not used.
asm_tns_filename	Identifies the Oracle *tnsnames.ora* filename where the ASM connection information is located. If this is the same *tnsnames.ora* file as configured in pds_tns_filename, you can leave this parameter unset. **Note** Set this parameter only if the redo logs of your primary Oracle are under Automatic Storage Management.
asm_tns_connection	Identifies the Oracle ASM connection name found in the *tnsnames.ora* file. **Note** Set this parameter only if the redo logs of your primary Oracle are under Automatic Storage Management.
asm_username	Identifies the Oracle user name to be used when connecting to the ASM server. **Note** Set this parameter only if the redo logs of your primary Oracle are under Automatic Storage Management.

Configuration parameters	Description
asm_password	Password for Oracle ASM access for the user specified in the asm_username. **Note** Set this parameter only if the redo logs of your primary Oracle are under Automatic Storage Management.
pds_username	The Oracle user ID that the Replication Agent uses to log in to the primary database. This user ID must be granted permissions as described in "Oracle primary database permissions" on page 68.
pds_password	Password for the Oracle user ID that the Replication Agent uses to log in to the primary database.
pds_host_name	Name of the host machine on which the Oracle data server resides.
pds_database_name	Name of the Oracle SID, from which transactions will be replicated.

Replication Server connectivity

Table 4-15 contains a description of the Replication Agent configuration parameters that must be set to allow Replication Agent to connect to the primary Replication Server.

Note Replication Agent uses TCP/IP and the Sybase JDBC driver (jConnect for JDBC, which is included in Replication Agent installation) to communicate with other Sybase servers. The Replication Agent does not rely on the Sybase interfaces file for connectivity information.

Table 4-15: Configuration parameters to connect to the primary Replication Server

Configuration parameters	Description
rs_host_name	The name of the host machine on which the primary Replication Server resides.

Configuration parameters	Description
rs_charset	The character set that Replication Agent uses when creating LTL commands for Replication Server. It must match the Replication Server character set, defined by the RS_charset property in the Replication Server configuration file (*$SYBASE/REP-15_1/install/rssrvname.cfg)*. **Note** Setting this property to anything other than the character set of the primary Replication Server may cause it to do incorrect character set conversion of the LTL commands it receives from Replication Agent. If this value is different than the RA_JAVA_DFLT_CHARSET value (which should match the primary database character set), the Replication Agent will do character set conversion. Character set conversion slows performance.
rs_port_number	The client socket port number where the primary Replication Server listens for connections.
rs_username	The user ID that the Replication Agent uses to log in to the primary Replication Server. This user ID must be defined and granted connect source permission in the Replication Server.
rs_password	The password for the user ID that the Replication Agent uses to log in to the primary Replication Server.
rs_source_ds	The server name of the primary data server specified in the Replication Server database connection.
rs_source_db	The database name of the primary database specified in the Replication Server database connection.

RSSD connectivity

Table 4-16 contains a description of the Replication Agent configuration parameters that must be set to allow the Replication Agent to connect to the RSSD of the primary Replication Server.

Table 4-16: Configuration parameters to connect to RSSD

Configuration parameters	Description
rssd_host_name	The name of the host machine for the data server that contains the RSSD.
rssd_port_number	The client socket port number where the RSSD data server listens for connections.

Configuration parameters	Description
rssd_username	The user ID that the Replication Agent uses to log in to the RSSD. This user ID must be defined and granted select permission in the RSSD.
rssd_password	The password for the user ID that the Replication Agent uses to log in to the RSSD.
rssd_database_name	The database name of the RSSD. When logging in to the data server specified in the rssd_host_name parameter, the Replication Agent invokes a use for the database name specified in the rssd_database_name parameter.

Replication Agent objects

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

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

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

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

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

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

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

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

```
pdb_xlog
```

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

Table objects

Table 4-17 lists the table that is considered a Replication Agent object.

Table 4-17: Replication Agent tables

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

Marker objects

Table 4-18 lists the Replication Agent objects related to Replication Server markers. No permissions are granted when these objects are created.

Table 4-18: Replication Agent marker objects

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

Sequences

Table 4-19 lists the Oracle sequences that are considered Replication Agent objects.

Table 4-19: Replication Agent sequences

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

Marked procedures

Table 4-20 lists the Replication Agent object that is created for each primary procedure that is marked for replication. This object is created only when a procedure is marked for replication.

Table 4-20: Replication Agent objects for each marked procedure

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

Oracle primary database configuration issues

All the installation issues and configuration parameter details for an Oracle primary data server are provided in the Replication Agent *Installation Guide*. Following are issues that you may need to review:

- When you install Replication Agent, a Java Runtime Environment (JRE) that is compatible with the Replication Agent may be installed for you. For each operating system, you should download and install the most recent recommended patches specified by your operating system vendor for Java compatibility.
- Replication Agent requires a JDBC driver for connectivity to the primary data server. Sybase does not provide a JDBC driver for Oracle data servers. You must contact your database vendor for more information about JDBC drivers for Oracle data servers.
- All configuration parameter values in the Replication Agent configuration file are case sensitive. Be careful when specifying the values of the rs_source_ds and rs_source_db parameters, as Replication Server is also case sensitive. If the same case is not used in both Replication Agent and Replication Server parameters, no connection occurs.
- The Replication Agent filter_maint_userid configuration parameter controls whether the Replication Agent forwards transactions performed by the Maintenance User to the primary Replication Server. The Maintenance User name is defined in the Replication Server create connection command for the primary database.

  In a bidirectional replication environment (replicating both into and out of the same database), the value of the filter_maint_userid parameter should be set to true. If it is not, transactions replicated to another site could return to be applied at the originating site, creating an endless loop.
- The Replication Agent ltl_character_case configuration parameter controls the case in which the Replication Agent sends database object names to the primary Replication Server.

For example, if a replication definition is created for all tables named testtab, the table name sent by the Replication Agent must be testtab, or no match occurs. Because Replication Server is case sensitive, a value of TESTTAB does not match a value of testtab.

If you create replication definitions, choose a default case (for example, create all replication definitions in either all uppercase or all lowercase), and change the value of the Replication Agent ltl_character_case parameter to match.

In an Oracle database, object names are stored in all uppercase by default, if no case was forced when the object was created. That means the Replication Agent sends object names in uppercase to the primary Replication Server, unless configured to do otherwise.

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

- No Open Client interface application (such as isql) is provided as part of the Replication Agent installation. Use the Replication Server Manager, or use an Open Client application provided with another Sybase product (for example, the isql installed with Adaptive Server or Replication Server).

Oracle primary datatype translation issues

The Replication Agent allows you to control how it sends Oracle DATE column values to the Replication Server. There are two options:

- Send the value as a character string (the default).
- Send the value as a Sybase datetime value.

The value of the Replication Agent pdb_convert_datetime configuration parameter determines how the Replication Agent handles temporal datatypes.

If you set the pdb_convert_datetime parameter to true, all corresponding datatypes in a replication definition for DATE columns are converted to the Sybase datetime datatype.

If you set the pdb_convert_datetime parameter to false, the datatype in a replication definition for a DATE column must be either:

- A Replication Server user-defined datatype (UDD), or
- A character datatype.

The character (char or varchar) datatype specified in a replication definition for for a DATE column must have sufficient length to accommodate the column's default display length.

For a full description of the pdb_convert_datetime parameter and a complete list of datatype mapping for Oracle datatypes, see the Replication Agent *Administration Guide*. For more information on UDDs and their use, see the Replication Server *Administration Guide*.

Automatic Storage Management

Replication Agent for Oracle supports the use of the Oracle Automatic Storage Management (ASM) feature for online and archive redo logs. ASM provides file system and volume management support for an Oracle database environment. ASM can be used in both Real Application Cluster (RAC) and non-RAC environments. ASM provides similar benefits as a redundant array of independent disks (RAID) or a logical volume manager (LVM). Similar to those technologies, ASM allows the definition of a single disk group from a collection of individual disks. ASM attempts to balance loads across all of the devices defined in the disk group. ASM also provides striping and mirroring capabilities. Unlike RAID or LVMs, ASM only supports files created and read by the Oracle database. ASM cannot be used for a general-purpose file system and cannot store binaries or flat files. Also, ASM files cannot be directly accessed by the operating system.

For more information about Replication Agent support for Oracle ASM, see the Replication Agent *Primary Database Guide*.

Real Application Clusters

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

Note Replication of a RAC database is the same as replication from a non-RAC database.

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

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

For more information about Replication Agent support for Oracle RAC, see the Replication Agent *Primary Database Guide*.

Replicate data servers

In addition to Adaptive Server Enterprise, Sybase replication technology supports transaction replication into the following relational database servers:

- IBM DB2 Universal Database
- Microsoft SQL Server
- Oracle

To find out more about the current supported versions of these data servers, refer to the documentation for the ECDA database gateway associated with a particular non-Sybase data server.

General issues for non-Sybase replicate data servers

There are several issues related to non-Sybase replicate data servers in a Sybase replication system. A successful replication system must address all of the issues for each replicate data server.

Non-Sybase replicate data server issues include:

- The requirements of the ECDA database gateway for the non-Sybase data server
- The access and permissions required in the replicate data server for the replication system to apply transactions to the replicate database
- The connectivity requirements for communication between the replicate data server and other components of the replication system
- The limitations on replication into the non-Sybase data server
- The intrusion and impact of the database objects required to support Replication Server operations
- The replication system management issues specific to the non-Sybase data server

IBM DB2 Universal Database replicate data servers

In the following sections describing replicate database issues, the IBM DB2 Universal Database server is treated separately for the z/OS platform, and for UNIX and Microsoft Windows platforms:

- IBM DB2 Universal Database replicate data servers on IBM z/OS
- IBM DB2 Universal Database replicate data servers on Linux, UNIX, or Windows

IBM DB2 Universal Database replicate data servers on IBM z/OS

This section describes the replicate database issues and considerations specific to the IBM DB2 Universal Database on the IBM z/OS platform in a Sybase replication system.

As a replicate data server in a gateway environment, IBM DB2 Universal Database for z/OS interacts with the Mainframe Connect DirectConnect for z/OS Option database gateway, which is responsible for accepting commands from the replicate Replication Server and applying those commands to a replicate IBM DB2 Universal Database.

As a replicate data server in a “gatewayless” environment, IBM DB2 Universal Database for z/OS interacts with Mainframe Connect DirectConnect for DB2 UDB through the AMD2 CICS transaction. AMD2 accepts commands from Replication Server and applies those commands to an IBM DB2 Universal Database. Then, AMD2 retrieves the results from those commands and returns the results to Replication Server.

Note The gatewayless environment requires a TCP/IP connection to the mainframe. For more information about gatewayless connections, see the Mainframe Connect Server Option for IBM IMS and MVS *Installation and Administration Guide*.

DB2 for z/OS replicate database permissions

To apply transactions in a replicate database, Replication Server requires a Maintenance User ID that you specify in the Replication Server create connection command. The Maintenance User ID must be defined to the IBM DB2 Universal Database for z/OS data server and granted authority to apply transactions in the replicate database. The Maintenance User ID must have the following permissions in the replicate IBM DB2 Universal Database:

- CREATE TABLE authority to create tables used for Replication Server processing
- UPDATE authority to all replicate tables and EXECUTE authority to all replicate stored procedures

Replication intrusions and impacts in DB2 for z/OS

The only significant intrusions or impacts to the replicate IBM DB2 Universal Database are the database objects created by the *hds_db2_setup_for_replicate.sql* script to support Replication Server replicate database operations. This script creates three tables in the replicate database to support Replication Server operations:

- RS_INFO, which contains information about the sort order and character set used by the replicate database.

  Note Be sure to confirm that the INSERT statements for this table (in the *hds_db2_setup_for_replicate.sql* script) specify the proper character set and sort order for your data server.

  When using Replication Server version 12.5 or later, the replicate database sort order and character set must be recorded in the RS_INFO table. To do so, use the Replication Server rs_get_charset and rs_get_sortorder functions to retrieve the character set and sort order from the RS_INFO table in the replicate database.

- RS_LASTCOMMIT, which contains information about replicated transactions applied to the replicate database.

  Each row in the RS_LASTCOMMIT table identifies the most recent committed transaction that was distributed to the replicate database from a primary database. Replication Server uses this information to ensure that all transactions are distributed.

  The Replication Server rs_get_lastcommit function retrieves information about the last transaction committed in the replicate database. For non-Sybase replicate databases, the rs_get_lastcommit function is replaced in the database-specific function-string class by the query required to access the RS_LASTCOMMIT table in the replicate database.

  Note Replication Server version 12.0 or later must use the RS_LASTCOMMIT table instead of the LTMLASTCOMMIT table that may have been installed if Replication Agent for DB2 UDB was installed.

- RS_TICKET_HISTORY, which contains the execution results of Replication Server command rs_ticket.

The rs_ticket command can be issued for the primary database to measure the amount of time it takes for a command to move from the primary database to the replicate database. You can use this information to monitor Replication Server performance, module heartbeat, replication health, and table-level quiesce. The results of each execution of rs_ticket is stored in a single row of the RS_TICKET_HISTORY table in the replicate database. You can query each row of the RS_TICKET_HISTORY table to obtain results of individual rs_ticket executions, or compare the results from different rows. The data stored in this table is not required to support replication and may be manually truncated.

Note The RS_TICKET_HISTORY table is only available in Replication Server release 15.1 and later.

Replicate database connectivity for DB2 for z/OS

A Replication Server database connection name is made up of two parts: a data server name (server_name) and a database name (db_name).

When using the Mainframe Connect DirectConnect for z/OS Option database gateway, the server_name is the name of the database gateway server, and the db_name is the name of the replicate IBM DB2 Universal Database.

The replicate Replication Server looks for an interfaces file entry for the database gateway server_name specified in the Replication Server database connection. The replicate Replication Server logs in to the replicate data server using the user_name and password specified in the database connection.

You must make an entry in the Replication Server interfaces file to identify the host and port where the Mainframe Connect DirectConnect for z/OS Option database gateway server is listening. The interfaces file entry name must match the server_name portion of the Replication Server database connection.

Gatewayless connections for DB2 for z/OS

With a gatewayless connection from Replication Server to DB2 UDB using the MainframeConnect DirectConnect for z/OS Option, the server_name is the mainframe host name, and the db_name is the name of the replicate IBM DB2 Universal Database. The interfaces file entry for the server_name maps to the mainframe IP address and port.

Gatewayless replication to IBM DB2 Universal Database for z/OS requires that you:

- Use the Mainframe Connect version 15.0 or later.

- Define the rs_get_textptr and rs_writetext function strings using the writetext method. See the Replication Server *Reference Manual* for more information.
- Run the following Replication Server HDS scripts:
  - *hds_clt_xxx_to_db2.sql* (class-level translations for the primary database)
  - *hds_clt_ase_to_xxx.sql* (execute for each database that you connect to)
  - *hds_db2_funcstrings.sql*
  - *hds_db2_udds.sql*

Replicate database limitations in DB2 for z/OS

The following replication limitations exist with an IBM DB2 Universal Database for z/OS replicate data server:

- Replication of large object (LOB) datatypes (BLOB, CLOB, and DBCLOB) is supported over a gatewayless connection using MainframeConnect for DB2 version 15.0 or later.
- Replication of large object (LOB) datatypes (BLOB, CLOB, and DBCLOB) is not supported directly by Mainframe Connect DirectConnect for z/OS Option.
- Replication Server cannot send an IBM DB2 Universal Database binary value as a binary string because the Mainframe Connect DirectConnect for z/OS Option database gateway performs an ASCII to EBCDIC translation on the value. Therefore, all binary or varbinary datatypes replicated to IBM DB2 Universal Database for z/OS must be mapped to the rs_db2_char_for_bit or rs_db2_varchar_for_bit datatype.

DB2 for z/OS replicate database configuration issues

The heterogeneous datatype support (HDS) feature of Replication Server provides a number of sample SQL scripts that help you set up the HDS feature in the replicate Replication Server and the IBM DB2 Universal Database for z/OS replicate database. These include:

hds_db2_setup_for_replicate.sql script

A script to be applied to the IBM DB2 Universal Database for z/OS replicate database.

The *hds_db2_setup_for_replicate.sql* script creates the RS_INFO, RS_LASTCOMMIT, and the RS_TICKET_HISTORY tables in the replicate database. The script includes GRANT statements that need to be changed before execution to reference the correct user ID name of the Maintenance User defined in the Replication Server database connection for the replicate database. The Maintenance User ID must have UPDATE authority on the tables.

Note Sybase recommends using the Maintenance User ID to execute this script.

To replicate large-object (LOB) datatypes, you must create rs_writetext and rs_get_textptr function strings. These are not included in the Replication Server HDS sample scripts.

For descriptions of the rs_writetext and rs_get_textptr function strings, see the Replication Server *Reference Manual*.

hds_db2_funcstrings.sql **and** ***hds_db2_udds.sql*** **scripts**

Scripts to be applied to Replication Server System Database (RSSD).

The *hds_db2_udds.sql* script adds the user-defined datatypes (UDDs) that define the attributes of IBM DB2 Universal Database native datatypes to the RSSD. The UDDs are required to ensure that datatypes received from primary transactions are properly formatted for application to the IBM DB2 Universal Database for z/OS replicate database.

Note You may need to modify the *hds_db2_udds.sql* script to reference the correct RSSD database name.

The *hds_db2_funcstrings.sql* script replaces several default Replication Server function strings with custom function strings designed to communicate with IBM DB2 Universal Database for z/OS and access the tables and procedures created by the *hds_db2_setup_for_replicate.sql* script. These function strings are added to the Replication Server default rs_db2_function_class.

Note You may need to modify the script to reference the correct RSSD database name.

If you are using a Mainframe Connect DirectConnect for z/OS Option database gateway for replication to IBM DB2 Universal Database for z/OS, you must set the following properties in the DirectConnect *db2.cfg* access service configuration file:

```
SQLTransformation=passthrough
TransactionMode=long
```

Note If you use function strings from the rs_db2_function_class in a Replication Server version prior to 12.5, do *not* run the *hds_db2_funcstrings.sql* script, as it will overwrite the function strings you are currently using.

Class-level datatype translation scripts to be applied to RSSD

Class-level translations identify primary datatypes and the replicate datatypes the data should be translated into. For example, Oracle DATE should be translated to IBM DB2 Universal Database TIMESTAMP.

Note These translations can affect Replication Server performance. Only the translations needed for your specific primary database and replicate database should be applied to the RSSD.

Class-level translation scripts supplied for the replicate IBM DB2 Universal Database for z/OS are:

- *hds_clt_ase_to_db2.sql* – translates Adaptive Server datatypes to IBM DB2 Universal Database datatypes.
- *hds_clt_udb_to_db2.sql* – translates IBM DB2 Universal Database (for UNIX and Windows) datatypes to IBM DB2 Universal Database for IBM z/OS datatypes.

- *hds_clt_msss_to_db2.sql* – translates Microsoft SQL Server datatypes to DB2 datatypes.
- *hds_clt_oracle_to_db2.sql* – translates Oracle datatypes to IBM DB2 Universal Database datatypes.
- *hds_db2_connection_sample.sql* – a script to be applied to create a connection to the DB2 database. (The connection may be to ECDA.)

  The *hds_db2_connection_sample.sql* script provides a template for creating the Replication Server database connection for a replicate IBM DB2 Universal Database for z/OS using the pre-defined IBM DB2 Universal Database function-string class provided with Replication Server.

Note You must modify the *hds_db2_connection_sample.sql* template to include your actual server, database, and Maintenance User names before executing the script.

DB2 for z/OS system management issues

The following system management issues are specific to a replicate IBM DB2 Universal Database for z/OS data server:

- The create connection command's dsi_sql_data_style parameter was used in previous versions of Replication Server to provide some data translations for the IBM DB2 Universal Database for z/OS database.

  With the introduction of heterogeneous datatype support (HDS) in Replication Server version 12.0, the create connection command's dsi_sql_data_style parameter is no longer valid. Do *not* use this parameter with Replication Server version 12.0 or later. The default setting should be " "(blank space).
- The Component Integration Services (CIS) feature allows users to access both Adaptive Server databases and non-Sybase databases on different servers, through a local Adaptive Server connection.

  CIS allows Replication Server to connect to an Adaptive Server database, using default Adaptive Server function strings and connection properties, while CIS handles the data conversion and manipulation, and all communications with the ECDA database gateway.

  See "Component Integration Services" on page 104 for a description of the advantages and disadvantages of this approach.

IBM DB2 Universal Database replicate data servers on Linux, UNIX, or Windows

This section describes the replicate database issues and considerations specific to the IBM DB2 Universal Database server on a Linux, UNIX, or Microsoft Windows platform.

As a replicate data server in a replication system, the IBM DB2 Universal Database interacts with the ECDA Option for ODBC database gateway. ECDA Option for ODBC is responsible for accepting commands from the replicate Replication Server, and applying those commands to a database residing in an IBM DB2 Universal Database server.

Replication intrusions and impacts in DB2 Universal Database

The only significant intrusions or impacts to the IBM DB2 Universal Database replicate database are the database objects created by the *hds_udb_setup_for_replicate.sql* script to support Replication Server replicate database operations.This script creates two tables in the replicate database to support Replication Server operations:

- RS_INFO, which contains information about the sort order and character set used by the replicate database.

  Note Be sure to confirm that the INSERT statements for this table (in the *hds_udb_setup_for_replicate.sql* script) specify the proper character set and sort order for your IBM DB2 Universal Database server.

  When using Replication Server version 12.0 or later, the replicate database sort order and character set must be recorded in the RS_INFO table.

  The Replication Server rs_get_charset and rs_get_sortorder functions retrieve the character set and sort order from the RS_INFO table in the replicate database.

- RS_LASTCOMMIT, which contains information about replicated transactions applied to the replicate database.

  Each row in the RS_LASTCOMMIT table identifies the most recent committed transaction that was distributed to the replicate database from a primary database. Replication Server uses this information to ensure that all transactions are distributed.

The Replication Server rs_get_lastcommit function retrieves information about the last transaction committed in the replicate database. For non-Sybase replicate databases, the rs_get_lastcommit function is replaced in the database-specific function-string class by the query required to access the RS_LASTCOMMIT table in the replicate database.

Replicate database limitations in the DB2 Universal Database

Replication of large object (LOB) datatypes (BLOB, CLOB, and LVARCHAR) is not supported directly from Replication Server to ECDA Option for ODBC.

Note For information about a possible workaround for this LOB data replication limitation, see “Large object replication” on page 27.

IBM DB2 Universal Database system management issues

The following system management issues are specific to a replicate IBM DB2 Universal Database data server:

- The create connection command’s dsi_sql_data_style parameter was used in previous versions of Replication Server to provide some data translations for the replicate database.

  With the introduction of the heterogeneous datatype support (HDS) feature in Replication Server version 12.0, the create connection command’s dsi_sql_data_style parameter is no longer valid. Do *not* use this parameter with Replication Server version 12.0 or later. The default setting should be " "(blank space).

- The Component Integration Services (CIS) feature allows users to access both Adaptive Server databases and non-Sybase databases on different servers, through a local Adaptive Server connection.

  CIS allows Replication Server to connect to an Adaptive Server database, using default Adaptive Server function strings and connection properties, while CIS handles the data conversion and manipulation, and all communications with the ECDA database gateway.

  See “Component Integration Services” on page 104 for a description of the advantages and disadvantages of this approach.

DB2 Universal Database replicate database permissions

To apply transactions in a replicate database, Replication Server requires a Maintenance User ID that you specify in the Replication Server create connection command. The Maintenance User ID must be defined at the IBM DB2 Universal Database server and granted authority to apply transactions in the replicate database. The Maintenance User ID must have the following permissions in the IBM DB2 Universal Database replicate database:

- CREATE TABLE authority to create tables used for Replication Server processing.
- UPDATE authority on all replicate tables.

Replicate database connectivity for IBM DB2 Universal Database

A Replication Server database connection name is made up of two parts: a data server name (server_name) and a database name (db_name). The server_name is the name of the ECDA Option for ODBC database gateway server, and the db_name is the name of the replicate database residing in the IBM DB2 Universal Database.

The replicate Replication Server looks for an interfaces file entry for the database gateway server_name specified in the Replication Server database connection. The replicate Replication Server logs in to the replicate data server using the user_name and password specified in the database connection.

You must make an entry in the Replication Server interfaces file to identify the host and port where the ECDA Option for ODBC database gateway server is listening. The interfaces file entry name must match the server_name portion of the Replication Server database connection.

IBM DB2 Universal Database replicate database configuration issues

The heterogeneous datatype support (HDS) feature of Replication Server provides a number of sample SQL scripts that help you set up the HDS feature in the replicate Replication Server and the IBM DB2 Universal Database replicate database.

hds_udb_setup_for_replicate.sql script

A script to be applied to the IBM DB2 Universal Database replicate database. This script creates the RS_INFO, RS_LASTCOMMIT, and RS_TICKET_HISTORY tables in the replicate database. The script includes GRANT statements that need to be changed before execution to reference the correct user ID name of the Maintenance User defined in the Replication Server database connection for the replicate database. The Maintenance User ID must have UPDATE authority on the tables.

Note Sybase recommends using the Maintenance User ID to execute this script.

hds_udb_funcstrings.sql and *hds_udb_udds.sql* scripts

Scripts to be applied to Replication Server System Database (RSSD).

The *hds_udb_udds.sql* script adds the user-defined datatypes (UDDs) that define the attributes of the IBM DB2 Universal Database native datatypes to the RSSD. The UDDs are required to ensure datatypes received from primary transactions are properly formatted for application to the IBM DB2 Universal Database replicate database.

Note You might need to modify the *hds_udb_udds.sql* script to reference the correct RSSD database name.

The *hds_udb_funcstrings.sql* script replaces several default Replication Server function strings with custom function strings designed to communicate with the IBM DB2 Universal Database and access the tables and procedures created by the *hds_udb_setup_for_replicate.sql* script. These function strings are added to the Replication Server default rs_udb_function_class.

Note You may need to modify the script to reference the correct RSSD database name.

Class-level datatype translation scripts to be applied to RSSD

Class-level translations identify primary datatypes and the replicate datatypes the data should be translated into (for example, Microsoft SQL Server binary should be translated to IBM DB2 Universal Database CHAR FOR BIT DATA).

Note These translations can affect Replication Server performance. Only the translations needed for your specific primary database and replicate database should be applied to the RSSD.

Class-level translation scripts supplied for the IBM DB2 Universal Database replicate database are:

- *hds_clt_ase_to_udb.sql* – translates Adaptive Server datatypes to IBM DB2 Universal Database datatypes.
- *hds_clt_db2_to_udb.sql* – translates DB2 for z/OS datatypes to IBM DB2 Universal Database datatypes.
- *hds_clt_msss_to_udb.sql* – translates Microsoft SQL Server datatypes to IBM DB2 Universal Database datatypes.
- *hds_clt_oracle_to_udb.sql* – translates Oracle datatypes to IBM DB2 Universal Database datatypes.

hds_udb_connection_sample.sql

A script to be applied to the replicate Replication Server.

The *hds_udb_connection_sample.sql* script provides a template for creating the Replication Server database connection for a replicate IBM DB2 Universal Database using the pre-defined IBM DB2 Universal Database function-string class provided with Replication Server.

Note You must modify the *hds_udb_connection_sample.sql* template to include your actual server, database, and Maintenance User names before executing the script.

Microsoft SQL Server replicate data servers

This section describes the replicate database issues and considerations specific to the Microsoft SQL Server data server in a Sybase replication system.

As a replicate data server, Microsoft SQL Server interacts with the ECDA Option for ODBC database gateway. The ECDA Option for ODBC server is responsible for accepting commands from the replicate Replication Server, and applying those commands to a Microsoft SQL Server database.

Note ECDA Option for ODBC supports replication of large object (LOB) datatypes (image, ntext, and text) from Replication Server directly to a Microsoft SQL Server database.

Microsoft SQL Server replicate database permissions

To apply transactions in a replicate database, Replication Server requires a Maintenance User ID that you specify in the Replication Server create connection command. The Maintenance User ID must be defined at the Microsoft SQL Server data server and granted authority to apply transactions in the replicate database. The Maintenance User ID must have the following permissions in the Microsoft SQL Server replicate database:

- create table authority to create tables used for Replication Server processing
- update authority on all replicate tables
- execute authority on all replicate stored procedures

Replication intrusions and impacts on Microsoft SQL Server

The only significant intrusions or impacts to the Microsoft SQL Server replicate database are the database objects created by the *hds_msss_setup_for_replicate.sql* script to support Replication Server replicate database operations.

The *hds_msss_setup_for_replicate.sql* script creates three tables in the replicate database to support Replication Server operations:

- RS_INFO, which contains information about the sort order and character set used by the replicate database

  Note Verify that the insert statements for the RS_INFO table (in the *hds_msss_setup_for_replicate.sql* script) specify the proper character set and sort order for your Microsoft SQL Server data server.

  When using Replication Server version 12.0 or later, the replicate database sort order and character set must be recorded in the RS_INFO table.

  The Replication Server rs_get_charset and rs_get_sortorder functions retrieve the character set and sort order from the RS_INFO table in the replicate database.

- RS_LASTCOMMIT, which contains information about replicated transactions applied to the replicate database

  Each row in the RS_LASTCOMMIT table identifies the most recent committed transaction that was distributed to the replicate database from a primary database. Replication Server uses this information to ensure that all transactions are distributed.

  The Replication Server rs_get_lastcommit function retrieves information about the last transaction committed in the replicate database. For non-Sybase replicate databases, the rs_get_lastcommit function is replaced in the database-specific function-string class by the query required to access the RS_LASTCOMMIT table in the replicate database.

- RS_TICKET_HISTORY, which contains the execution results of Replication Server command rs_ticket.

The rs_ticket command can be issued for the primary database to measure the amount of time it takes for a command to move from the primary database to the replicate database. You can use this information to monitor Replication Server performance, module heartbeat, replication health, and table-level quiesce. The results of each execution of rs_ticket is stored in a single row of the RS_TICKET_HISTORY table in the replicate database. You can query each row of the RS_TICKET_HISTORY table to obtain results of individual rs_ticket executions, or compare the results from different rows. The data stored in this table is not required to support replication and may be manually truncated.

Note The RS_TICKET_HISTORY table is only available in Replication Server release 15.1 and later.

Replicate database limitations on Microsoft SQL Server

The following replication limitations exist with a Microsoft SQL Server replicate data server:

- Microsoft SQL Server supports either 28 digits of precision or 38 digits of precision, depending on the server's start-up options. The default precision is 28 digits. Replication Server does not provide user-defined datatypes (UDDs) to support the default 28 digits of precision.

  If you attempt to replicate numeric data to a Microsoft SQL Server database in excess of the server's configured precision, Replication Server returns the following error:

  ```
  E. 2007/12/14 11:14:58. ERROR #1028 DSI EXEC(134(1)
     dcm_gabeat70_devdb.devdb)
     - dsiqmint.c(2888)
     Message from server: Message: 30291, State 0,
     Severity 19 --
     '[VENDORLIB] Vendor Library Error: [[Message
     Iteration=1|Data Source Name=mssql70_devdb|
     SQLState=22003|Native Error=1007|Message=
     [Microsoft] [ODBC SQL Server Driver][SQL
     Server]The number
     '999999999999999999.999999999999999999' is out
     of the range for numeric representation (maximum
     precision 28).
     [Message Iteration=2|SQLState=22003|Native
     Error=|Message=[Microsoft][ODBC SQL Server
     Driver][SQL Server]The number
  ```

```
'0.99999999999999999999999999999999999999' is out
of the range for numeric representation (maximum
precision 28).] <DCA>'
```

- Microsoft SQL Server appears to support identity columns in the same manner as Adaptive Server Enterprise, so the Replication Server function strings that set identity insert off and on work correctly with Microsoft SQL Server. However, to support 28-digit numeric precision, the Sybase native numeric datatype must be translated to the rs_msss_numeric datatype, and as a result of this translation, the identity characteristic is lost.

  If you choose to use the numeric to rs_msss_numeric datatype translation to support 28-digit precision in a Microsoft SQL Server replicate database, the replicate table cannot declare the numeric column receiving that data as an identity.

  If a replicate Microsoft SQL Server table declares a numeric column receiving translated data as an identity, Replication Server returns the following error:

  ```
  E. 2007/12/14 12:05:39. ERROR #1028 DSI EXEC(134(1)
  dcm_gabeat70_devdb.devdb)
  - dsiqmint.c(2888)
  Message from server: Message: 30291, State 0,
  Severity 19 --
  '[VENDORLIB] Vendor Library Error: [[Message
  Iteration=1|Data Source Name=mssql70_devdb|SQL
  Function=INSERT|SQLState=23000|Native
  Error=544|Message=[Microsoft][ODBC SQL Server
  Driver][SQL Server]Cannot insert explicit value
  for identity column in table 'ase_alltypes' when
  IDENTITY_INSERT is set to OFF.] <DCA>'
  ```

Microsoft SQL Server system management issues

The following system management issues are specific to a Microsoft SQL Server replicate data server:

- The create connection command's dsi_sql_data_style parameter was used in previous versions of Replication Server to provide some data translations for the replicate database.

With the introduction of the heterogeneous datatype support (HDS) feature in Replication Server version 12.0, the create connection command's dsi_sql_data_style parameter is no longer valid. Do *not* use this parameter with Replication Server version 12.0 or later. The default setting should be " "(blank space).

- The Component Integration Services (CIS) feature allows users to access both Adaptive Server databases and non-Sybase databases on different servers, through a local Adaptive Server connection.

  CIS allows Replication Server to connect to an Adaptive Server database, using default Adaptive Server function strings and connection properties, while CIS handles the data conversion and manipulation, and all communications with the ECDA database gateway.

  See "Component Integration Services" on page 104 for a description of the advantages and disadvantages of this approach.

Replicate database connectivity for Microsoft SQL Server

A Replication Server database connection name is made up of two parts: a data server name (server_name) and a database name (db_name). The server_name is the name of the ECDA for ODBC database gateway server, and the db_name is the name of the Microsoft SQL Server replicate database.

The replicate Replication Server looks for an interfaces file entry for the database gateway server_name specified in the Replication Server database connection. The replicate Replication Server logs in to the replicate data server using the user_name and password specified in the database connection.

You must make an entry in the Replication Server interfaces file to identify the host and port where the ECDA Option for ODBC database gateway server is listening. The interfaces file entry name must match the server_name portion of the Replication Server database connection.

Microsoft SQL Server replicate database configuration issues

The heterogeneous datatype support (HDS) feature of Replication Server provides a number of sample SQL scripts that help you set up the HDS feature in the replicate Replication Server and the Microsoft SQL Server replicate database.

hds_msss_setup_for_replicate.sql script

A script to be applied to the Microsoft SQL Server replicate database.

The *hds_msss_setup_for_replicate.sql* script creates the RS_INFO, RS_LASTCOMMIT, and RS_TICKET_HISTORY tables in the replicate database. The script includes grant statements that need to be changed before execution to reference the correct user ID name of the Maintenance User defined in the Replication Server database connection for the replicate database. The Maintenance User ID must have update authority on the tables.

Note Sybase recommends using the Maintenance User ID to execute this script.

hds_msss_funcstrings.sql and ***hds_msss_udds.sql*** scripts

Scripts to be applied to Replication Server System Database (RSSD).

The *hds_msss_udds.sql* script adds the user-defined datatypes (UDDs) that define the attributes of Microsoft SQL Server native datatypes to the RSSD. The UDDs are required to ensure datatypes received from primary transactions are properly formatted for application to the Microsoft SQL Server replicate database.

Note You might need to modify the *hds_msss_udds.sql* script to reference the correct RSSD database name.

The *hds_msss_funcstrings.sql* script replaces several default Replication Server function strings with custom function strings designed to communicate with Microsoft SQL Server and access the tables and procedures created by the *hds_msss_setup_for_replicate.sql* script. These function strings are added to the Replication Server default rs_msss_function_class.

Note You might need to modify the script to reference the correct RSSD database name.

Class-level datatype translation scripts to be applied to RSSD

Class-level translations identify primary datatypes and the replicate datatypes the data should be translated into (for example, IBM DB2 Universal Database TIMESTAMP should be translated to Microsoft SQL Server datetime).

Note These translations can affect Replication Server performance. Only the translations needed for your specific primary database and replicate database should be applied to the RSSD.

Class-level translation scripts supplied for the Microsoft SQL Server replicate database are:

- *hds_clt_db2_to_msss.sql* – translates IBM DB2 Universal Database for IBM z/OS datatypes to Microsoft SQL Server datatypes.
- *hds_clt_udb_to_msss.sql* – translates IBM DB2 Universal Database (for UNIX and Windows) datatypes to Microsoft SQL Server datatypes.
- *hds_clt_oracle_to_msss.sql* – translates Oracle datatypes to Microsoft SQL Server datatypes.

Note Class-level translations are not supplied for Adaptive Server datatypes to Microsoft SQL Server datatypes (or Microsoft SQL Server datatypes to Adaptive Server datatypes) because Microsoft SQL Server datatypes are directly compatible with Adaptive Server datatypes and they require no translation.

hds_msss_connection_sample.sql script

A script to be applied to the replicate Replication Server.

The *hds_msss_connection_sample.sql* script provides a template for creating the Replication Server database connection for a Microsoft SQL Server replicate database using the pre-defined Microsoft SQL Server function-string class provided with Replication Server.

Note You must modify the *hds_msss_connection_sample.sql* template to include your actual server, database, and Maintenance User names before executing the script.

Oracle replicate data servers

This section describes the replicate database issues and considerations specific to the Oracle data server in a Sybase replication system.

As a replicate data server, Oracle interacts with the ECDA Option for Oracle database gateway. The ECDA Option for Oracle is responsible for accepting commands from the replicate Replication Server, and applying those commands to an Oracle database.

Replication intrusions and impacts on Oracle

The only significant intrusions or impacts to the Oracle replicate database are the database objects created by the *hds_oracle_setup_for_replicate.sql* script, which creates two tables in the replicate database to support Replication Server operations:

- RS_INFO, which contains information about the sort order and character set used by the replicate database. When using Replication Server version 12.0 or later, the replicate database sort order and character set must be recorded in the RS_INFO table.

  Note You should confirm that the INSERT statements for this table (in the *hds_oracle_setup_for_replicate.sql* script) specify the proper character set and sort order for your Oracle data server.

  The Replication Server rs_get_charset and rs_get_sortorder functions retrieve the character set and sort order from the RS_INFO table in the replicate database.

- RS_LASTCOMMIT, which contains information about replicated transactions applied to the replicate database. Each row in the RS_LASTCOMMIT table identifies the most recent committed transaction that was distributed to the replicate database from a primary database. Replication Server uses this information to ensure that all transactions are distributed.

  The Replication Server rs_get_lastcommit function retrieves information about the last transaction committed in the replicate database. For non-Sybase replicate databases, the rs_get_lastcommit function is replaced in the database-specific function-string class by the query required to access the RS_LASTCOMMIT table in the replicate database.

- RS_TICKET_HISTORY, which contains the execution results of Replication Server command rs_ticket.

The rs_ticket command can be issued for the primary database to measure the amount of time it takes for a command to move from the primary database to the replicate database. You can use this information to monitor Replication Server performance, module heartbeat, replication health, and table-level quiesce. The results of each execution of rs_ticket is stored in a single row of the RS_TICKET_HISTORY table in the replicate database. You can query each row of the RS_TICKET_HISTORY table to obtain results of individual rs_ticket executions, or compare the results from different rows. The data stored in this table is not required to support replication and may be manually truncated.

Note The RS_TICKET_HISTORY table is only available in Replication Server release 15.1 and later.

Oracle system management issues

The following system management issues are specific to an Oracle replicate data server:

- The create connection command's dsi_sql_data_style parameter was used in previous versions of Replication Server to provide some data translations for the replicate database.

  With the introduction of the heterogeneous datatype support (HDS) feature in Replication Server version 12.0, the create connection command's dsi_sql_data_style parameter is no longer valid. Do *not* use this parameter with Replication Server version 12.0 or later. The default setting should be " "(blank space).

- The Component Integration Services (CIS) feature allows users to access both Adaptive Server databases and non-Sybase databases on different servers, through a local Adaptive Server connection.

  CIS allows Replication Server to connect to an Adaptive Server database, using default Adaptive Server function strings and connection properties, while CIS handles the data conversion and manipulation, and all communications with the ECDA database gateway.

  See "Component Integration Services" on page 104 for a description of the advantages and disadvantages of this approach.

Other replicate database issues for Oracle

The following issues must be considered when using an Oracle replicate data server:

- Because the Oracle data server does not support multiple databases within the same server instance, the effect of executing the Replication Server rs_usedb function is null. (The function string created by the *hds_oracle_funcstrings.sql* script performs no operation against the Oracle database for this function call.)
- In ECDA Option for Oracle version 12.0 or later, an additional trace flag allows the replicate Replication Server to control transaction commit boundaries when applying transactions to an Oracle replicate database.

  Setting the value of the ECDA autocommit trace flag to 0 (zero) in the ECDA Option for Oracle configuration file allows Replication Server to control when a COMMIT command should be sent to Oracle. When the value of the autocommit trace flag is not set, ECDA Option for Oracle commits each individual operation (INSERT, UPDATE, and DELETE) sent by the replicate Replication Server.

  Having ECDA commit each operation can cause a problem at the replicate database if an error occurs in the middle of a multiple operation transaction. In that event, the replicate Replication Server may attempt to re-send the entire transaction, while ECDA has already committed each individual operation. To avoid this problem, set the value of the ECDA autocommit trace flag to 0 (zero).

Oracle replicate database permissions

To apply transactions in a replicate database, Replication Server requires a Maintenance User ID that you specify in the Replication Server create connection command. The Maintenance User ID must be defined at the Oracle data server and granted authority to apply transactions in the replicate database. The Maintenance User ID must have the following permissions in the Oracle replicate database:

- CREATE TABLE authority to create tables used for Replication Server processing.
- UPDATE authority on all replicate tables and EXECUTE authority on all replicate stored procedures.

Replicate database connectivity for Oracle

A Replication Server database connection name is made up of two parts: a data server name (server_name) and a database name (db_name). The server_name is the name of the ECDA Option for Oracle database gateway server, and the db_name is the name of the Oracle SID for the replicate database.

The replicate Replication Server looks for an interfaces file entry for the database gateway server_name specified in the Replication Server database connection. The replicate Replication Server logs in to the replicate data server using the user_name and password specified in the database connection.

You must make an entry in the Replication Server interfaces file to identify the host and port where the ECDA Option for Oracle database gateway server is listening. The interfaces file entry name must match the server_name portion of the Replication Server database connection.

Oracle replicate database configuration issues

The heterogeneous datatype support (HDS) feature of Replication Server provides a number of sample SQL scripts that help you set up the HDS feature in the replicate Replication Server and the Oracle replicate database. These include:

- hds_oracle_setup_for_replicate.sql scripts
- hds_oracle_funcstrings.sql and hds_oracle_udds.sql scripts
- Class-level datatype translation scripts to be applied to RSSD
- hds_oracle_connection_sample.sql script

hds_oracle_setup_for_replicate.sql scripts

A script to be applied to the Oracle replicate database.

The *hds_oracle_setup_for_replicate.sql* script creates the RS_INFO, RS_LASTCOMMIT, and RS_TICKET_HISTORY tables in the replicate database. The script includes GRANT statements that need to be changed before execution to reference the correct user ID name of the Maintenance User defined in the Replication Server database connection for the replicate database. The Maintenance User ID must have UPDATE authority on the tables.

Note Sybase recommends using the Maintenance User ID to execute this script.

hds_oracle_funcstrings.sql and *hds_oracle_udds.sql* scripts

hds_oracle_funcstrings.sql and *hds_oracle_udds.sql* – scripts to be applied to Replication Server System Database (RSSD).

The *hds_oracle_udds.sql* script adds the user-defined datatypes (UDDs) that define the attributes of Oracle native datatypes to the RSSD. The UDDs are required to ensure datatypes received from primary transactions are properly formatted for application to the Oracle replicate database.

Note You might need to modify the *hds_oracle_udds.sql* script to reference the correct RSSD database name.

The *hds_oracle_funcstrings.sql* script replaces several default Replication Server function strings with custom function strings designed to communicate with an Oracle data server and access the tables and procedures created by the *hds_oracle_setup_for_replicate.sql* script. These function strings are added to the Replication Server default rs_oracle_function_class.

Note You might need to modify the script to reference the correct RSSD database name.

Class-level datatype translation scripts to be applied to RSSD

Class-level translations identify primary datatypes and the replicate datatypes the data should be translated into (for example, IBM DB2 Universal Database TIMESTAMP should be translated to Oracle DATE).

Note These translations can affect Replication Server performance. Only the translations needed for your specific primary database and replicate database should be applied to the RSSD.

Class-level translation scripts supplied for the Oracle replicate database are:

- *hds_clt_ase_to_oracle.sql* – translates Adaptive Server datatypes to Oracle datatypes.
- *hds_clt_db2_to_oracle.sql* – translates IBM DB2 Universal Database for z/OS datatypes to Oracle datatypes.
- *hds_clt_udb_to_oracle.sql* – translates IBM DB2 Universal Database (for UNIX and Windows) datatypes to Oracle datatypes.

- *hds_clt_msss_to_oracle.sql* – translates Microsoft SQL Server datatypes to Oracle datatypes.

hds_oracle_connection_sample.sql script

A script to be applied to the replicate Replication Server.

The *hds_oracle_connection_sample.sql* script provides a template for creating the Replication Server database connection for an Oracle replicate database using the pre-defined Oracle function-string class provided with Replication Server.

Note You must modify the *hds_oracle_connection_sample.sql* template to include your actual server, database, and Maintenance User names before executing the script.

Component Integration Services

Component Integration Services (CIS) is a feature of Adaptive Server Enterprise. CIS allows Adaptive Server to present a uniform view of enterprise data to client applications and provides location transparency to enterprise-wide data sources.

CIS allows users to access both Sybase and non-Sybase data servers. These external data sources include host data files and tables, views, and remote procedure calls (RPCs) in data servers such as Adaptive Server Enterprise, IBM DB2 Universal Database, and Oracle.

Replication with CIS

The CIS feature allows an Adaptive Server to act as a proxy for a non-Sybase data server. Replication Server connects to the Adaptive Server proxy database as if it were a normal Adaptive Server database. CIS then takes responsibility, along with a ECDA gateway, for translating the replication data and commands to the replicate data server's native formats.

Replication without CIS

The heterogeneous datatype support (HDS) feature of Replication Server can provide datatype translation that was previously available only with CIS. Datatype translation is accomplished using translation definitions installed in the RSSD. Command translations are provided with custom function strings and function-string classes.

Advantages and disadvantages of CIS

If you are a current user of CIS, there might not be an advantage to using the HDS feature of Replication Server. Both CIS and Replication Server HDS facilitate replication into a non-Sybase replicate database.

If you are not currently using CIS to communicate with a non-Sybase database, using the HDS feature of Replication Server provides some advantages over using CIS:

- Replication system topography is simpler without CIS, and there is no CIS overhead.
- You have some greater freedom of component location if you do not use CIS; you are not tied to an existing Adaptive Server location or required to install and maintain another Adaptive Server to decentralize the processing.
- The HDS feature of Replication Server does not support replicating to a non-Sybase data server that is not supported by HDS. The HDS feature in Replication Server supports replicate databases only in IBM DB2 Universal Database, Microsoft SQL Server, and Oracle data servers. Any other non-Sybase replicate database requires the use of CIS.

Note For more information about using CIS, refer to the *Component Integration Services User's Guide*, which is part of the Adaptive Server Enterprise documentation set.

CHAPTER 5

Replication Server Issues

This chapter describes the issues specific to Replication Server in a replication system with heterogeneous or non-Sybase data servers.

Relationship with other system components

Replication Server interacts with other components of a replication system as either a *server* or a *client*.

As a server, Replication Server supports connections from:

- Replication Agents, across which database commands are sent from primary databases
- Other Replication Servers, thus distributing the processing involved in message delivery and providing a degree of scalability in a replication system
- Users or management tools for administration, data server identification, message publication and subscription, and so on

As a client, Replication Server connects to:

- Replicate Replication Servers for message delivery
- Other Replication Servers, thus distributing the processing involved in message delivery and providing a degree of scalability in a replication system

- Replication Server System Database (RSSD) which may be on an external Adaptive Server Enterprise database or may be the internal embedded RSSD (ERSSD).
- Database gateway to connect to the replicate non-Sybase database.

Replication Server communication protocols

Replication Server is an Open Client and Open Server application that uses Sybase Tabular Data Stream™ (TDS) as the underlying communication protocol. As such, any clients that request services from Replication Server must implement an Open Client interface. This includes Replication Agents, system management tools, and user interface tools such as isql.

As a client distributing messages to other Replication Servers or to replicate data servers, Replication Server speaks with an Open Client interface. Therefore, when Replication Server needs to send a message to a data server, either that data server must support an Open Server interface running on TDS, or there must be an Open Server/TDS bridge or "gateway" application between Replication Server and the replicate data server.

Typically, this gateway software is in the form of a Sybase ECDA database gateway. Some ECDA gateways bridge from Open Server/TDS to the native interface of the replicate data server (for example, ECDA Option for Oracle), while others bridge from Open Server/TDS to an ODBC or JDBC driver for the data server. Replication Server configurations vary, depending on the gateway used.

Replication Server user IDs and permissions

Replication Server requires several different user IDs. Some user IDs are required for other components (or users) to access the Replication Server, and others are required for the Replication Server to have access to other components in a replication system.

User IDs are defined in the Replication Server using the Replication Server create connection command.

Note Depending on how your replication system is configured, some of the user IDs in the following list might not be required. For example, if you have separate Replication Servers for primary and replicate databases, the primary Replication Server does not require a user ID for access to a replicate database.

The following user IDs are defined in a Replication Server:

- Replication Agent user – used by a Replication Agent to log in to a primary Replication Server. This user ID must have connect source permission to deliver database commands through the LTL interface.
- Replication Server user – used by other Replication Servers to log in to a Replication Server and forward messages. This user ID must have connect source permission to forward database commands through the RCL interface.
- SysAdmin user – used by System Administrators or system administration tools to perform administration activities. Depending on the task, this user ID must have sa, create object, or primary subscribe permission.
- Maintenance user – used by a replicate Replication Server to deliver messages to a replicate data server. This user ID must have the necessary permissions in the replicate data server to execute the commands to which messages to be delivered are mapped.
- Replicate user – used by a replicate Replication Server to deliver messages to a primary data server. For delivery for "request" messages, that is, messages from a replicate data server that are selected for delivery to the primary data server, Replication Server uses the user ID of the user who executes the command in the replicate database. This user ID must have the necessary permissions in the primary data server to execute the commands to which messages to be delivered are mapped.
- RSI user – used by Replication Server to log in to other Replication Servers to forward messages to be delivered. This user ID must have connect source permission in the replicate Replication Server.
- RSSD user – used by Replication Server to log in to the Replication Server System Database (RSSD) that manages its operational data. This user ID must have full control in the RSSD to create and drop objects, execute procedures, and query and update tables.

Relationship with Replication Agents

While Replication Server is extensible to meet the needs of replicate data servers (customizable function strings and error handling, custom datatype definitions, and translations between datatypes), Replication Server support of primary data servers is limited.

The Replication Server interface for primary data servers is its proprietary Log Transfer Language (LTL). Transactions from a primary data server must be translated to LTL to be delivered to a primary Replication Server. Therefore, primary data servers are limited to those for which Sybase provides a Replication Agent to perform the translation to LTL for primary database operations.

Replication Server interfaces on both the primary and replicate sides are supported by the underlying Open Client/Open Server interface running on TDS.

LTM locator updates

The primary Replication Server maintains a "locator" value (LTM locator) that is used to identify the last point in a transaction log from which all data has been successfully received by the primary Replication Server. The Replication Agent periodically requests this value from the Replication Server connection to identify a position in the transaction log, which can then be used to identify where older data can be released or removed from the log.

There is a performance trade-off in determining how often to request an LTM locator update. Frequent queries of the LTM locator value from a Replication Server can slow down replication (the Replication Agent must stop sending LTL commands long enough to request and receive the LTM locator value) while it provides more frequent opportunities to release data from the primary database transaction log. When restarting, the Replication Agent must re-send all data in the log that exists since the last LTM locator value was received from Replication Server.

Generally, if replication throughput performance is a priority, you should acquire enough log resource to allow less frequent log truncation and less frequent retrieval of the LTM locator value. If log resources are scarce, more frequent retrieval of the LTM locator value and more frequent truncation may be necessary.

For more information on using the LTM locator, refer to the appropriate Replication Agent *Administration Guide*.

LTL generation

The number of bytes of information sent to Replication Server has a direct impact on the performance of the replication system; more data and commands received by Replication Server require more work and time to process. In addition, more data also requires more resources from the network environment. There are several configuration options available for the Replication Agent that you can use to minimize this impact. Among these are:

- Using the RSSD. By reading replication definitions from the RSSD, the Replication Agent can send the column data in the same column order as specified by the replication definition. This allows Replication Server to bypass sorting the column information before processing. Furthermore, column names are not sent with the data, which reduces the number of bytes of information required.
- Sending minimal columns. When an update operation occurs on a table, only a portion of the columns may have been altered. By sending the before and after images of only those columns that changed, the Replication Agent sends less information.
- Batch mode. A Replication Agent must "wrap" transactions in a limited amount of administrative LTL for the Replication Server. In batch mode, the Replication Agent can wrap multiple commands in the same set of administrative commands, which reduces the overall LTL generated and processed by the network and the Replication Server.

  In addition to batch mode, most Replication Agents have a "batch timeout" parameter, which allows a partial batch to be sent to the Replication Server after the Replication Agent waits a specified period of time and no additional transactions are received to fill the batch.

  Note Replication Agent batch mode should not be used if any Replication Server user-defined datatype (UDD) translations are used, either column-level or class-level.

- Origin time. Each transaction sent to Replication Server has an *origin queue ID*. The origin queue ID may include the time that the transaction was committed at the primary database. If the origin time is not sent by the Replication Agent, its processing effort is reduced somewhat, but the quantity of LTL sent to the Replication Server is the same.

  Note The origin time must be sent if you want to use Replication Server Manager to calculate replication latency.

For a complete description of the Replication Agent configuration parameters that affect LTL output, refer to the appropriate Replication Agent *Administration Guide*.

Database connections

Replication Server keeps track of other components in a replication system using *connections* that identify primary and replicate databases and *routes* that identify other Replication Servers.

Since Replication Server was originally designed for Adaptive Server Enterprise database replication, the definition of a connection in Replication Server follows the Sybase standard of a *server name* followed by a *database name*. For example, a Replication Server connection to an Adaptive Server named ASE1 and database PUBS would be ASE1.PUBS.

Replication Server cannot connect directly to a non-Sybase data server. In the case of a primary database, Replication Server allows a connection from a Replication Agent on behalf of the non-Sybase primary database. In the case of a replicate database, Replication Server connects to an ECDA database gateway, which in turn connects to the non-Sybase replicate data server. Since Replication Agents and ECDA gateways are not data servers themselves, the Replication Server connection properties for those components may have different meanings than they would for a database server connection.

Note If Replication Server uses CIS to connect to a replicate data server, the server name and database name are the Adaptive Server name and the Adaptive Server database name that represent the CIS connection.

A single Replication Server connection can support data flow in either one or two directions. Data flows *in* through a Replication Server connection by way of the Replication Agent User thread. Data flows *out* through a Replication Server connection by way of the Data Server Interface (DSI) thread. Each Replication Server connection can support either outbound data flow only (through the DSI thread), or both inbound and outbound data flow (through the Replication Agent User and DSI threads).

Replication Agent User thread

Replication Server receives all data-change operations or transactions to be replicated from a primary data server through the Replication Agent User thread of the database connection for that data server. Every primary database that supplies transactions to be replicated must be represented by a Replication Server database connection with an enabled Replication Agent User thread.

Replication Server establishes a connection directly with the primary database, if it resides in an Adaptive Server. If the primary database resides in a non-Sybase data server, a separate Replication Agent component communicates with the Replication Server, using a Replication Agent User thread connection, on behalf of the primary database.

Note Replication Server never attempts to connect to the Replication Agent User thread of a connection. The only entity that can initiate communication to a Replication Agent User thread is the primary data server or the Replication Agent.

On a Replication Agent User thread, the primary data server or Replication Agent is the client, and the primary Replication Server is the server.

DSI thread

The DSI thread of a Replication Server connection is where the replicated transaction is delivered by Replication Server. Every replicate database expected to receive replicated transactions must be represented by a Replication Server connection with an enabled DSI thread.

Replication Server establishes a connection directly with the replicate database, if it resides in an Adaptive Server. If the replicate database resides in a non-Sybase data server, Replication Server communicates with an ECDA database gateway (or Mainframe Connect DirectConnect for DB2 UDB in a gatewayless environment), by way of the connection's DSI thread.

Note A replicate data server or database gateway never attempts to connect to the DSI thread of a connection. The only entity that can initiate communication to a DSI thread is the Replication Server.

On a DSI thread, the Replication Server is the client, and the replicate data server or database gateway is the server.

Replication Agent connections

A Replication Agent sends data one-way to Replication Server. It logs in to the Replication Server, connects as an LTM thread to the Replication Agent User thread of a Replication Server connection, and communicates with Replication Server over that connection. This has the following implications:

- A valid user ID, which the Replication Agent uses to log in to the Replication Server, must be defined at the Replication Server.
- The Replication Agent user ID must be granted connect source permission in Replication Server. connect source permission allows the Replication Agent to send commands that are valid only on a Replication Agent User thread.
- The Replication Agent must record this user ID and associated password.
- The Replication Agent must record the *server* and *database* portions of the Replication Server connection definition to identify and connect to the correct Replication Agent User thread.
- The user_name and password defined in the Replication Server create connection command must be a valid user ID and password for the primary database represented by the Replication Agent.

  The Replication Agent validates that the connection user_name exists in the primary database. However, Replication Server does not know if (or when) a DSI thread will be used. Therefore, the user ID and password must be valid in case the DSI thread is active.

Note The requirement for a valid primary database user ID varies by Replication Agent. Some Replication Agents do not require (nor do they check for) a valid user ID on the Replication Server connection.

Interfaces file

For the interaction between a Replication Agent and a Replication Server, the only interfaces file entry that may be required is one identifying the Replication Server.

The Replication Agent for DB2 UDB requires an interfaces file entry for the Replication Server.

The Replication Agent (for DB2 Universal Database on UNIX and Windows platforms, Microsoft SQL Server, and Oracle) does not require an interfaces file entry, as it records the Replication Server host name and port number in configuration parameters.

The values of the server and database names are not significant in a Replication Server connection definition for a Replication Agent User thread. However, they must be properly configured in the Replication Agent.

The Replication Agent rs_source_ds and rs_source_db configuration parameters must be configured to match the ds.db values of the Replication Server connection name. If they do not match, the Replication Agent will not find a Replication Server connection to send information to.

Since Replication Server never attempts to contact the Replication Agent User thread of a connection, if the connection is only for a Replication Agent User thread, there is no need for the server name portion of the connection to exist in an interfaces file. The database name portion of the Replication Server connection is also arbitrary, if the connection is only for a Replication Agent User thread.

Any unique combination of two strings (namex.namey) can be used to identify a Replication Agent User thread connection. For convenience, you should choose names that have some meaning in your particular environment (for example, host.db2_subsystem, host.oracle_sid, and so on).

ECDA database gateways

In a Sybase replication system, the purpose of an ECDA database gateway is to apply transactions from a Replication Server to a non-Sybase replicate database.

To accomplish this, Replication Server logs in to the ECDA gateway using the information specified for a Replication Server connection. Replication Server logs in to the server using the user_name and password, and issues a use database command for the database defined in the connection.

For Replication Server, there is nothing to distinguish an ECDA gateway from an Adaptive Server replicate database. Replication Server delivers the same commands—and expects the same results—from any DSI thread it communicates with.

This has the following implications:

- A valid user ID, which the Replication Server uses to log in to the replicate database, must be defined in a Replication Server connection.
- This user ID must be granted permissions to update replicate tables and execute replicate procedures.
- The replicate database must be able to maintain a RS_LASTCOMMIT table and a RS_TICKET_HISTORY table and support rs_get_lastcommit functionality.

  Replication Server provides sample scripts to set up the tables and functions required for a replicate database in DB2 Universal Database, Microsoft SQL Server, and Oracle databases.

  For an overview of the expectations of a replicate data server and gateway, see Chapter 6, “Replicating Data into Foreign Data Servers,” in the Replication Server *Design Guide*.
- Datatype representations must be translated to match the native datatypes of the replicate database. Replication Server provides sample scripts to set up the function strings, function-string classes, and base datatype definitions and translations necessary to support replication into DB2 Universal Database, Microsoft SQL Server, and Oracle data servers.
- The Replication Server command resume connection attempts to initiate activity with the DSI thread of the specified connection. For an ECDA, this is a sequence of logging in to the DirectConnect server, accessing the RS_LASTCOMMIT table in the replicate database, and then applying transactions to the replicate database. Any failure in this sequence is recorded as a failure in the Replication Server log.

Interfaces file

Replication Server is an Open Server application. In an Open Server application, the preferred method for determining the location (host and port number) of another Open Server is to look up the location in an interfaces file. The interfaces file contains a list of labels, typically server names, of which each have a corresponding host name and port number, where the identified server should be “listening” for login requests.

In the interaction between an ECDA database gateway and a Replication Server, the interfaces file is important. Because the Replication Server does attempt to log in to the server identified by the server name in the Replication Server connection, that server name must exist in the Replication Server interfaces file. In addition, the interfaces file entry must also exist as a service name in the ECDA gateway configuration file entries.

A single ECDA can act as a gateway for one or many different database instances. In the ECDA configuration, each database to be accessed by the ECDA is configured as a unique *service name*. For ECDA to know which configured service name a client wants to connect to, it uses the server name passed at login time and expects to find a matching service name to use to complete the connection. Not only must the Replication Server server_name in the connection match an interfaces file entry, but that interfaces file entry must match the DirectConnect service name describing the database you want to connect to. For more information on the role of service names and their configurations, refer to the ECDA *Access Service User's Guide*.

Connection shared by Replication Agent and ECDA

A single Replication Server connection can support both an ECDA gateway and a Replication Agent, because each of these components connects to the Replication Server on a different thread. If you replicate information both into and out of the same database, having a common connection for both a database gateway and a Replication Agent can make the replication system network topology less resource intensive. On the other hand, because the Replication Agent and the ECDA gateway are two separate servers, it might be more intuitive to have a separate connection for each one.

To share a Replication Server connection between an ECDA gateway and a Replication Agent, you must define the connection to correctly support the replicate database, then configure the Replication Agent appropriately:

- In the Replication Server, use the create connection command to define the server_name and database_name for the connection. The server_name value must match a configured service name in the ECDA.
- In the Replication Agent, set the value of the rs_source_ds parameter to that server_name, and set the value of the rs_source_db parameter to the desired database_name.

Maintenance User

This section describes the role and usage of the Replication Server Maintenance User.

Maintenance User purpose

To update replicated data, Replication Server logs in to the replicate data server as the Maintenance User. The Database Owner (or the System Administrator) must grant to the Maintenance User the permissions required to insert, delete, and update rows in replicated tables, and to execute replicated stored procedures. In an Adaptive Server replicate database, Sybase Central or rs_init creates the user ID for the Replication Server Maintenance User and adds the user to the replicate database.

The Maintenance User ID and password are defined to Replication Server with the Replication Server create connection command for the replicate database. In Adaptive Server, Sybase Central or the rs_init program executes this command automatically. If you change the password for the Maintenance User ID in the data server, you can use Sybase Central or the Replication Server alter connection command to change the password for the Replication Server connection.

The Replication Server Maintenance User must also have permission to access the rs_lastcommit and rs_info system tables in the replicate database, and any stored procedures that use those tables.

Neither Sybase Central nor rs_init grants database permissions to the Maintenance User for user tables and stored procedures. You must grant database permissions on replicated tables and stored procedures before you can replicate transactions for replicated tables or replicate executions of the replicated stored procedures. For each table replicated in the database, and for each stored procedure executed due to replication, you must execute the following grant command:

```
grant all on table_name to maint_user
```

Alternatively, you can assign the Maintenance User ID (maint_user) to a Database Administrator role, if that role has the required authority on all replicate objects.

Replication Agent Maintenance User processing

When the Replication Agent connects to a Replication Server connection, the Replication Agent requests the user ID of the Maintenance User and validates that the user ID exists in the primary database. This validation requires that the Maintenance User ID defined in any Replication Server connection be valid for the database the connection represents, regardless of whether that connection is for primary transactions only, replicate transactions only, or both.

The Replication Agent does not log in to the primary database using the Maintenance User ID. Other than validating that the user ID exists, the only reference the Replication Agent makes to the Maintenance User ID is to filter out primary database transactions created by the Maintenance User.

The Replication Agent filters out Maintenance User transactions to avoid having a transaction applied more than once to the primary database. In a bidirectional replication scheme, replication can occur both to and from the same database (a single database having both a primary and a replicate role). When a primary transaction is applied to a replicate database, the applying user ID is the Maintenance User for the replicate database. A Replication Agent scanning transactions at the replicate database must ignore the transactions applied by the Replication Server Maintenance User to prevent those transactions from being sent back and applied to the primary database.

The Replication Agent accesses the database using a user ID defined at the primary database (or for DB2, a user ID that can access the DB2 log files). This user ID is not the same as the Maintenance User defined in the Replication Server connection. The Replication Agent user ID used to access the primary database has a different role and purpose than the Maintenance User defined to apply replicated transactions.

There may also be another user ID defined to the Replication Agent that is used to administer the Replication Agent. This user ID is also separate from the Replication Server Maintenance User that applies replicate transactions.

A Replication Agent can have knowledge of three different users:

- A user ID defined at the primary database, which the Replication Agent uses to log in to the primary data server and manipulate primary replication objects or read the database transaction log.
- A user ID that can log in to the Replication Agent and issue Replication Agent commands and configure Replication Agent parameters.

- A Maintenance User ID, defined at the primary database and recorded in the primary Replication Server connection. The Replication Agent validates this user ID on behalf of the Replication Server, and the Replication Agent can be configured to ignore transactions that are created by this user ID.

ECDA Maintenance User processing

In general, using an ECDA database gateway does not change the use or purpose of the Replication Server Maintenance User. The only difference is that Adaptive Server provides some utilities that automatically create and assign permissions to the Maintenance User (for example, rs_init). With non-Sybase databases, the Maintenance User ID must be defined and granted the appropriate permissions.

Different types of users

In a Sybase replication system, there are several unique user IDs required. Sybase recommends that you provide a unique user ID for each role and component combination in your environment.

Each component (database, Replication Server, Replication Agent, and ECDA) must have some user ID defined to maintain or administer that component. Typically, maintenance tasks involve changing configuration parameters and access for the component "server."

For Sybase products, the default component System Administrator user ID is typically sa. While it may be tempting to leave all of the components with an sa user defined, you may grant access to more components and more authority than intended or prudent. Do not confuse a component System Administrator user with a Replication Server connection Maintenance User. The Replication Server Maintenance User has authority to apply database transactions at a replicate database. It does not require authority to maintain the configuration of a replication system component.

Each Replication Agent requires a user ID in the primary database that can modify primary tables and procedures marked for replication and scan log files. This user ID does not perform the same function as the Replication Server Maintenance User ID, which is responsible for applying replicate transactions in a replicate database.

Maintenance User checklist

Following are the requirements for the Replication Server Maintenance User:

- A Maintenance User ID must be created in every primary and replicate database in a replication system.
- The Maintenance User ID must be identified in the Replication Server create connection command for every primary and replicate database in a replication system.
- For replicate databases, the Maintenance User ID must have authority to apply replicate transactions and update any Replication Server tables or functions installed at the replicate database (for example, RS_INFO, RS_LASTCOMMIT, and RS_TICKET_HISTORY tables).

Replication Server heterogeneous datatype support

Reconciling incompatible data representations between different types of data servers is one of the major issues that must be addressed to implement a successful heterogeneous replication system.

Heterogeneous datatype support (HDS) allows you to specify automatic datatype translation when primary and replicate data servers use datatypes that are not directly compatible.

The HDS feature provides datatype definitions for the following non-Sybase data servers:

- DB2 Universal Database
- Microsoft SQL Server
- Oracle

Datatype translation

In general, datatype translation can be divided into two categories: trivial and nontrivial.

Trivial datatype translation

Trivial datatype translations can be reduced to a translation between two Replication Server native datatypes (for example, int to char, datetime to char, or decimal to real). These translations are supported by the Open Client/Server function cs_convert. Refer to the Open Client/Server documentation on cs_convert for the kinds of translations supported and any limitations.

Nontrivial datatype translation

Nontrivial datatype translations are those in which the target datatype is a user-defined datatype, and where the user-altered attributes include more than just delimiters. For example, translation from the Sybase datetime datatype to the DB2 TIMESTAMP datatype is considered nontrivial, because these two temporal datatypes support different ranges of values. However, a translation from binary to the DB2 CHAR FOR BIT DATA datatype is trivial, because the two sets of attributes are identical except for their delimiters.

Levels of datatype translation

In Replication Server, datatype translations can be specified at:

- The function-string class level (or class-level translations)
- The replication definition level (or column-level translations)

Each Replication Server connection is associated with exactly one function-string class. Therefore, the datatype translations defined for a function-string class are associated with each connection that is defined with that function-string class. Class-level translations define default translations for the connection by specifying a set of source and target datatypes so that any value sent over that connection with a datatype that matches the source datatype is translated into the associated target datatype.

Unlike function strings, class-level translations are not inherited. Moreover, class-level translations offer the only means to translate values associated with system-defined variables.

Replication Server has no user interface for creating class-level translations. Therefore, Sybase supplies SQL scripts to create class-level translations that support some of the more common datatype translations.

Column-level translations can be specified in a replication definition for a given column by using the map to clause of the create replication definition or alter replication definition command. You can use column-level translations to fine-tune datatype translation on a column-by-column basis, per replication definition.

Column-level translations provide a performance advantage over class-level translations. However, only class-level translations can be used to translate values of system-defined variables.

HDS issues and limitations

This section describes some known issues and limitations with the HDS feature in Replication Server.

Source value exceeds target datatype bounds

The datatype translations provided by Sybase specify that the thread attempting a translation where the source value exceeds the bounds of the target datatype should be stopped with the following error message:

```
E. 2007/12/14 11:14:54. ERROR #32055 DSI EXEC(135(1)
   snickers_dco.ora805) -
   /nrm/nrm.c(7023)
   Class Level translation for column/parameter
   'datetimecol' failed.
   Source DTID is 'datetime'.
   Target DTID is 'rs_oracle_datetime'.
   Function String Class ID 'rs_oracle_function_class'.
   Value length is '21'; Maximum target length is '20';
   The value is '99991231 23:59:59:010'
```

Typically, these are the most difficult translation problems to diagnose because there appears to be no problem with either the pairing of source/target datatypes or the value to be translated.

To diagnose this type of problem, you have to be familiar with the datatype value boundary limits of all the translated target datatypes. For example, to diagnose the error shown, you have to know that the upper boundary of an Oracle DATE value is 12/31/4712.

There are other options:

- Use the maximum value of the datatype definition.
- Use the minimum value for the datatype definition.
- Use the default value for the datatype definition.

The only way you can specify these options is to edit the minimum and maximum boundary error action column values in each *hds_xxx_udds.sql* script and re-install the datatype definitions by re-executing the *hds_xxx_udds.sql* scripts in the RSSD.

Exact numeric datatype issues

There may be problems with exact numeric datatypes when the values replicated are at the boundaries (maximum or minimum values) of what is supported by the datatype definitions.

Microsoft SQL Server supports either 28 or 38 digits of precision, depending on how the server is started. By default, Microsoft SQL Server supports 28 digits of precision.

Sybase does not provide datatype definitions that support the default 28 digits of precision. Datatype definitions are not needed to support 38 digits of precision, because the Replication Server native numeric datatypes support up to 72 digits of precision.

When a number exceeds numeric precision of the Microsoft SQL Server replicate database, Replication Server returns the following error:

```
E. 2007/12/14 11:14:58. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|Data Source
   Name=mssql70_devdb|SQLState=22003|Native
   Error=1007|Message=[Microsoft][ODBC SQL Server
   Driver][SQL Server]The number
   '9999999999999999999.9999999999999999999' is out of
   the range for numeric representation (maximum
   precision 28).[Message Iteration=2|SQLState=22003|
   Native Error=|Message=[Microsoft][ODBC SQL Server
   Driver][SQL Server]The number
   '0.99999999999999999999999999999999999999' is out of
   the range for numeric representation (maximum
   precision 28).] <DCA>'
```

The most difficult numeric datatype issues involve precision and scale. Replication Server does not allow the precision and scale of a decimal datatype to be specified. A datatype definition can specify the maximum precision and maximum scale to be supported. However, if this does not equate to the specified precision and scale of an individual replicate column, then as the data approaches values near or at the boundaries, you may encounter problems that are reported differently, depending on the replicate data server.

For example, suppose you have a primary column declared as decimal (8,5) (8 digits of precision and a scale of 5), and suppose the replicate column is declared as decimal (6,4), even though the replicate data server can support a maximum of 7 digits precision and a scale of 7. In the replication definition, you specify the translation for the primary data server decimal datatype and for which there is a class-level translation to the replicate data server decimal datatype. Both datatype definitions specify the associated data server's maximum precision and scale.

If the value 999.99999 comes from the primary database, and the replicate data server's datatype definition specifies that rounding should be attempted, Replication Server attempts to apply a value of 1000.000. Even though this value satisfies the replicate database requirements for maximum precision and scale, it fails the precision and scale specified for this particular column. And if you specify for the replicate database's datatype definition that it should replace the value with the specified maximum value for the datatype definition, Replication Server attempts to apply a value of 9999999, which also fails the specified precision and scale for this particular column.

Error messages you might see from various data servers in this case include:

- The following DB2 error:

```
E. 2007/12/14 15:03:11. ERROR #1028 DSI EXEC(129(1)
   dwm5_via_rct.dwmdbas)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|SQLState=22003|Native Error=
   -413|Message=[Sybase][ClearConnect ODBC][DB2]The
   decimal or numeric value had an incorrect wire
   length compared to its specified FDOCA length
   10000000000000000000.00000000000] <DCA>'.
```

- The following Microsoft SQL Server error:

```
E. 2007/12/14 12:29:16. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1|Data Source Name=mssql70_devdb|SQL
   Function=INSERT|SQLState=22003|Native Error=
   8115|Message=[Microsoft][ODBC SQL Server Driver]
   [SQL Server]Arithmetic overflow error converting
```

```
numeric to data type numeric.[Message Iteration=
2|SQLState=01000|Native Error=|Message=
[Microsoft][SQL Server]The statement has been
terminated.] <DCA>'
```

Numeric translation and identity columns in Microsoft SQL Server

Replication Server function strings to set identity insert off and on work in Microsoft SQL Server because it supports identity columns in the same manner as Adaptive Server. However, to support 28-digit precision in a Microsoft SQL Server database, the numeric datatype must be translated to the rs_msss_numeric datatype, and as a result, the identity characteristic is lost. To avoid this problem, the Microsoft SQL Server replicate table must not declare a translated numeric column as an identity.

If you attempt to replicate a translated numeric datatype into an identity column in Microsoft SQL Server, you receive an error similar to this:

```
E. 2007/12/14 12:05:39. ERROR #1028 DSI EXEC(134(1)
   dcm_gabeat70_devdb.devdb)
   - dsiqmint.c(2888)
   Message from server: Message: 30291, State 0,
   Severity 19 --
   '[VENDORLIB] Vendor Library Error: [[Message
   Iteration=1 |Data Source Name=mssql70_devdb|SQL
   Function=INSERT|SQLState=23000|Native Error=544
   |Message=[Microsoft][ODBC SQL Server Driver][SQL
   Server]Cannot insert explicit value for identity
   column in table 'ase_alltypes' when IDENTITY_INSERT
   is set to OFF.] <DCA>'
```

Datatypes and datatype definitions

Datatype definitions for a particular data server datatype are grouped in a *datatype class*.

For more information about datatype definitions (user-defined datatypes), see the description of the RSSD rs_datatype table in the Replication Server *Reference Manual*.

Restricted datatype

The rs_address datatype cannot be used as either the source or target of column-level or class-level translations.

Error and function-string classes for non-Sybase data servers

Sybase provides function-string classes and associated function strings for all supported non-Sybase replicate data servers. Sybase does not provide unique error classes for non-Sybase data servers.

The following limitations apply to HDS function strings:

- The function-string classes are loaded into the RSSD when the RSSD is installed. However, the individual function strings must be loaded separately.
- The function strings are designed to work with the replicate setup scripts provided by Sybase. Using any other technique to set up the replicate data server, in particular creating the Replication Server support objects (tables), may result in the function strings not working properly.
- After the function strings are installed, the Replication Server must be restarted. This is because, with the exception of "input template" function strings, Replication Server caches all function strings at start-up.

Object publication and subscriptions

The following limitations apply to object publications and subscriptions in a Sybase replication system:

- When declaring columns in a replication definition for a non-Sybase primary database, you must use the Replication Server datatype that matches the datatype of the column in the primary database. If there is no matching native Replication Server datatype, you must find a datatype definition that matches the primary database datatype.
- When creating subscriptions with `where` clauses predicated on a column involved in column-level translation, you must specify the predicate value in "declared" format (that is, before translation).

Command batching for non-ASE servers

Replication Server provides the ability to batch commands for non-ASE database servers. By batching commands, you may be able to achieve improved performance of Replication Server. Support for command batching requires the following:

- Using the two function strings, rs_batch_start and rs_batch_end.
- Using the DSI connection parameters to control the processing of the two function strings.

Using function strings

Support for command batching to non-ASE servers is achieved through the use of two function strings, rs_batch_start and rs_batch_end. These function strings store the SQL translation needed for marking the beginning and end of command batches. Use of these function strings is not necessary for ASE or any other data server where the function strings rs_begin and rs_commit already support the needed functionality

Using connection settings

A DSI connection parameter, use_batch_markers, is used to control the processing of the two function strings, rs_batch_start and rs_batch_end. use_batch_markers can be set with the alter connection and configure connection commands. If use_batch_markers is set to *on* the function strings rs_batch_start and rs_batch_end are executed. The default is *off*.

Note This parameter is only to be set to *on* for replicate data servers that require additional SQL to be sent at the beginning and end of a batch of commands that are not contained in the rs_begin function string.

Order of processing

When a connection is configured to use the batch marker function strings, statements are sent to the data server in the following order:

1 The rs_begin command is sent to the replicate data server first, either separately or grouped with the batch of commands, based on the configuration parameter batch_begin as it is with current functionality.

2 The rs_batch_start command is processed and sent only when use_batch_markers is configured to true.

The rs_batch_start marker is grouped with the commands being sent as a batch. Valid rs_begin and rs_batch_start function strings allows processing of both single and batched transactions to the data servers.

3 A batch of commands is sent to the replicate data server.

The size of the batch varies, and sending of the batch follows the existing rules for terminating the grouping and flushing of the commands to the replicate data server. These commands contain a command separator between each individual command.

4 The rs_batch_end command is the last command in the batch of commands. The rs_batch_end marker is sent only when the configuration parameter use_batch_markers is set to true.

The rs_batch_start, a batch of commands, and rs_batch_end may be repeated if more than one batch is required when commands have been flushed by limits such as dsi_cmd_batch_size.

5 After the final rs_batch_end command has been sent, the rs_commit command is sent to the replicate data server. The rs_commit is processed according to the present rules.

DSI Configuration

There are three DSI configuration parameters that need to considered for each connection that will be batching commands:

- batch
- batch_begin
- use_batch_markers

The following sections describe the configuration for the data servers and databases.

ASE, ASA, and Microsoft SQL Server

For these three data servers, use the following configuration:

```
batch = on

batch_begin = on or off
```

The use of *on* reduces the number of network transfers.

```
use_batch_markers = off
```

Additional batch markers are not required for these data servers.

DB2 and UDB

For these two databases, use the following configuration:

```
batch = on
batch_begin = off
dsi_cmd_separator = ";"(semicolon)
```

DB2 and UDB use the re_begin function string for a setup command to ECDA that cannot be batched.

```
use_batch_markers = on
```

Oracle

For Oracle data servers, use the following configuration:

```
batch = on
batch_begin = off
```

As a result of a placeholder command that is used in the rs_begin function string, setting batch_begin to *on* can cause problems with starting DSI correctly. By setting batch_begin to *off* allows the rs_begin and the rs_commit commands to be sent independently of the batches of commands and insures correct SQL in all the commands being transferred.

```
use_batch_markers = on
```

Oracle requires a BEGIN and END to mark batches of commands that are not required for commands being sent individually. By configuring use_batch_markers to *on*, these are added from the rs_batch_start and rs_batch_end function strings.

Emulating rs_init activity for a non-Sybase database

The rs_init utility is a Replication Server utility that can add an Adaptive Server database to a replication system, as either a primary or replicate database. When adding a replicate database, rs_init does the following:

- Creates a Replication Server connection for the database
- Creates a Replication Server Maintenance User ID for the database
- Creates tables and procedures in the replicate database to support replication

To support the same functionality for a non-Sybase replicate database, Replication Server provides sample scripts. These scripts are located in the *$SYBASE/$SYBASE_RS/scripts* directory.

To use these scripts, you must perform the following steps manually:

1 At the replicate database:

- Create a Maintenance User ID, with permissions required to update and execute all replicate transactions.
- Execute the *hds_xxx_setup_for_replicate.sql* script.

  This script creates the rs_info and rs_lastcommit tables in the replicate database.

  This script includes grant statements that need to be changed before execution to reference the correct name of the Maintenance User ID defined in the Replication Server connection. The Maintenance User ID must have update authority to the rs_info and rs_lastcommit tables.

  Note Sybase recommends using the Maintenance User ID to execute this script.

2 At the Replication Server System Database (RSSD):

- Execute the *hds_xxx_udds.sql* script.

  This script adds the user-defined datatypes (UDDs), which define the attributes of the replicate database native datatypes to the Replication Server RSSD. These datatype definitions are required to ensure that datatypes received from primary database transactions are properly formatted for application to the replicate database. You may need to modify the script to reference the correct RSSD database name.

- Execute the *hds_xxx_funcstrings.sql* script.

  This script replaces several default Replication Server function strings with custom function strings designed to communicate with the replicate database and access the tables and procedures created by the *hds_xxx_setup_for_replicate.sql* script. These function strings are added to the Replication Server rs_xxx_function_class. You may need to modify the script to reference the correct RSSD database name.

  Note If you use older function strings that were supplied for DB2 in a version of Replication Server prior to 12.0, do *not* run the *hds_db2_setup_for_replicate.sql* script because the existing rs_db2_function_class function strings will be replaced.

- Execute the class-level translation scripts.

You must always execute the *hds_clt_ase_to_xxx.sql* script for your replicate *xxx* database. This allows you to do the translations that are required to support replication of the data found in the rs_update_lastcommit function. For example, data-like timestamps that would otherwise be in ASE datetime format.

Class-level translations identify primary database native datatypes and the replicate database native datatypes the data should be translated into. For example, Oracle DATE should be translated to DB2 TIMESTAMP.

Class-level translations impact Replication Server performance. Only the translations that are needed should be applied to the RSSD. The following scripts are an example for DB2:

- *hds_clt_ase_to_db2.sql* – translations for Adaptive Server datatypes to DB2 datatypes.
- *hds_clt_msss_to_db2.sql* – translations for Microsoft SQL Server datatypes to DB2 datatypes.
- *hds_clt_oracle_to_db2.sql* – translations for Oracle datatypes to DB2 datatypes.

3 At the replicate Replication Server:

- Execute the *hds_xxx_connection_sample.sql* script.

  This script provides a template for creating the Replication Server connection for a replicate database, using the default database-specific function-string class provided with Replication Server. Before executing the script, the template must be modified to include your actual replicate data server name, database name, and Maintenance User ID.

Note The Replication Server create connection command uses the dsi_sql_data_style option, even for DB2. If you use this option, you effectively turn off datatype translations for that connection. The dsi_sql_data_type option remains available in Replication Server for backward compatibility only.

Case sensitivity in publications and subscriptions

In a replication system with only Sybase databases, case sensitivity is not a major issue because Sybase databases all use the same object name case conventions. In Sybase databases, uppercase and lowercase are both allowed, with the default being all lowercase.

Non-Sybase data servers have different object name case conventions. Like Sybase, Microsoft SQL Server defaults to all lowercase. Both DB2 and Oracle default to all uppercase.

Because Replication Server follows the case conventions of Sybase databases, the default for database object names (for example, in replication definitions and subscriptions) is all lowercase.

Replication Server object names are case sensitive

Replication Server object names (connections, tables, procedures, columns, subscriptions, and replication definitions) are *case sensitive*. The exact case must be provided whenever you identify or reference those objects.

In a heterogeneous replication system, you must be aware of object name case conventions in all of the components to ensure that the connection attempts from the Replication Agent to the Replication Server, and the LTL generated by Replication Agent, match the connection names and replication definition object names known to Replication Server.

Replication Agent uses a configuration parameter (ltl_character_case) to specify the case used when communicating with Replication Server. For more information, see the Replication Agent *Administration Guide*.

Object names are not case sensitive

In a heterogeneous replication system, it is possible to have a variety of servers with different case sensitivity or different default display formats. If the primary and replicate data servers store database object names in different cases, you may have to explicitly specify the proper “replicate name” to be used for table and column names when you create replication definitions in Replication Server. The following example illustrates this.

Oracle case sensitivity example

By default, Oracle processing of object names is not case sensitive, however, object names are physically stored in all uppercase. The statement create table test1 (col1 int) creates a table named "test1," which is identified in the Oracle system catalog with name "TABLE1." The following SQL statements both return the same result from the same table:

```
select from TEST1
```

and

```
select from test1
```

As Replication Server processes information, it requires the LTL from the Replication Agent to match the character case of the table and column names specified in the replication definition. As the data moves to the replicate site, the commands created by Replication Server to be applied to the replicate database use the case specified in the replication definition for table and column names (for example, insert into TEST1 (COL1) values (1)).

If the *replicate* database is case sensitive, the commands presented by Replication Server may be rejected with an "object not found" error. In this example, if the replicate database is Adaptive Server (and the Adaptive Server is installed as case sensitive) and the replicate table was created with the statement create table test1 (col1 int), the following statement would fail: insert into TEST1 (COL1) values (1)

The LTL generated by the Replication Agent must refer to table TEST1 and column COL1 to be accepted by Replication Server. For the Replication Agent, the value of the ltl_character_case configuration parameter must be set to asis to send LTL object names in the same case as they are returned by the primary database.

If no other changes are made, an insert into table test1 with a value of "1" results in the following command sent to the replicate database:

```
insert into TEST1 (COL1) values (1)
```

This statement may fail if the replicate table name is case sensitive and the name of the table is not TEST1 with a column named COL1.

Object name case sensitivity solutions

Ideally, if all the data servers in your replication system are not case sensitive, no special consideration is required.

The following items can be used to help in a situation where some of the servers are case sensitive:

- Use the replicate as clause in replication definitions. When you create a replication definition, either manually, you can specify a separate replicate name for table and column names. Use this feature to specify the case required by your replicate database.

  Note If your primary data is replicated to more than one replicate site, you might have to create a separate replication definition for each unique replicate table name.

- Configure the Replication Agent accordingly. If the option is available, you can configure the Replication Agent to send object names in the same case as the primary database stores them. You can force the case to be anything you like, as long as the LTL generated by the Replication Agent matches the case used in the replication definition.
- Use an alias at the replicate site. As an alternative to creating replication definitions with the replicate as clause, you can create an alias at the replicate database that matches the case in the primary database and points to the desired replicate table name.

PART 3

Implementing Heterogeneous Replication

Chapters in this part describe how to create and maintain a replication system with heterogeneous or non-Sybase data servers, using Sybase replication technology.

- Chapter 6, “Replication System Configuration Examples,” gives examples of typical replication system configurations with heterogeneous or non-Sybase data servers.
- Chapter 7, “Administering Heterogeneous Replication Systems,” describes administration tasks that are specific to replication systems with heterogeneous or non-Sybase data servers.
- Chapter 8, “Troubleshooting Heterogeneous Replication Systems,” describes common problems and troubleshooting procedures for replication systems with heterogeneous or non-Sybase data servers.

CHAPTER 6

Replication System Configuration Examples

This chapter gives examples of several replication system configurations with heterogeneous or non-Sybase data servers and describes the issues involved with each configuration.

Non-Sybase primary to Adaptive Server replicate

The simplest heterogeneous replication scenario is replicating one-way from a non-Sybase primary database to an Adaptive Server replicate database. The only unique requirements are the addition of a Replication Agent designed to extract transaction data from the non-Sybase primary database, and the application of the HDS feature of Replication Server to translate primary database native datatypes to Adaptive Server datatypes.

Replication system components

The following components are required for a non-Sybase primary to Adaptive Server replicate configuration:

- Non-Sybase primary data server (for example, Oracle)
- Replication Agent designed for the primary data server (for example, Replication Agent)
- Replication Server
- Adaptive Server replicate data server

Replication system issues

You must consider the following issues in a non-Sybase primary to Adaptive Server replicate configuration:

- The Replication Server database connection for the primary database must include a valid user ID and password for the primary database, even though this user ID does not apply transactions to the primary database in this configuration.
- If you use a Replication Server heterogeneous datatype support (HDS) datatype definition to define a column's datatype in a replication definition, the *hds_clt_xxx_to_ase.sql* script must be applied to the RSSD of the replicate Replication Server to provide automatic translation of that primary database datatype to an Adaptive Server native datatype.

  If you can use Replication Server native datatypes to represent all the primary column datatypes (that is, no datatype definitions are required), then no HDS translation scripts need be applied.
- Sybase recommends that you do not configure the Replication Agent to perform any datatype translation, unless all replicate databases require the data in that translated form. In that case, it may be a more effective use of resources to let the Replication Agent do the translation once, rather than having replicate Replication Servers do it for each replicate database.

Adaptive Server primary to non-Sybase replicate

A simple heterogeneous replication scenario is replicating one-way from an Adaptive Server primary database to a non-Sybase replicate. The only unique requirements are the addition of a ECDA database gateway to apply transaction data to the replicate database, and the application of the HDS feature of Replication Server to translate Adaptive Server datatypes to native datatypes of the replicate database.

Replication system components

The following components are required for an Adaptive Server primary to non-Sybase replicate configuration:

- Adaptive Server primary database

- Replication Server
- ECDA database gateway designed for the replicate data server (for example, ECDA Option for Oracle
- Non-Sybase replicate data server (for example, Oracle)

Replication system issues

You must consider the following issues in an Adaptive Server primary to non-Sybase replicate configuration:

- The Replication Server database connection for the replicate database must include a valid user ID and password (the Maintenance User) for the replicate database. This user ID must have authority to apply replicate transactions in the replicate database.
- The *hds_xxx_setup_for_replicate.sql* script must be executed against the replicate database to create the rs_info and rs_lastcommit tables in the replicate database.
- The Replication Server replicate database connection must be created referencing the correct function-string class for this particular replicate database.
- The *hds_xxx_funcstrings.sql* script must be applied to the RSSD of the replicate Replication Server.
- The *hds_clt_ase_to_xxx.sql* script must be applied to the RSSD of the replicate Replication Server to provide class-level translation of Adaptive Server native datatypes to replicate database native datatypes.

Note Class-level translations are also required to perform translations on system variable values used by Replication Server to maintain the rs_lastcommit table in the replicate database.

Non-Sybase primary to non-Sybase replicate

This scenario varies in complexity, depending on the mix of non-Sybase data servers. If the primary and replicate databases are the same type such as Oracle to Oracle, fewer class-level translations are required (you do not have to

translate an Oracle DATE datatype to another Oracle DATE datatype). If the primary and replicate databases are different types, you must apply additional class-level translations.

Replication system components

The following components are required for a non-Sybase primary to non-Sybase replicate configuration:

- Non-Sybase primary data server (for example, Oracle)
- Replication Agent designed for the primary data server (for example, Replication Agent)
- Replication Server
- ECDA database gateway designed for the replicate data server (for example, ECDA Option for ODBC)
- Non-Sybase replicate data server (for example, Microsoft SQL Server)

Replication system issues

You must consider the following issues in a non-Sybase primary to non-Sybase replicate configuration:

- The Replication Server primary database connection must include a valid user ID and password for the primary database. This user ID must have authority to apply replicate transactions (even if no transactions will be replicated to the primary database).
- The *hds_xxx_setup_for_replicate.sql* script for the replicate database must be executed against the replicate database to create the rs_info and rs_lastcommit tables in the replicate database.
- The Replication Server replicate database connection must be created referencing the correct function-string class for this particular replicate database (for example, rs_oracle_function_class).
- The *hds_xxx_funcstrings.sql* script must be applied to the RSSD of the replicate Replication Server.
- The *hds_clt_ase_to_xxx.sql* script must be applied to the RSSD of the replicate Replication Server to provide class-level translation of Adaptive Server native datatypes to the replicate database native datatypes.

Even if you are not replicating from an Adaptive Server primary database, Adaptive Server native datatypes are used in the default Replication Server function string calls to maintain the rs_lastcommit table. In addition, replication definitions for a non-Sybase primary database may use Adaptive Server native datatypes.

- If you use a Replication Server HDS datatype definition to define a column's datatype in a replication definition, the *hds_clt_xxx_to_xxx.sql* script must be applied to the RSSD of the replicate Replication Server to provide automatic translation of that primary database datatype to a replicate database native datatype.

  For example, with an Oracle primary database and a Microsoft SQL Server replicate database, you must apply the *hds_clt_oracle_to_msss.sql* script to the RSSD of the replicate Replication Server.

  If you can use existing Replication Server native datatypes to represent all the columns (that is, no datatype definitions are required), then no HDS translation scripts need be applied.

Note If the same type of non-Sybase database is used for both primary and replicate databases (for example, Oracle to Oracle), no additional translation script is required.

Bidirectional non-Sybase to non-Sybase replication

In this scenario, replication occurs both to and from each database. Each non-Sybase database must have both a Replication Agent *and* an ECDA database gateway.

Replication system components

The following components are required for a bidirectional non-Sybase primary to non-Sybase replicate configuration:

- Non-Sybase primary data server (for example, Oracle)
- Replication Agent designed for the primary data server (for example, Replication Agent)

- ECDA database gateway designed for the “primary” data server acting as a replicate database (for example, ECDA Option for Oracle)
- Replication Server
- ECDA database gateway designed for the replicate data server (for example, ECDA Option for Oracle)
- Replication Agent designed for the “replicate” data server acting as a primary database (for example, Replication Agent)
- Non-Sybase replicate data server (for example, Oracle)

Replication system issues

From a technical standpoint, you can set up a bidirectional replication scenario using only two Replication Server database connections (one “primary-and-replicate” connection for each database). However, this document describes four connections, one for each database acting as a primary database, and one for each database acting as a replicate database.

Note In the following description of bidirectional replication issues, the two databases are referred to as Database #1 and Database #2, because both databases take on both “primary” and “replicate” roles in the replication system.

You must consider the following issues in a bidirectional non-Sybase primary to non-Sybase replicate configuration:

- The Replication Server primary database connection for Database #1 must include a valid user ID and password for the primary database. This user ID must be the same user ID specified in the Replication Server replicate database connection for Database #1 (the Maintenance User). This user ID must have authority to apply transaction operations to replicate tables in Database #1.
- The Replication Agent for Database #1 must be configured to bypass Maintenance User transactions to prevent a transaction from returning from the replicate tables in Database #1. Refer to the appropriate Replication Agent *Administration Guide* for details on configuring the Replication Agent to bypass Maintenance User transactions.

- The Replication Server primary database connection for Database #2 must include a valid user ID and password for the primary database. This user ID must be the same user ID specified in the Replication Server replicate database connection for Database #2 (the Maintenance User). This user ID must have authority to apply transaction operations to replicate tables in Database #2.
- The Replication Agent for Database #2 must be configured to bypass Maintenance User transactions to prevent a transaction from returning from the replicate tables in Database #2. Refer to the appropriate Replication Agent *Administration Guide* for details on configuring the Replication Agent to bypass Maintenance User transactions.
- The *hds_xxx_setup_for_replicate.sql* script for each database must be executed against the database to create the rs_info and rs_lastcommit tables that Replication Server requires for a replicate database.
- The Replication Server database connection for replicate Database #1 must be created referencing the correct function-string class designed for this particular database as a replicate database (for example, rs_oracle_function_class.)
- The Replication Server database connection for replicate Database #2 must be created referencing the correct function-string class designed for this particular database as a replicate database.
- The *hds_xxx_funcstrings.sql* script for each database must be applied to the RSSD of the replicate Replication Server for each database.
- The *hds_clt_ase_to_xxx.sql* script must be applied to the RSSD of each replicate Replication Server to provide class-level translation of Adaptive Server native datatypes to replicate database native datatypes.

  Even if you are not replicating from an Adaptive Server primary database, Adaptive Server native datatypes are used in the default Replication Server function-string calls to maintain the rs_lastcommit table. In addition, replication definitions for a non-Sybase primary database may use Adaptive Server native datatypes.
- If you use a Replication Server HDS datatype definition to define a column's datatype in a replication definition (for either Database #1 or Database #2), the *hds_clt_xxx_to_yyy.sql* script must be applied to the RSSD of the replicate Replication Server to provide automatic translation of the primary database datatype to a replicate database native datatype.

For example, you must apply the *hds_clt_msss_to_oracle.sql script* to the RSSD of the replicate Replication Server for Database #1. You must apply the *hds_clt_oracle_to_msss.sql* script to the RSSD of the replicate Replication Server for Database #2.

If you can use existing Replication Server native datatypes to represent all the columns (that is, no datatype definitions are required), then no HDS translation scripts need be applied.

Note No additional translation script is required if the same type of non-Sybase database is used for both primary and replicate databases (for example, Oracle to Oracle).

CHAPTER 7

Administering Heterogeneous Replication Systems

This chapter describes administration tasks for Sybase replication systems with heterogeneous or non-Sybase data servers.

Note This chapter describes only administration tasks that are unique to a Sybase replication system with heterogeneous or non-Sybase data servers. For information about basic replication system administration, see the Replication Server *Administration Guide*.

Replication system maintenance

This section describes the tasks required to maintain these components of a replication system:

- Primary data servers and Replication Agents
- Replication Server
- Replicate data servers and ECDA database gateways

Primary data servers and Replication Agents

In a replication system, the typical ongoing maintenance requirement at a primary database is monitoring and adjusting transaction log resources. All Replication Agents use the native transaction log of the database. An increase in transaction volume can increase the demand for log resources.

You should review unusual events for their impact on the transaction log. A one-time or periodic maintenance operation, which can impact a large number of database rows, can have a severe impact on transaction log and Replication Agent performance, if only temporarily.

Replication Server

Replication Server may suffer a minor performance impact when it has to process class-level translations, which provide automatic datatype translation for a replicate database.

If you remove a non-Sybase database from a Replication Server (that is, *all* connections associated with that database are removed), you can remove the class-level translations supporting that database if there are no other databases of the same type that are managed by the Replication Server. For example, you can remove an Oracle replicate database if there are no other Oracle databases managed by that Replication Server.

At a primary Replication Server, you can remove the translations for a specific database type only if no primary database connection exists for that database type and no replication definitions use a map to clause referencing a user-defined datatype for that database type.

Note Use caution when removing class-level translations. You must verify that for a particular Replication Server, *no* database connection for that database type exists at that Replication Server.

Replicate data servers and ECDA gateways

ECDA database gateways do not require any special maintenance; however, your resource or capacity planning at the replicate database site should include an estimate of the quantity of replicate transactions expected to be received by the replicate database from all primary databases to which the replicate subscribes.

CHAPTER 8

Troubleshooting Heterogeneous Replication Systems

This chapter describes common problems and troubleshooting procedures for Sybase replication systems with heterogeneous or non-Sybase data servers.

Troubleshooting overview

Common Replication Server troubleshooting tasks, such as dumping stable queues, debugging failures with the Data Server Interface (DSI) and Replication Server Interface (RSI), and diagnosing and correcting problems with subscriptions, are described in the Replication Server *Troubleshooting Guide*.

For non-Sybase primary and replicate databases, the Replication Agent and ECDA gateway documentation provide troubleshooting information for each specific database.

This chapter describes some basic troubleshooting if replication fails in a Sybase replication system with heterogeneous or non-Sybase data servers.

Inbound and outbound queue problems

The *inbound queue* is where Replication Server stores the data it receives from a primary database (through a Replication Agent or another Replication Server). The *outbound queue* is where Replication Server stores the data it needs to send to a replicate site (either a replicate database or another Replication Server).

This section describes how to troubleshoot problems with these Replication Server queues.

Inbound queue problems

You can tell that the Replication Server inbound queue for a primary database is not being updated if you issue the Replication Server admin who,sqm command at the primary Replication Server and the results indicate the following:

- The number of blocks being written in the Replication Server inbound queue for the connection in question is not changing.
- The number of duplicate messages being detected is not increasing.

❖ **To determine the reason the inbound queue is not being updated**

1 Verify the Replication Server connection Replication Agent User thread status.

You can issue an admin who command in the primary Replication Server to review the status of the Replication Agent User thread for the Replication Server database connection in question.

- If there is no Replication Agent User thread for the connection, the connection was not created with the with log transfer on clause. You can alter the Replication Server database connection to turn log transfer processing on, if needed.
- If the Replication Agent User thread status is down, the Replication Agent is not actively connected to the Replication Server. A down status is typical for Replication Agents that only connect to Replication Server when there is work to be sent, and then disconnect after a period of inactivity.

2 Verify that the expected Replication Agent is executing.

Verify that the expected Replication Agent is active, and that the values of the Replication Agent rs_source_ds and rs_source_db configuration parameters match the desired Replication Server connection name.

Refer to the appropriate Replication Agent documentation for other tests to validate that the Replication Agent is executing.

3 Verify that the expected table or procedure is marked for replication.

Replication Agent documentation describes the Replication Agent commands you can use to check replication status.

Replication Agent provides for separate *enabling* of replication, in addition to marking. In this case, make sure the marked object is also enabled for replication.

4 Verify that the Replication Agent is scanning new records.

If the database object is marked for replication, the log scanning process of the Replication Agent should record that additional information is being scanned.

To verify that new records are being scanned:

1 Start tracing in the Replication Agent.

2 Update or execute a primary database object that has been marked for replication.

3 Verify that scanning occurs.

Refer to the appropriate Replication Agent documentation to determine the trace flags you can use to validate the scanning process.

Outbound queue problems

You can tell that the Replication Server outbound queue for a replicate database is not being updated if you issue the Replication Server admin who,sqm command at the replicate Replication Server and the results indicate the following:

- The number of blocks being written in the Replication Server outbound queue for the connection in question is not changing.
- The number of duplicate messages being detected is not increasing.

Problems between inbound and outbound queues are often naming problems.

❖ **To determine the reason the outbound queue is not being updated**

1 Verify that any Replication Server routes are active.

Refer to the Replication Server *Troubleshooting Guide* for route validation techniques between primary and replicate Replication Servers.

2 Verify that the Replication Server connection DSI thread is not down.

Issue an admin who command in the replicate Replication Server to review the status of the DSI thread for the Replication Server connection.

If the DSI thread status is down, the Replication Server is not connected to the replicate database (or ECDA gateway). Review the Replication Server log for errors and attempt to resume the connection.

3 Verify that the DSI thread connection is not in "Loss Detected" mode by viewing the replicate Replication Server log for "Loss Detected" messages for the DSI thread in question.

When Replication Server detects a loss, no further messages are accepted on the DSI thread connection.

Refer to the Replication Server *Administration Guide* for information about recovering from this error.

4 Verify the primary replication definition.

The primary Replication Server inbound queue can receive data, but when it cannot apply that data to any replication definition, the reason is that the name of the replication definition does not match the name presented in the Log Transfer Language (LTL) that was created by the Replication Agent. This becomes more likely when you are using different non-Sybase database types with different default character cases.

Replication Server processing of replication commands is case sensitive. In a replication system with non-Sybase data servers, you must ensure that the LTL generated by Replication Agents matches the Replication Server connection names and replication definition object names.

Some Replication Agents always use lowercase names when they communicate with Replication Server (for example, Adaptive Server and DB2 Universal Database). However, the best option is to pick one character case (uppercase or lowercase) and use it consistently with all Replication Server connections, replication definitions, and subscription names.

The character case validation process is manual. You can use the rs_helprep command to verify the name of a replication definition. Then, you can then turn on LTL tracing in the Replication Agent and verify that the name provided in the LTL trace matches the spelling and character case of the name specified in the replication definition.

If the character case appears to be incorrect, review the Replication Agent documentation to verify the default character case settings and any possible configuration changes. If a name is misspelled, you must delete and then re-create the replication definition.

Replication failure problems

Following is a "Replicate database failure to be updated" problem in a replication system with heterogeneous or non-Sybase data servers. This section describes how to troubleshoot this problem.

Replicate database is not updated

If the Replication Server outbound queue is being updated but transaction data is not being applied at the replicate database, use the following procedure to determine the reason.

❖ **To determine why replicate transactions are not applied at the replicate database**

1 Determine if the subscription contains a where clause.

Verify that the transaction data expected will pass any where clause in the subscription definition. Use the rs_helpsub stored procedure to list the text of the subscription.

2 Verify HDS installation.

If you are using Replication Server HDS to support replication to or from a non-Sybase data server, verify that the HDS scripts have been properly applied.

See "Expected datatype translations do not occur" on page 156 for more information.

3 Verify that the rs_lastcommit table is set up correctly.

If you are using Replication Server HDS to support replication to or from a non-Sybase data server, verify that the HDS scripts have been properly applied.

Refer to "Updates to rs_lastcommit fail" on page 155 for more detail.

4 Review the replicate Replication Server log for errors.

5 Review the replicate database log for errors.

6 Verify manual access to replicate objects.

Log in to the replicate database (or ECDA gateway) using the Replication Server connection Maintenance User ID, and verify that you have update authority to the replicate table or procedure.

7 Validate commands sent to the replicate database:

- Turn on the DSI_BUF_DUMP trace in the replicate Replication Server and record to the Replication Server log the commands being sent to the replicate database.
- Verify that these commands, when manually applied, produce the expected results.

Note The DSI_BUF_DUMP trace flag can be used with any Replication Server. (By contrast, the similar DSI_CMD_DUMP trace flag is available only with the diagnostic version of Replication Server.) Refer to the Replication Server *Troubleshooting Guide* for more information about the Replication Server traces.

8 Turn on tracing at the ECDA gateway to see what commands are being received.

For example, these parameters in ECDA Option for Oracle configuration file cause ECDA to write additional information to the *DCO.log* file:

- network_tracing = 1
- traces = 1,2,3,4,5,6,10

Refer to the appropriate ECDA documentation for specific trace availability and syntax.

Troubleshooting specific errors

This section describes troubleshooting for specific errors you may encounter in a Sybase replication system with heterogeneous or non-Sybase data servers.

Date information does not include time values

This problem can occur with non-Sybase primary databases. Replication Agents have configuration parameters that must be set to request that the native date or time datatypes in the primary database be converted to Sybase datetime formats for replication. If this conversion is not requested, the default date format returned from the primary database is used, which may not provide a time value. For example, the Oracle default date display format is *DD-MON-YY*. If the Replication Agent is not configured to convert this display to a format that includes time (for example, *CCYY-MM-DD hh.mm.ss.mmm*), then the time value will be all zeroes whenever the date is replicated.

See the Replication Agent for DB2 UDB *User's and Troubleshooting Guide* for more information about the Date_in_char parameter. For more information about the pdb_convert_datetime parameter, see the Replication Agent *Administration Guide*.

When using a Replication Server datatype for a non-Sybase data server, you must ensure that the date datatype specified in a replication definition map to clause includes a time value.

Note The Replication Agent pdb_convert_datetime configuration parameter must be set *before* a table or procedure is marked for replication in a DB2 UDB, Microsoft SQL Server, or Oracle database. The Replication Agent uses that parameter value to determine how to configure the translation of the data from that specific table or procedure.

Updates to rs_lastcommit fail

When replicating into a non-Sybase replicate database, the replicate Replication Server updates the rs_lastcommit table as soon as the connection is resumed. If the replicate Replication Server error log displays a syntax error while updating the rs_lastcommit table, the following procedure may help identify the problem.

❖ **To troubleshoot rs_lastcommit update failure**

1 Verify that the table exists in the replicate database and that the schema matches the script in the *hds_xxx_setup_for_replicate.sql* file.

Note For a non-Sybase database, you should create the rs_lastcommit table by executing the *hds_xxx_setup_for_replicate.sql* script supplied with Replication Server.

2 Verify access authority.

Log in to the replicate database using the Replication Server Maintenance User ID and password specified in the create connection command for that database connection.

Verify that this user ID can update the rs_lastcommit table. You should be able to insert and delete a dummy entry without error.

3 Trace the actual command.

Turn on tracing in the replicate Replication Server (DSI_BUF_DUMP trace) or in the ECDA gateway and resume the Replication Server connection.

Identify the failing statement and correct as necessary.

Expected datatype translations do not occur

The most common reason for a datatype translation failure is an incomplete installation of the necessary user-defined datatypes (UDDs) and translations. You can use the following procedure to validate your UDD and translation installation.

❖ **To validate UDD and translation installation**

1 Reboot the Replication Servers. Replication Server caches all function-string information at start-up.

Subsequent changes to the function strings stored in the RSSD do not take effect until the Replication Server is restarted.

2 Verify that UDDs have been defined.

The Replication Server script *hds_xxx_udds.sql* provides the SQL statements necessary to apply UDDs to the RSSD for specific non-Sybase replicate databases.

For each datatype, the script issues a delete followed by an insert. You can re-run these scripts without failure.

These scripts must be executed at each Replication Server where:

- The non-Sybase data server is a replicate database, and
- The non-Sybase data server primary replication definition refers to a UDD using the map_to clause.

Verify that your copy of the scripts has been updated with the correct use statement for the database name of the RSSD.

3 Verify that class-level translations have been applied to the replicate Replication Server.

The Replication Server script *hds_clt_from_to_xxx.sql* provides the SQL statements necessary to apply class-level translations to the RSSD of the replicate Replication Server for a specific combination of non-Sybase primary databases to non-Sybase replicate databases.

Note The *hds_clt_ase_to_xxx.sql* script is required for any non-Sybase replicate database. For example, if you are replicating from ASE to Oracle, the *hds_clt_ase_to_oracle.sql* translations must be applied to ensure Replication Server updates to the rs_lastcommit table are properly translated and applied to the replicate database.

You can re-run these scripts without failure. Verify that your copy of the scripts has been updated with the correct use statement for the database name of the RSSD.

Note Class-level translations adversely affect performance. Do not install translations that are not required for your replication system.

---

4 Verify that your replicate database Replication Server connection is associated with the appropriate function-string class.

To take advantage of class-level translations, the replicate Replication Server connection must use the correct non-Sybase function-string class.

You can use the Replication Server rs_helpdb command to determine which function-string class is defined for a database connection.

Function-string classes for replicate databases are:

- Adaptive Server Enterprise – *rs_sqlserver_function_class*

- DB2 Universal Database on IBM z/OS platforms – *rs_db2_function_class*
- DB2 Universal Database on UNIX and Windows platforms – *rs_udb_function_class*
- Microsoft SQL Server – *rs_msss_function_class*
- Oracle – *rs_oracle_function_class*

You can use the Replication Server admin show_function_classes command to display a list of active function-string classes.

You can use the Replication Server alter connection command to change the function-string class of an existing database connection.

5 Verify that the non-Sybase function-string classes have been updated with appropriate function strings.

Replication Server script *hds_xxx_funcstrings.sql* provides the SQL statements necessary to apply function strings to the RSSD of the replicate Replication Server for a specific non-Sybase replicate database.

For each function string, the scripts issue a delete followed by an insert. You can re-run these scripts without failure.

Verify that your copy of the scripts has been updated with the correct use statement for the database name of the RSSD.

6 Use the Replication Server admin translate command.

The admin translate command allows you to verify the results of a specific translation. Use this command to verify that the translation engine is providing the translation results you expect.

For more information on heterogeneous datatype support (HDS) and the admin translate command, refer to the Replication Server *Administration Guide*.

LTL generation and tracing

This section describes how to trace the Log Transfer Language (LTL) commands sent to a primary Replication Server, as well as other significant Replication Agent traces.

Replication Agent for DB2 UDB

You can use the configuration parameters described in this section to obtain additional information that is not normally presented by Replication Agent for DB2 UDB.

Note There is usually some performance impact when you use these parameters. You should review the full description of a parameter in the Replication Agent for DB2 UDB *Installation Guide* before using it.

- If you need additional tracing to help debug the information passed to a Replication Agent user exit, you can set the value of the API_com_test configuration parameter to Y. You can also use this trace when no exit is being used.
- The LTL_test_only configuration parameter controls whether LTM for MVS connects to Replication Server and sends transaction operations for replication. When the value of the LTL_test_only parameter is Y, LTL that would normally be sent to Replication Server is written to the *LTLOUT* file instead.

  Note No replication takes place when the value of the LTL_test_only parameter is Y.

- The Suppress_col_names configuration parameter determines whether LTM for MVS suppresses column names from the LTL that is sent to Replication Server.

  If you are tracing LTL output, setting the value of this parameter to N ensures that column names are present in the generated LTL.

- The trace=1,4 configuration parameter traces Sybase Log Extract calls to the Replication API and sends the output to *SYSPRINT*. Set the value of the trace=1,4 parameter to Y to turn on this trace.
- The trace=1,11 configuration parameter writes LTL that is passed to Replication Server to *LTLOUT*. Set the value of the trace=1,11 parameter to Y to turn on this trace.

  Note The trace=1,11 trace option continues to send LTL to the primary Replication Server, whereas the LTL_test_only option does not.

- The Use_repdef configuration parameter allows LTM for MVS to send LTL to Replication Server that contains only the columns specified in the replication definition.

  Setting the value of the Use_repdef parameter to N may increase the amount of information provided in an LTL trace.

Replication Agent

You can use the trace flags and configuration parameters described in this section to obtain additional information that is not normally presented by the Replication Agent (for Microsoft SQL Server, Oracle, and UDB).

Note Some performance impact usually occurs when you use these trace flags and parameters. Before using it, you should review the full description of a trace flag or parameter in the Replication Agent *Administration Guide*.

Trace flags

Normal trace output is sent to the Replication Agent instance log file. However, output from the LTITRACELTL trace point is sent to a separate LTL output log file (*LTITRACELTL.log*).

The following trace flags are particularly useful for troubleshooting Replication Agent problems:

- LRTRACE – traces general execution of the Log Reader component.
- LTITRACE – traces general execution of the Log Transfer Interface component.
- LTITRACELTL – enables LTL statement tracing in the *LTITRACELTL.log* file.
- RACONTRC – traces connection and query execution.
- RACONTRCSQL – traces SQL statements sent to the primary database.

Configuration parameters

The settings of the following Replication Agent configuration parameters affect the trace information:

- compress_ltl_syntax – provides more verbose description of LTL commands when set to false.
- connect_to_rs – allows LTL to be generated without actual connection or sending information to Replication Server when set to false.
- log_trace_verbose – provides more verbose description of traced components when set to true.

- use_rssd – provides a complete generation of LTL commands without modification for replication definition information when set to false.
- column_compression – sends complete column information (all columns in after images) in the generated LTL for update operations when set to false.

For a complete description of Replication Agent trace flags and configuration settings, refer to the Replication Agent *Administration Guide*.

PART 4

Appendixes

Appendixes in this part provide supplemental information that can help you design and implement a replication system with heterogeneous or non-Sybase data servers.

- Appendix A, “Datatype Translation and Mapping,” lists the class-level datatype translations for non-Sybase data servers supported by Replication Server, and the Replication Server datatype names for non-Sybase datatypes.
- Appendix B, “Materialization Issues,” describes the issues related materializing subscriptions in a replication system with heterogeneous or non-Sybase data servers.
- Appendix C, “Heterogeneous Database Reconciliation,” describes the issues related to synchronizing and reconciling databases in a replication system with heterogeneous or non-Sybase data servers.
- Appendix D, “Replication with SQL Anywhere,” describes primary and replicate data server issues for Adaptive Server Anywhere in a Sybase replication system.

APPENDIX A

Datatype Translation and Mapping

This appendix lists the class-level datatype translations for non-Sybase data servers supported by Replication Server, and the Replication Server datatype names for non-Sybase datatypes.

For each supported non-Sybase data server, Replication Server provides class-level translations that define the default mapping from one datatype to another. Translations are provided for:

- Non-Sybase datatypes that do not correspond directly to Adaptive Server datatypes
- Adaptive Server datatypes that do not correspond directly to the non-Sybase datatypes
- Non-Sybase datatypes that do not correspond directly to the datatypes of another supported non-Sybase data server

Note Class-level translations are *not* provided for any datatype that corresponds directly to a datatype in another data server.

DB2 datatypes

This section lists class-level translations (default datatype mapping) for DB2 datatypes and Replication Server datatype names for DB2 datatypes.

Note This information about datatype translation applies to DB2 Universal Database (UDB) in either mainframe environments (such as IBM z/OS), or UNIX and Microsoft Windows environments.

DB2 class-level translations

The following sections list the class-level translations for Adaptive Server datatypes to DB2 datatypes, DB2 datatypes to Adaptive Server datatypes, and DB2 datatypes to datatypes of other supported non-Sybase data servers:

- Adaptive Server to DB2 datatypes
- DB2 to Adaptive Server datatypes
- DB2 to Microsoft SQL Server datatypes
- DB2 to Oracle datatypes

Adaptive Server to DB2 datatypes

Table A-1 lists class-level translations from Adaptive Server datatypes to DB2 datatypes.

Table A-1: Class-level translation from Adaptive Server to DB2 datatypes

Adaptive Server datatype	DB2 datatype
bigint	BIGINT
binary	CHAR FOR BIT DATA
bit	TINYINT
date	DATE (UNIX and Windows only)
datetime	TIMESTAMP
decimal	DECIMAL
int	NUMERIC
money	NUMERIC
numeric	NUMERIC
real	REAL (UNIX and Windows only)
smalldatetime	TIMESTAMP
smallint	NUMERIC

Adaptive Server datatype	DB2 datatype
smallmoney	NUMERIC
time	TIME (UNIX and Windows only)
tinyint	NUMERIC
unsigned bigint	DECIMAL (20,0)
unsigned int	BIGINT
unsigned smallint	INTEGER
unsigned tinyint	SMALLINT
unitext	DBCLOB
varbinary	VARCHAR FOR BIT DATA

DB2 to Adaptive Server datatypes

Table A-2 lists class-level translations from DB2 datatypes to Adaptive Server datatypes.

Table A-2: Class-level translation from DB2 to Adaptive Server datatypes

DB2 datatype	Adaptive Server datatype
CHAR FOR BIT DATA	binary
DATE	datetime
DOUBLE (UNIX and Windows only)	float
REAL (UNIX and Windows only)	real
TIME	datetime
TIMESTAMP	datetime
VARCHAR FOR BIT DATA	varbinary

DB2 to Microsoft SQL Server datatypes

Table A-3 lists class-level translations from DB2 datatypes to Microsoft SQL Server datatypes.

Table A-3: Class-level translation from DB2 to Microsoft SQL Server datatypes

DB2 datatype	Microsoft SQL Server datatype
CHAR FOR BIT DATA	binary
DATE	datetime
DOUBLE (UNIX and Windows only)	float
REAL (UNIX and Windows only)	real
TIME	datetime
TIMESTAMP	datetime
VARCHAR FOR BIT DATA	varbinary

DB2 to Oracle datatypes

Table A-4 lists class-level translations from DB2 datatypes to Oracle datatypes.

Table A-4: Class-level translation from DB2 to Oracle datatypes

DB2 datatype	Oracle datatype
CHAR FOR BIT DATA	RAW
DATE	DATE
DOUBLE (UNIX and Windows only)	FLOAT
REAL (UNIX and Windows only)	REAL
TIME	DATE (with time)
TIMESTAMP	DATE (with time)
VARCHAR FOR BIT DATA	RAW

Replication Server datatype names for DB2

Table A-5 lists the Replication Server user-defined datatype (UDD) names that identify DB2 datatypes for DB2 data servers on z/OS platforms.

Table A-5: Replication Server names for DB2 z/OS datatypes

DB2 z/OS datatype	Replication Server name
CHAR FOR BIT DATA	rs_db2_char_for_bit
DATE	rs_db2_date
DECIMAL	rs_db2_decimal, rs_db2_numeric
TIME	rs_db2_time
TIMESTAMP	rs_db2_timestamp
VARCHAR FOR BIT DATA	rs_db2_varchar_for_bit

Table A-6 lists the Replication Server user-defined datatype (UDD) names that identify DB2 datatypes for DB2 data servers on UNIX and Microsoft Windows platforms.

Table A-6: Replication Server names for DB2 UNIX and Windows datatypes

DB2 UNIX and Windows datatypes	Replication Server name
CHAR FOR BIT DATA	rs_udb_char_for_bit
DATE	rs_udb_date
DOUBLE	rs_udb_double
INTEGER	rs_udb_bigint
REAL	rs_udb_real
TIME	rs_udb_time
TIMESTAMP	rs_udb_timestamp
VARCHAR FOR BIT DATA	rs_udb_varchar_for_bit

Microsoft SQL Server datatypes

This section lists class-level translations (default datatype mapping) for Microsoft SQL Server datatypes and Replication Server datatype names for Microsoft SQL Server datatypes.

Microsoft SQL Server class-level translations

The following sections list class-level translations for Microsoft SQL Server datatypes to datatypes of other supported non-Sybase data servers:

- Microsoft SQL Server to DB2 datatypes
- Microsoft SQL Server to Oracle datatypes

Adaptive Server to Microsoft SQL Server datatypes

Table A-1 lists class-level translations from Adaptive Server datatypes to Microsoft SQL Server datatypes for the unsigned datatypes.

The remaining class-level translations are not supplied for Adaptive Server datatypes to Microsoft SQL Server datatypes (or Microsoft SQL Server datatypes to Adaptive Server datatypes) because Microsoft SQL Server datatypes are directly compatible with Adaptive Server datatypes and they require no translation.

Table A-7: Class-level translation from Adaptive Server to Microsoft SQL Server datatypes

Adaptive Server datatype	Microsoft SQL Server datatype
unsigned bigint	DECIMAL (20,0)
unsigned int	BIGINT
unsigned smallint	INT
unsigned tinyint	SMALLINT
unitext	NTEXT

Microsoft SQL Server to DB2 datatypes

Table A-8 lists class-level translations from Microsoft SQL Server datatypes to DB2 datatypes.

Table A-8: Class-level translation from Microsoft SQL Server to DB2 datatypes

Microsoft SQL Server datatype	DB2 datatype
binary	CHAR FOR BIT DATA
bit	TINYINT
datetime	TIMESTAMP
decimal	DECIMAL
money	NUMERIC
numeric	NUMERIC
smalldatetime	TIMESTAMP
smallmoney	NUMERIC
varbinary	VARCHAR FOR BIT DATA

Microsoft SQL Server to Oracle datatypes

Table A-9 lists class-level translations from Microsoft SQL Server datatypes to Oracle datatypes.

Table A-9: Class-level translation from Microsoft SQL Server to Oracle datatypes

Microsoft SQL Server datatype	Oracle datatype
binary	RAW
datetime	DATE (with time)
money	DECIMAL
smalldatetime	DATE
smallmoney	DECIMAL
varbinary	RAW

Replication Server datatype names for Microsoft SQL Server

All Microsoft SQL Server datatypes are compatible with the corresponding Adaptive Server datatypes. Only one Microsoft SQL Server datatype has a user-defined datatype definition.

Table A-10 lists the Replication Server user-defined datatype (UDD) name that identifies a Microsoft SQL Server datatype.

Table A-10: Replication Server names for Microsoft SQL Server datatypes

Microsoft SQL Server datatype	Replication Server name
integer	rs_msss_bigint

Oracle datatypes

This section lists class-level translations (default datatype mapping) for Oracle datatypes and Replication Server datatype names for Oracle datatypes.

Oracle class-level translations

The following sections list class-level translations for Adaptive Server datatypes to Oracle datatypes, Oracle datatypes to Adaptive Server datatypes, and Oracle datatypes to datatypes of other supported non-Sybase data servers:

- Adaptive Server to Oracle datatypes
- Oracle to Adaptive Server Datatypes
- Oracle to DB2 datatypes
- Oracle to Microsoft SQL Server datatypes

Adaptive Server to Oracle datatypes

Table A-11 lists class-level translations from Adaptive Server datatypes to Oracle datatypes.

Table A-11: Class-level translation from Adaptive Server to Oracle datatypes

Adaptive Server datatype	Oracle datatype
bigint	NUMBER
binary	RAW
date	DATE
datetime	DATE (with time)
money	DECIMAL
smalldatetime	DATE
smallmoney	DECIMAL
time	DATE (with time)
unsigned tinyint	SMALLINT
unsigned smallint	INTEGER
unsigned int	NUMBER
unsigned bigint	NUMBER
unitext	NCLOB
varbinary	RAW

Oracle to Adaptive Server Datatypes

Table A-12 lists class-level translations from Oracle datatypes to Adaptive Server datatypes.

Table A-12: Class-level translation from Oracle to Adaptive Server datatypes

Oracle datatype	Adaptive Server datatype
RAW	varbinary
DATE	datetime
TIMESTAMP (0)	datetime
TIMESTAMP (1)	datetime
TIMESTAMP (2)	datetime
TIMESTAMP (3)	datetime
TIMESTAMP (4)	datetime
TIMESTAMP (5)	datetime
TIMESTAMP (6)	datetime
TIMESTAMP (7)	datetime
TIMESTAMP (8)	datetime
TIMESTAMP (9)	datetime

Oracle to DB2 datatypes

Table A-13 lists class-level translations from Oracle datatypes to DB2 datatypes.

Table A-13: Class-level translation from Oracle to DB2 datatypes

Oracle datatype	DB2 datatype
RAW	CHAR FOR BIT DATA
DATE	DATE
DATE (with time)	TIMESTAMP
FLOAT	DOUBLE (UNIX and Windows only)
INTEGER	INTEGER (UNIX and Windows only)
TIMESTAMP (0)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (1)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (2)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (3)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (4)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (5)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (6)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (7)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (8)	TIMESTAMP (UNIX and Windows only)
TIMESTAMP (9)	TIMESTAMP (UNIX and Windows only)

Oracle to Microsoft SQL Server datatypes

Table A-14 lists class-level translations from Oracle datatypes to Microsoft SQL Server datatypes.

Table A-14: Class-level translation from Oracle to Microsoft SQL Server datatypes

Oracle datatype	Microsoft SQL Server datatype
RAW	varbinary
DATE	datetime
TIMESTAMP (0)	datetime
TIMESTAMP (1)	datetime
TIMESTAMP (2)	datetime
TIMESTAMP (3)	datetime
TIMESTAMP (4)	datetime
TIMESTAMP (5)	datetime
TIMESTAMP (6)	datetime
TIMESTAMP (7)	datetime
TIMESTAMP (8)	datetime
TIMESTAMP (9)	datetime

Replication Server datatype names for Oracle

Table A-15 lists the Replication Server user-defined datatype (UDD) names that identify Oracle datatypes.

Table A-15: Replication Server names for Oracle datatypes

Oracle datatype	Replication Server name
RAW	rs_oracle_binary
DATE	rs_oracle_datetime
ROWID	rs_oracle_rowid
INTEGER	rs_oracle_int
INTERVAL	rs_oracle_interval
BINARY_FLOAT	rs_oracle_float
NUMBER	rs_oracle_decimal
TIMESTAMP(n)	rs_oracle_timestamp9
TIMESTAMP(n) (with local time zone)	rs_oracle_timestamptz
UDD object type	rs_rs_char_raw

APPENDIX B

Materialization Issues

This appendix describes the subscription materialization issues that you must consider when implementing a replication system with heterogeneous or non-Sybase data servers. It also describes how to materialize subscriptions to primary tables in a non-Sybase database.

Materialization overview

Materialization is a process of creating and activating subscriptions and copying data from a primary database to a replicate database, thereby initializing the replicate database.

Before you can replicate data from a primary database, you must set up and populate each replicate database so that the replicate objects (such as tables) are in a state consistent with those in the primary database.

Types of materialization

Two types of subscription materialization are supported by Replication Server:

- *Bulk materialization* – manually creating and activating a subscription and populating a replicate database using data unload and load utilities outside the control of the replication system.
- *Automatic materialization* – creating a subscription and populating a replicate database using Replication Server commands.

For more information about subscription materialization methods, see the Replication Server *Administration Guide*.

Heterogeneous materialization issues

To materialize subscriptions to primary data in a non-Sybase data server, you may use a bulk materialization or automatic materialization, if it applies. With bulk materialization methods, you must coordinate and manually perform the following activities:

- Define, activate, and validate the subscription (or create the subscription without materialization).
- Unload the subscription data at the primary database.
- Move the unloaded data to the replicate database site.
- Load the primary data into the replicate database tables.
- Resume the database connection from the replicate Replication Server to the replicate data server so that the replicate database can receive replicated transactions.
- Resume replication at the Replication Agent instance.

Bulk materialization options

There are two bulk materialization options for subscriptions to primary data in a non-Sybase database:

- *Atomic bulk materialization*
  - Stop updates to the primary table and dump the subscription data from the primary database.
  - In the replicate Replication Server, define the subscription.
  - In the primary database, use the rs_marker function to activate the subscription using the with suspension option. See the Replication Server *Reference Manual* for how to apply the rs_marker function.
  - Load the subscription data into the replicate table.
  - Resume the database connection from the replicate Replication Server to the replicate database.
  - In the replicate Replication Server, validate the subscription.
- *Nonatomic bulk materialization*
  - In the replicate Replication Server, use the set autocorrection command.
  - In the replicate Replication Server, define the subscription.
  - In the primary database, use the rs_marker stored procedure to activate the subscription using the with suspension option.
  - Dump the subscription data from the primary database.

- In the primary database, use the rs_marker stored procedure to validate the subscription.
- Load the subscription data into the replicate table.
- Resume the database connection from the replicate Replication Server to the replicate database.
- When the subscription becomes valid at all Replication Servers, turn off autocorrection.

Unloading data from a primary database

Part of the subscription materialization process involves unloading the subscription data from the primary table so it can be loaded into the replicate table. *Subscription data* is the data in the primary table that is requested by the subscription.

Data unloading utilities are usually provided with data server software. You can use one of the OEM-supplied data unloading utilities or a database unload utility of your choice.

Note Once subscription data is unloaded from a primary database, you may need to perform datatype translation on the unloaded data before loading the data into the replicate database. For more information, see "Datatype translation issues" on page 177.

Datatype translation issues

Note If you are not using the unload utility and are using automatic materialization, then Replication Server performs the translations.

If you use the heterogeneous datatype support (HDS) feature of Replication Server to perform either column- or class-level translations on replicated data, you must perform datatype translations on the subscription data you unload from the primary database for materialization:

- If any column-level translation is specified in a replication definition for the data, you must perform the datatype translation specified in each replication definition for the primary data.
- If class-level translations are specified for the database connection to a replicate database (in the function-string class for the database connection), you must perform the datatype translations specified for each replicate database connection.

Note When Replication Server processes data for replication, column-level translations are performed before class-level translations. Therefore, if you must perform datatype translations on subscription data based on both column-level translation and class-level translations, you must perform the column-level translation first.

Datatype definitions are specified in scripts provided with Replication Server. These scripts define the attributes of the non-Sybase database native datatypes in terms of Replication Server native datatypes. You can use the datatype definitions in these scripts to determine how to translate the datatypes in the subscription data that you unload from the primary database.

Following are the scripts that support non-Sybase data servers:

- *hds_db2_udds.sql* – DB2 Universal Database on IBM z/OS platforms
- *hds_udb_udds.sql* – DB2 Universal Database on UNIX and Windows platforms
- *hds_msss_udds.sql* – Microsoft SQL Server
- *hds_oracle_udds.sql* – Oracle

Class-level translations identify primary database native datatypes and the replicate database native datatypes that data should be translated into. (For example, Oracle DATE should be translated to DB2 TIMESTAMP.)

A set of scripts is provided with Replication Server for each combination of supported primary and replicate data servers. For example, the following scripts are provided for DB2 UDB as a replicate database:

- *hds_clt_ase_to_db2.sql* – translations for Adaptive Server datatypes to DB2 datatypes.
- *hds_clt_msss_to_db2.sql* – translations for Microsoft SQL Server datatypes to DB2 datatypes.

- *hds_clt_oracle_to_db2.sql* – translations for Oracle datatypes to DB2 datatypes.

You can use the datatype translations specified in these scripts to determine how to translate the datatypes in the subscription data that you unload from the primary database.

Note Datatype translations based on class-level translation scripts must be performed *after* column-level translations.

For more information about Replication Server HDS datatype translations, see "Replication Server heterogeneous datatype support" on page 121.

Loading data into replicate databases

Part of the subscription materialization process involves loading the subscription data from the primary table into the replicate table.

Note After subscription data is unloaded from a primary database, you may need to perform datatype translation on the unloaded data before loading the data into the replicate database.

If you are using Adaptive Server Enterprise as the data server for the replicate database, you can use the Sybase bcp utility to load subscription data into the replicate database.

If you are using a non-Sybase data server as the data server for the replicate database, you can use the load utility of your choice to load subscription data into the replicate database.

For more information about using bcp with Adaptive Server, see Sybase Adaptive Server Enterprise *Utility Guide*.

Atomic bulk materialization

Atomic bulk materialization assumes that all applications updating the primary table can be suspended while a copy of the table is made. The copy is then loaded at the replicate site.

You can use this method to retrieve data from the primary database if you can (at least temporarily) suspend updates to the primary data.

Preparing for materialization

Before you start an atomic bulk materialization procedure, verify the following:

- The primary table exists and contains data.
- You have access to a user ID with ownership or select privilege on the primary table (or a column to be replicated in the primary table).
- The replicate table exists and contains the appropriate column(s) with the appropriate datatype(s).
- You have successfully configured all Replication Servers in your replication system.
- You have created the replication definition correctly at the primary Replication Server.
- If you are using Replication Agent for a DB2 Universal Database, Microsoft SQL Server, or Oracle primary database:
  - You have successfully initialized the Replication Agent which also creates some objects in the primary database.
  - You have marked and enabled replication for the primary table in the primary database.
  - You have started the Replication Agent instance and put it in the *Replicating* state.

Performing atomic bulk materialization

❖ **To perform atomic bulk materialization**

1 Use isql to log in to the replicate Replication Server as the System Administrator (sa):

```
isql -Usa -Psa_password -SRRS_servername
```

where *sa* is the System Administrator user ID, *sa_password* is the password for the System Administrator user ID, and *RRS_servername* is the server name of the replicate Replication Server.

2 Define the subscription at the replicate Replication Server:

```
1> define subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> [where search_conditions]
5> go
```

The dataserver.database must match the Replication Server connection name you use for the replicate database.

3 Check the subscription at both the primary and replicate Replication Servers. Use the following command to verify that the subscription status is DEFINED:

```
1> check subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

4 Lock the primary table to prevent primary transaction activity. This prevents updates to the primary table during materialization.

5 Unload the subscription data at the primary site using your site's preferred database unload method to select or dump the data from the primary table.

Note When unloading subscription data from the primary table, make sure you select only the columns specified in the replication definition and the rows specified in the subscription.

6 Perform any datatype translations necessary for the subscription data.

If any column-level translation is specified in the replication definition for this data, you must perform the datatype translation specified in the replication definition.

If class-level translations are specified for the subscription, you must perform the datatype translations specified for the subscription.

7 Activate the subscription using the with suspension option at the replicate Replication Server:

```
1> activate subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> with suspension
 5> go
```

8 Wait for the subscription to become active at both the primary and replicate Replication Servers. Execute the check subscription command at both the primary and replicate Replication Servers to verify that the subscription status is ACTIVE.

When the subscription status is ACTIVE at the replicate Replication Server, the database connection for the replicate database is suspended.

9 Restore the primary table to read-write access (unlock).

10 Load the subscription data into the replicate database using the bcp utility or your site's preferred database load utility.

11 From the replicate Replication Server, resume the database connection for the replicate database:

```
1> resume connection
2> to dataserver.database
3> go
```

12 Validate the subscription at the replicate Replication Server:

```
1> validate subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

13 Wait for the subscription to become valid at both the primary and replicate Replication Servers, then execute the check subscription command at both the primary and replicate Replication Servers to verify that the status is VALID.

When you complete this procedure, the subscription is created, the replicate data is consistent with the primary data, and replication is in progress.

If replication is not in progress when you complete this procedure, see Chapter 8, "Troubleshooting Heterogeneous Replication Systems."

See also

- Replication Server *Reference Manual* for information on Replication Command Language (RCL) commands.
- Replication Server *Administration Guide* for information on configuring Replication Servers and materialization methods.

Nonatomic bulk materialization

Nonatomic bulk materialization assumes applications updating the primary table cannot be suspended while a copy of the table is made. Therefore, nonatomic materialization requires the use of the Replication Server *autocorrection* feature to get the replicate database synchronized with the primary database.

Note You cannot use nonatomic materialization if the replicate minimal columns feature is set for the replication definition for the primary table.

Preparing for materialization

Before you start a nonatomic bulk materialization procedure, verify the following:

- The primary table exists and contains data.
- You have access to a user ID with ownership or select privilege on the primary table (or a column to be replicated in the primary table).
- The replicate table exists and contains the appropriate columns.
- You have successfully configured all Replication Servers in your replication system.
- You created the replication definition correctly at the primary Replication Server.
- If you are using Replication Agent for a DB2 Universal Database, Microsoft SQL Server, or Oracle primary database:
  - You have successfully initialized the Replication Agent which also creates some objects in the primary database.

- You have marked and enabled replication for the primary table in the primary database.
- You have started the Replication Agent instance and put it in the *Replicating* state.

Performing nonatomic bulk materialization

❖ **To perform nonatomic bulk materialization**

1 Use isql to log in to the replicate Replication Server as the System Administrator (sa):

```
isql -Usa -Psa_password -SRRS_servername
```

where:

- *sa* is the System Administrator user ID.
- *sa_password* is the password for the System Administrator user ID.
- *RRS_servername* is the server name of the replicate Replication Server.

2 Turn on the autocorrection feature at the replicate Replication Server:

```
1> set autocorrection on
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

3 Define the subscription using the with suspension option at the replicate Replication Server:

```
1> define subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> with suspension
5> go
```

The dataserver.database must match the Replication Server connection name you use for the replicate database.

4 In the primary database, invoke the rs_marker stored procedure to activate the subscription.

5 Check the subscription at both the primary and replicate Replication Servers. Use the following command to verify that the subscription status is ACTIVE:

```
1> check subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

When the subscription status is ACTIVE at the replicate Replication Server, the database connection for the replicate database is suspended.

6 Unload the subscription data at the primary site using your site's preferred database unload method to select or dump the data from the primary tables.

Note When unloading subscription data from the primary table, make sure you select only the columns specified in the replication definition and the rows specified in the subscription.

---

7 Perform any datatype translations necessary for the subscription data.

If any column-level translation is specified in the replication definition for this data, you must perform the datatype translation specified in the replication definition.

If class-level translations are specified for the subscription, you must perform the datatype translations specified for the subscription.

8 In the primary database, invoke the rs_marker stored procedure to validate the subscription.

9 Wait for the subscription to become valid at both the primary and replicate Replication Servers, then execute the check subscription command at both the primary and replicate Replication Servers to verify that the status is VALID.

10 Load the subscription data into the replicate database using the bcp utility or your site's preferred database load utility.

11 From the replicate Replication Server, resume the database connection for the replicate database:

```
1> resume connection
2> to dataserver.database
3> go
```

12 Wait for the subscription to become valid at both the primary and replicate Replication Servers, then execute the check subscription command at both the primary and replicate Replication Servers to verify that the status is VALID.

When the subscription's status is VALID at the replicate Replication Server, the replicate database is synchronized with the primary database and you can turn off autocorrection.

13 Turn off the autocorrection feature at the replicate Replication Server:

```
1> set autocorrection off
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

When you complete this procedure, the subscription is created, the replicate data is consistent with the primary data, and replication is in progress.

If replication is not in progress when you complete this procedure, see Chapter 8, "Troubleshooting Heterogeneous Replication Systems."

See also

- Replication Server *Reference Manual* for information on Replication Command Language (RCL) commands.
- Replication Server *Administration Guide* for information on configuring Replication Servers and materialization methods.

APPENDIX C

Heterogeneous Database Reconciliation

This appendix describes the issues involved with comparing and reconciling data from different databases in a heterogeneous replication system.

Sybase rs_subcmp utility

The Sybase rs_subcmp utility allows you to compare primary and replicate tables in Adaptive Server databases, and reconcile any differences found. Sybase provides the rs_subcmp executable program with Replication Server.

Some other database vendors may provide a similar "compare" utility that can perform the same function for their own databases, but there is no equivalent utility to support different types of non-Sybase data servers (for example, comparing tables in an Oracle database to tables in a Microsoft SQL Server database).

For non-Sybase database support, the only options are:

- Acquire third-party tools that provide such functionality, or
- Build your own application.

Database comparison application

You can develop a custom application to perform the same functions as the rs_subcmp utility. The application's complexity depends on the number of different data server types, the complexity of the tables to be compared, the amount of data translation involved, and so forth.

The following list describes the major issues that a database comparison application must accommodate to be successful in a heterogeneous replication environment:

- Connectivity – the application must be able to communicate with both the primary and replicate databases. If multiple database vendors are involved, ODBC and JDBC protocols can provide a common interface and functionality.
- Sort order – the default sort order may be different for different databases. The application may need to force the sort order to improve comparison performance.
- Character sets – some primary and replicate databases may store character data in different character sets. Your custom application may need to support these translations.
- Object identification – primary and replicate tables may not be named the same, or have exactly the same schema or column names. The comparison application may need to accept very explicit instructions for location, database, and table and column names to be referenced.
- Subset comparison – the application may need to compare only a portion of a table. The ability to specify a where clause type of select for both primary and replicate tables may be important.
- Latency – in a replication system, there is always some latency (a measure of the time it takes a primary transaction to appear in a replicate table). A comparison application must include some tolerance to distinguish between rows that are "not there" and "not there yet."
- Data transformation – the application must be able to handle difference in precision and format between different databases, the same way Replication Server supports class-level translations. An option to simplify the application may be to allow certain columns to be excluded from the comparison process, based on datatype (for example, do not compare the DATE datatypes of different database vendors).
- Large object (LOB) data – Large object (for example, LOB, CLOB, TEXT, or IMAGE) datatypes cause additional processing issues because of their

size. To improve performance, you can limit the number of bytes used for comparison, if the likelihood of a “non-match” can still be relied on.

For more information about the rs_subcmp utility, see the Replication Server *Administration Guide* and the Replication Server *Reference Manual*.

APPENDIX D

Replication with SQL Anywhere

This appendix describes the issues involved with using SQL Anywhere in a Replication Server system.

SQL Anywhere primary data servers

This section describes the primary database issues and considerations specific to the SQL Anywhere data server in a Sybase replication system.

SQL Anywhere is a different relational database than Sybase Adaptive Server Enterprise, which is the original primary data server supported by Replication Server. Adaptive Server Enterprise is designed specifically for high-performance OLTP (online transaction processing) and mixed workload enterprise computing. By contrast, SQL Anywhere is designed for the following applications:

- Embedded database

  Many applications (personal information managers, document management systems—nearly any application that stores information) require a database "behind the scenes." SQL Anywhere is intended to be the database for these applications. The UltraLite deployment option is intended for embedded environments that have limited resources.

- Mobile computing

  With its SQL Remote replication, SQL Anywhere extends transaction-based computing throughout the enterprise. The UltraLite

deployment option and MobiLink synchronization technology provide full database functionality on devices with limited resources.

- Workgroup server

  Workgroups ranging in size from a few people to a few hundred people in an organization may require a data server that they can share. SQL Anywhere is a multiuser server that can provide a high-performance database for workgroups, well-suited for (but not limited to) environments where administration and hardware resources are limited.

SQL Anywhere Replication Agent

As a primary data server in a replication system, SQL Anywhere interacts with the Replication Agent. The SQL Anywhere Replication Agent is responsible for identifying and transferring transactions from the SQL Anywhere primary database to a primary Replication Server.

SQL Anywhere primary database permissions

A script file is provided with SQL Anywhere to set up an SQL Anywhere database to function as a primary database. The *rssetup.sql* script file performs the following operations to set up SQL Anywhere as a primary database for replication:

- Creates a user ID named dbmaint, with password dbmaint, with DBA permissions. This is the Replication Server Maintenance User ID and password required to connect to the primary database.
- Creates a user named sa, with password sysadmin, with DBA permissions. This is the user ID used by Replication Server when materializing data.
- Adds sa and dbmaint to a group named rs_systabgroup.

Replication intrusions and impacts in SQL Anywhere

Setting REPLICATE ON for an SQL Anywhere table places additional information in the database transaction log whenever an insert, update, or delete operation occurs on the table. The additional information significantly increases the log resources used by the server. The SQL Anywhere Replication Agent uses this information to submit the full *before image* of the row, where required, to Replication Server for replication.

Primary database connectivity for SQL Anywhere

By default, the SQL Anywhere Replication Agent uses the sa user ID and password created by the *rssetup.sql* command file to log in to the SQL Anywhere primary database.

You must create the user ID and password that the SQL Anywhere Replication Agent uses to log in to the primary Replication Server. Record these values in the rs_user and rs_pw configuration parameters in the SQL Anywhere Replication Agent configuration file. This user ID must have been defined and granted connect source permission in the primary Replication Server.

A Replication Server database connection name is made up of two parts: a data server name (server_name) and a database name (db_name). Record these values in the RS_source_ds and RS_source_db configuration parameters in the SQL Anywhere Replication Agent configuration file.

The primary Replication Server does not use the source_db portion of the Replication Server database connection. Instead, Replication Server obtains the database name from the command line of the data server identified in the source_ds portion of the Replication Server database connection. However, you must include a database name in the Replication Server create connection statement to conform to the syntax.

Primary database limitations in SQL Anywhere

One SQL Anywhere Replication Agent is required for each SQL Anywhere primary database from which transactions are replicated.

The SQL Anywhere Replication Agent does not “pre-read” data from the RSSD of the primary Replication Server, as other Replication Agents do.

You cannot substitute an SQL Anywhere Replication Agent for an Adaptive Server Enterprise Replication Agent, because the SQL Anywhere and Adaptive Server Enterprise transaction logs have different formats.

SQL Anywhere primary database configuration issues

In SQL Anywhere, all database object identifiers are not case sensitive (that is, uppercase and lowercase are treated as the same). In Adaptive Server Enterprise, database object identifiers are case sensitive by default. With SQL Anywhere, you must ensure that the case of database object identifiers matches in all parts of the SQL statements to ensure compatibility with Adaptive Server Enterprise.

Replication definitions for primary tables in SQL Anywhere

Because the SQL Anywhere Replication Agent does not "pre-read" data from the RSSD of the primary Replication Server, the entire row for all replicated operations is sent to the primary Replication Server. By contrast, other Replication Agents can selectively send only those columns identified by the replication definition.

SQL Anywhere primary datatype translation issues

SQL Anywhere supports data of zero length that is not NULL. However, non-null long, varchar, and long binary data of zero length are replicated to a replicate site as NULL.

If a primary table has columns with unsupported datatypes, you can replicate the data if you create a replication definition using a compatible supported datatype. For example, to replicate a double column, you could define the column as float in the replication definition.

SQL Anywhere system management issues

Replication Server Manager support for SQL Anywhere is not implemented in version 15.1.

Other primary database issues for SQL Anywhere

One of the differences between the Adaptive Server Enterprise Replication Agent and the SQL Anywhere Replication Agent is that while the Adaptive Server Enterprise RepAgent thread depends on a temporary recovery database for access to old transactions, the SQL Anywhere Replication Agent depends on access to old transaction logs. No temporary recovery database exists for the SQL Anywhere Replication Agent.

SQL Anywhere replicate data servers

This section describes the replicate database issues and considerations specific to the SQL Anywhere data server in a Sybase replication system.

SQL Anywhere is a different relational database from Sybase Adaptive Server Enterprise, which is the original replicate data server supported by Replication Server.

As a replicate data server in a replication system, SQL Anywhere interacts directly with the replicate Replication Server. The replicate Replication Server logs in to the SQL Anywhere replicate database and applies replicated transactions.

SQL Anywhere replicate database permissions

The rssetup script file is provided with SQL Anywhere to set up an SQL Anywhere database to function as a replicate database. It performs the following:

- Creates a user ID named dbmaint, with password dbmaint, with DBA permissions. This is the Replication Server Maintenance User ID and password required to connect to the primary database.
- Creates a user named sa, with password sysadmin, with DBA permissions. This is the user ID used by Replication Server when materializing data.
- Adds sa and dbmaint to a group named rs_systabgroup.
- Creates the system tables and stored procedures necessary in the replicate database to support Replication Server.

Replication intrusions and impacts in SQL Anywhere

The only significant intrusions or impacts to the SQL Anywhere replicate database are the database objects created by the *rssetup.sql* script to support Replication Server replicate database operations.

The *rssetup.sql* script creates two tables in the replicate database to support Replication Server operations:

- rs_info – contains information about the sort order and character set used by the replicate database. You should confirm that the insert statements for this table set the proper character set and sort order for your replicate database.
- rs_lastcommit – holds information about the last successfully committed replicate transaction applied to the replicate database.
- rs_ticket _history_table– supports Replication Server rs_ticket processing by placing an rs_ticket marker in the Primary database transaction log. This was created in support of the Replication Server rs_ticket feature.

Replicate database connectivity for SQL Anywhere

You do not need to use a database gateway when you use SQL Anywhere as a replicate data server; the replicate Replication Server connects directly to the SQL Anywhere replicate data server.

A Replication Server database connection name is made up of two parts: a data server name (server_name) and a database name (db_name). The replicate Replication Server looks for an interfaces file entry for the replicate database server_name specified in the database connection.

Replication Server logs in to the replicate data server using the user_name and password specified in the database connection. For SQL Anywhere replicate databases, the user_name and password should be the dbmaint user ID and password that were created when the *rssetup.sql* script was executed.

The replicate Replication Server does not use the source_db portion of the Replication Server database connection. Instead, Replication Server obtains the database name from the command line of the data server identified in the source_ds portion of the Replication Server database connection. However, you must include a database name in the Replication Server create connection statement to conform to the syntax.

You also must make an entry in the Replication Server interfaces file to identify the host and port where the SQL Anywhere replicate data server is listening. The interfaces file entry name must match the server_name portion of the Replication Server database connection.

SQL Anywhere replicate datatype translation issues

SQL Anywhere databases do not require Heterogeneous Datatype Support (HDS) support.

Glossary

active database — In a warm standby application, a database that is replicated to a standby database. See also **warm standby application**.

Adaptive Server — The Sybase version 11.5 and later relational database server.

applied function — A replicated function, associated with a function replication definition, that Replication Server delivers from a primary database to a subscribing replicate database. The function passes parameter values to a stored procedure that is executed at the replicate database. See also **replicated function delivery**, **request function**, and **function replication definition**.

article — A replication definition extension for tables or stored procedures that can be an element of a publication. Articles may or may not contain where clauses, which specify a subset of rows that the replicate database receives.

asynchronous procedure delivery — A method of replicating, from a source to a destination database, a stored procedure that is associated with a table replication definition.

asynchronous command — A command that a client submits when the client is not prevented from proceeding with other operations before the completion status is received. Many Replication Server commands function as asynchronous commands within the replication system.

atomic materialization — A materialization method that copies subscription data from a primary to a replicate database through the network in a single atomic operation, using a select operation with a holdlock. No changes to primary data are allowed until data transfer is complete. Replicate data may be applied either as a single transaction or in increments of ten rows per transaction, which ensures that the replicate database transaction log does not fill. Atomic materialization is the default method for the create subscription command. See also **nonatomic materialization**, **bulk materialization** and **no materialization**.

autocorrection — A setting applied to replication definitions, using the set autocorrection command, to prevent failures caused by missing or duplicate rows in a copy of a replicated table. When autocorrection is enabled, Replication Server converts each update or insert operation into a delete followed by

an insert. Autocorrection should *only* be enabled for replication definitions whose subscriptions use nonatomic materialization.

base class — A function-string class that does not inherit function strings from a parent class. See also **function-string class**.

bitmap subscription — A type of subscription that replicates rows based on bitmap comparisons. Create columns using the int datatype, and identify them as the rs_address datatype when you create a replication definition. When you create a subscription, compare each rs_address column to a bitmask using a bitmap comparison operator (&) in the where clause. Rows matching the subscription's bitmap are replicated.

bulk materialization — A materialization method whereby subscription data in a replicate database is initialized outside of the replication system. For example, data may be transferred from a primary database using media such as magnetic tape, diskette, CD-ROM, or optical storage disk. Bulk materialization involves a series of commands, starting with define subscription. You can use bulk materialization for subscriptions to table replication definitions or function replication definitions. See also **atomic materialization**, **nonatomic materialization**, and **no materialization**.

centralized database system — A database system where data is managed by a single database management system at a centralized location.

class — See **error class** and **function-string class**.

class tree — A set of function-string classes, consisting of two or more levels of derived and parent classes that derive from the same base class. See also **function-string class**.

client — A program connected to a server in a client/server architecture. It may be a front-end application program executed by a user or a utility program that executes as an extension of the system.

Client/Server Interfaces (C/SI) — The Sybase interface standard for programs executing in a client/server architecture.

concurrency — The ability of multiple clients to share data or resources. Concurrency in a database management system depends upon the system protecting clients from conflicts that arise when data in use by one client is modified by another client.

connection — A connection from a Replication Server to a database. See also **Data Server Interface (DSI)** and **logical connection**.

coordinated dump — A set of database dumps or transaction dumps that is synchronized across multiple sites by distributing an rs_dumpdb or rs_dumptran function through the replication system.

database — A set of related data tables and other objects that is organized and presented to serve a specific purpose.

database generation number — Stored in both the database and the RSSD of the Replication Server that manages the database, the first part of the origin queue ID (QID) of each log record. The origin queue ID ensures that the Replication Server does not process duplicate records. During recovery operations, you may need to increment the database generation number so that Replication Server does not ignore records submitted after the database is reloaded.

database replication definition — A description of a set of database objects—tables, transactions, functions, system stored procedures, and DDL—for which a subscription can be created.

You can also create table replication definitions and function replication definitions. See also **table replication definition** and **function replication definition**.

database server — A server program, such as Sybase Adaptive Server, that provides database management services to clients.

data definition language (DDL) — The set of commands in a query language, such as Transact-SQL, that describes data and their relationships in a database. DDL commands in Transact-SQL include those using the create, drop, and alter keywords.

data manipulation language (DML) — The set of commands in a query language, such as Transact-SQL, that operates on data. DML commands in Transact-SQL include select, insert, update, and delete.

data server — A server whose client interface conforms to the Sybase Client/Server Interfaces and provides the functionality necessary to maintain the physical representation of a replicated table in a database. Data servers are usually database servers, but they can also be any data repository with the interface and functionality Replication Server requires.

Data Server Interface (DSI) — Replication Server threads corresponding to a connection between a Replication Server and a database. DSI threads submit transactions from the DSI outbound queue to a replicate data server. They consist of a scheduler thread and one or more executor threads. The scheduler thread groups the transactions by commit order and dispatches them to the executor threads. The executor threads map functions to function strings and execute the transactions in the replicate database. DSI threads use an Open Client connection to a database. See also **outbound queue** and **connection**.

data source A specific combination of a database management system (DBMS) product such as a relational or non-relational data server, a database residing in that DBMS, and the communications method used to access that DBMS from other parts of a replication system. See also **database** and **data server**.

decision support application A database client application characterized by ad hoc queries, reports, and calculations and few data update transactions.

declared datatype The datatype of the value delivered to the Replication Server from the Replication Agent:

- If the Replication Agent delivers a base Replication Server datatype (such as datetime) to the Replication Server, the declared datatype is the base datatype.
- Otherwise, the declared datatype must be the UDD for the original datatype at the primary database.

default function string The function string that is provided by default for the system-provided classes rs_sqlserver_function_class and rs_default_function_class and classes that inherit function strings from these classes, either directly or indirectly. See also **function string**.

dematerialization The optional process when a subscription is dropped, whereby specific rows that are not used by other subscriptions are removed from the replicate database.

derived class A function-string class that inherits function strings from a parent class. See also **function-string class** and **parent class**.

direct route A route used to send messages directly from a source to a destination Replication Server, with no intermediate Replication Servers. See also **indirect route** and **route**.

disk partition See **partition**.

distributed database system A database system where data is stored in multiple databases on a network. The databases may be managed by data servers of the same type (for example, Adaptive Server) or by heterogeneous data servers.

Distributor A Replication Server thread (DIST) that helps to determine the destination of each transaction in the inbound queue.

dump marker A message written by Adaptive Server in a database transaction log when a dump is performed. In a warm standby application, when you are initializing the standby database with data from the active database, you can specify Replication Server to use the dump marker to determine where in the

transaction stream to begin applying transactions in the standby database. See also **warm standby application**.

Embedded Replication Server System Database (ERSSD)

The SQL Anywhere database that stores Replication Server system tables. You can choose whether to store Replication Server system tables on the ERSSD or the Adaptive Server RSSD. See also **Replication Server System Database (RSSD)**.

Enterprise Connect Data Access (ECDA)

An integrated set of software applications and connectivity tools that allow access to data within a heterogeneous database environment, such as a variety of LAN-based, non-Sybase data sources, and mainframe data sources.

error action

A Replication Server response to a data server error. Possible Replication Server error actions are ignore, warn, retry_log, log, retry_stop, and stop_replication. Error actions are assigned to specific data server errors.

error class

A name for a collection of data server error actions that are used with a specified database.

exceptions log

A set of three Replication Server system tables that holds information about transactions that failed on a data server. The transactions in the log must be resolved by a user or by an intelligent application. You can use the rs_helpexception stored procedure to query the exceptions log.

Failover

Sybase Failover allows you to configure two version 12.0 and later Adaptive Servers as companions. If the primary companion fails, that server's devices, databases, and connections can be taken over by the secondary companion.

For more detailed information about how Sybase Failover works in Adaptive Server, refer to *Using Sybase Failover in a High Availability System*, which is part of the Adaptive Server Enterprise documentation set.

For instructions on how to enable failover support for non-RSSD Replication Server connections to Adaptive Server, see "Configuring the Replication System to Support Sybase Failover" in Chapter 16, "Replication System Recovery," of the Replication Server *Administration Guide*.

fault tolerance

The ability of a system to continue to operate correctly even though one or more of its component parts is malfunctioning.

function

A Replication Server object that represents a data server operation such as insert, delete, select, or begin transaction. Replication Server distributes such operations to other Replication Servers as functions. Each function consists of a function name and a set of data parameters. In order to execute the function in a destination database, Replication Server uses function strings to convert a

function to a command or set of commands for a type of database. See also **user-defined function**, and **replicated function delivery**.

function replication definition A description of a replicated function used in replicated function delivery. The function replication definition, maintained by Replication Server, includes information about the parameters to be replicated and the location of the primary version of the affected data. See also **replicated function delivery**.

function scope The range of a function's effect. Functions have replication definition scope or function-string class scope. A function with replication definition scope is defined for a specific replication definition and cannot be applied to other replication definitions. A function with function-string class scope is defined once for a function-string class and is available only within that class.

function string A string that Replication Server uses to map a database command to a data server API. For the rs_select and rs_select_with_lock functions only, the string contains an input template, used to match function strings with the database command. For all functions, the string also contains an output template, used to format the database command for the destination data server.

function-string class A named collection of function strings used with a specified database connection. Function-string classes include those provided with Replication Server and those you have created. Function-string classes can share function string definitions through function-string inheritance. The three system-provided function-string classes are rs_sqlserver_function_class, rs_default_function_class, and rs_db2_function_class. See also **base class**, **class tree**, **derived class**, **function-string inheritance**, and **parent class**.

function-string inheritance The ability to share function string definitions between classes, whereby a derived class inherits function strings from a parent class. See also **derived class**, **function-string class**, and **parent class**.

function-string variable An identifier used in a function string to represent a value that is to be substituted at runtime. Variables in function strings are enclosed in question marks (?). They represent column values, function parameters, system-defined variables, or user-defined variables.

function subscription A subscription to a function replication definition (used in applied function delivery).

gateway Connectivity software that allows two or more computer systems with different network architectures to communicate.

generation number See **database generation number**.

heterogeneous data servers Data servers that are supplied by more than one vendor used together in a distributed database system.

high availability (HA) Very low downtime. Computer systems that provide HA usually provide 99.999% availability, or roughly five minutes unscheduled downtime per year.

hibernation mode A Replication Server state in which all DDL commands, except admin and sysadmin commands, are rejected; all routes and connections are suspended; most service threads, such as DSI and RSI, are suspended; and RSI and RepAgent (or LTM) users are logged off and not allowed to log on. Used during route upgrades and may be turned on for a Replication Server to debug problems.

hot standby application A database application in which the standby database can be placed into service without interrupting client applications and without losing any transactions. See also **warm standby application**.

ID Server One Replication Server in a replication system is the ID Server. In addition to performing the usual Replication Server tasks, the ID Server assigns unique ID numbers to every Replication Server and database in the replication system, and maintains version information for the replication system.

inbound queue A stable queue used to spool messages from a Replication Agent to a Replication Server.

indirect route A route used to send messages from a source to a destination Replication Server, through one or more intermediate Replication Servers. See also **direct route** and **route**.

interfaces file A file containing entries that define network access information for server programs in a Sybase client/server architecture. Server programs may include Adaptive Servers, SQL Servers, gateways, Replication Servers, and Replication Agents such as LTM for SQL Server. The interfaces file entries enable clients and servers to connect to each other in a network.

latency The measure of the time it takes to distribute to a replicate database a data modification operation first applied in a primary database. The time includes Replication Agent processing, Replication Server processing, and network overhead.

local-area network (LAN) A system of computers and devices, such as printers and terminals, connected by cabling for the purpose of sharing data and devices.

locator value The value stored in the rs_locater table of the Replication Server's RSSD that identifies the latest log transaction record received and acknowledged by the Replication Server from each previous site during replication.

logical connection A database connection that Replication Server maps to the connections for the active and standby databases in a warm standby application. See also **connection** and **warm standby application**.

login name The name that a user or a system component such as Replication Server uses to log in to a data server, Replication Server, or Replication Agent.

Log Transfer Language (LTL) A subset of the Replication Command Language (RCL). A Replication Agent such as RepAgent or LTM for SQL Server uses LTL commands to submit to Replication Server the information it retrieves from primary database transaction logs.

Log Transfer Manager (LTM) The Replication Agent program for Sybase SQL Server. See also **Replication Agent** and **RepAgent thread**.

Maintenance User A data server login name that Replication Server uses to maintain replicate data. In most applications, Maintenance User transactions are not replicated.

materialization The process of copying data specified by a subscription from a primary database to a replicate database, thereby initializing the replicate table. Replicate data can be transferred over a network, or, for subscriptions involving large amounts of data, loaded initially from media. See also **atomic materialization**, **bulk materialization**, **no materialization**, and **nonatomic materialization**.

materialization queue A stable queue used to spool messages related to a subscription being materialized or dematerialized.

missing row A row missing from a replicated copy of a table but present in the primary table.

mixed-version system A replication system containing Replication Servers of different software versions that have different capabilities based on their different software versions and site versions. Mixed-version support is available only if the system version is 11.0.2 or greater.

For example, a replication system containing Replication Servers version 11.5 or later and version 11.0.2 is a mixed-version system. A replication system containing Replication Servers of releases earlier than release 11.0.2 is not a mixed-version system, because any newer Replication Servers are restricted by the system version from using certain new features. See also **site version** and **system version**.

more columns Columns in a replication definition exceeding 250 but limited to 1024. More columns are supported by Replication Server version 12.5 and later.

multi-site availability (MSA) Methodology for replicating database objects—tables, functions, transactions, system stored procedures, and DDL from the primary to the replicate database. See also **database replication definition**.

name space The scope within which an object name must be unique.

nonatomic materialization A materialization method that copies subscription data from a primary to a replicate database through the network in a single operation, without a holdlock. Changes to the primary table are allowed during data transfer, which may cause temporary inconsistencies between replicate and primary databases. Data is applied in increments of ten rows per transaction, which ensures that the replicate database transaction log does not fill. Nonatomic materialization is an optional method for the create subscription command. See also **autocorrection**, **atomic materialization**, **no materialization**, and **bulk materialization**.

network-based security Secure transmission of data across a network. Replication Server supports third-party security mechanisms that provide user authentication, unified login, and secure message transmission between Replication Servers.

no materialization A materialization method that lets you create a subscription when the subscription data already exists at the replicate site. Use the create subscription command with the without materialization clause. You can use this method to create subscriptions to table replication definitions and function replication definitions. See also **atomic materialization** and **bulk materialization**.

online transaction processing (OLTP) application A database client application characterized by frequent transactions involving data modification (inserts, deletes, and updates).

origin queue ID (QID) Formed by the Replication Agent, the qid uniquely identifies each log record passed to the Replication Server. It includes the date and time from the primary data server, and the database generation number. See also **database generation number**.

orphaned row A row in a replicated copy of a table that does not match an active subscription.

outbound queue A stable queue used to spool messages. The DSI outbound queue spools messages to a replicate database. The RSI outbound queue spools messages to a replicate Replication Server.

parallel DSI A method of configuring a database connection so that transactions are applied to a replicate data server using multiple DSI threads operating in parallel, rather than a single DSI thread. See also **connection** and **Data Server Interface (DSI)**.

parameter An identifier representing a value that is provided when a procedure executes. Parameter names are prefixed with an @ character in function strings. When a procedure is called from a function string, Replication Server passes the parameter values, unaltered, to the data server. See also **searchable parameter**.

parent class A function-string class from which a derived class inherits function strings. See also **function-string class** and **derived class**.

partition A raw disk partition or operating system file that Replication Server uses for stable queue storage. Use operating system files only in a test environment.

physical connection See **connection**.

primary data The definitive version of a set of data in a replication system. The primary data is maintained on a data server that is known to all of the Replication Servers with subscriptions for the data.

primary database Any database that contains data that is replicated to another database through the replication system.

primary fragment A horizontal segment of a table that holds the primary version of a set of rows.

primary key A set of table columns that uniquely identifies each row.

primary site A Replication Server where a function-string class or error class is defined. See **error class** and **function-string class**.

principal user The user who starts an application. When using network-based security, Replication Server logs in to remote servers as the principal user.

projection A vertical slice of a table that represents a subset of the table's columns.

publication A group of articles from the same primary database. A publication lets you collect replication definitions for related tables and/or stored procedures and then subscribe to them as a group. You collect replication definitions as articles in a publication at the source Replication Server and subscribe to them with a publication subscription at the destination Replication Server. See also **article** and **publication subscription**.

publication subscription A subscription to a publication. See also **article** and **publication**.

published datatype The datatype of the column after the column-level translation (and before a class-level translation, if any) at the replicate data server. The published datatype must be either a Replication Server base datatype or a UDD for the datatype in the target data server. If the published datatype is omitted from the replication definition, it defaults to the declared datatype

query — In a database management system, a request to retrieve data that meets a given set of criteria. The SQL database language includes the select command for queries.

quiescent — A replication system in which all data-changing operations (or transactions) have been propagated to their destinations. Some Replication Server commands or procedures require that you first quiesce the replication system.

remote procedure call (RPC) — A request to execute a procedure that resides in a remote server. The server that executes the procedure could be a Adaptive Server, a Replication Server, or a server created using Open Server. The request can originate from any of these servers or from a client application. The RPC request format is a part of the Sybase Client/Server Interfaces.

RepAgent thread — The Replication Agent for Adaptive Server Enterprise. The RepAgent thread is an Adaptive Server thread. It sends transaction log information from the primary database to the primary Replication Server.

Rep Agent User thread — The thread on a Replication Server database connection that a Replication Agent connects with, on behalf of a primary database. See also **Data Server Interface (DSI)**.

replicate database — Any database that contains data that is replicated from another database through the replication system.

replicated function delivery — A method of replicating, from a source to a destination database, a stored procedure that is associated with a function replication definition. See also **applied function**, **request function**, and **function replication definition**.

replicated stored procedure — An Adaptive Server stored procedure that is marked as replicated using the sp_setrepproc or the sp_setreplicate system procedure. Replicated stored procedures can be associated with function replication definitions or table replication definitions. See also **replicated function delivery** and **asynchronous procedure delivery**.

replicated table — A table that is maintained by Replication Server, in part or in whole, in databases at multiple locations. There is one primary version of the table, which is marked as replicated using the sp_setreptable or the sp_setreplicate system procedure; all other versions are replicated copies.

Replication Agent — A program or module that sends transaction log information from a primary data server to a primary Replication Server. The Replication Agent for Adaptive Server Enterprise is the RepAgent thread. Sybase provides separate Replication Agent software products to support non-Sybase data servers in a replication system.

Replication Command Language (RCL)
The commands used to manage information in Replication Server.

replication definition
Usually, a description of a table 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 version of the table.

You can also create function replication definitions; sometimes the term "table replication definition" is used to distinguish between table and function replication definitions. See also **function replication definition**.

Replication Server
The Sybase server program that maintains replicated data, typically on a LAN, and processes data transactions received from other Replication Servers on the same LAN or on a WAN.

Replication Server Interface (RSI)
A thread that logs in to a destination Replication Server and transfers commands from the RSI outbound stable queue to the destination Replication Server. There is one RSI thread for each destination Replication Server that is a recipient of commands from a primary or intermediate Replication Server. See also **outbound queue** and **route**.

Replication Server Manager (RSM)
An application for managing and monitoring a replication system and its components. Includes status and monitoring functions, diagnostics and troubleshooting functions, asynchronous notification of user-defined events, and operational control over many aspects of the system.

Replication System Administrator
The System Administrator who manages routine operations in the Replication Server.

Replication Server System Database (RSSD)
The Adaptive Server database containing a Replication Server system tables. You can choose whether to store Replication Server system tables on the RSSD or the Adaptive Server Anywhere (ASA) ERSSD. See also **Embedded Replication Server System Database (ERSSD)**.

Replication Server system Adaptive Server
The Adaptive Server with the database containing a Replication Server's system tables (the RSSD).

replication system
A data processing system where data is replicated in multiple databases to provide remote users with the benefits of local data access. Specifically, a replication system that is based upon Replication Server and includes other components such as Replication Agents and data servers.

replication system domain
All replication system components that use the same ID Server.

request function A replicated function, associated with a function replication definition, that Replication Server delivers from a replicate database to a primary database. The function passes parameter values to a stored procedure that is executed at the primary database. See also **replicated function delivery**, **request function**, and **function replication definition**.

route A one-way message stream from a source Replication Server to a destination Replication Server. Routes carry data modification commands (including those for RSSDs) and replicated functions or stored procedures between Replication Servers. See also **direct route** and **indirect route**.

route version The lower of the site version numbers of the route's source and destination Replication Servers. Replication Server version 11.5 and later use the route version number to determine which data to send to the replicate site. See also **site version**.

row migration The process whereby column value changes in rows in a primary version of a table cause corresponding rows in a replicate version of the table to be inserted or deleted, based on comparison with values in a subscription's where clause.

SQL Server The Sybase relational database pre-11.5 server.

schema The structure of the database. DDL commands and system procedures change system tables stored in the database. Supported DDL commands and system procedures can be replicated to standby databases when you use Replication Server version 11.5 or later and Adaptive Server version 11.5 or later.

searchable column A column in a replicated table that can be specified in the where clause of a subscription or article to restrict the rows replicated at a site.

searchable parameter A parameter in a replicated stored procedure that can be specified in the where clause of a subscription to help determine whether or not the stored procedure should be replicated. See also **parameter**.

secondary truncation point See **truncation point**.

site An installation consisting of, at minimum, a Replication Server, data server, and database, and possibly a Replication Agent, usually at a discrete geographic location. The components at each site are connected over a WAN to those at other sites in a replication system. See also **primary site**.

site version The version number for an individual Replication Server. Once the site version has been set to a particular level, the Replication Server enables features specific to that level, and downgrades are not allowed. See also **software version**, **route version**, and **system version**.

software version The version number of the software release for an individual Replication Server. See also **site version** and **system version**.

Stable Queue Manager (SQM) A thread that manages the stable queues. There is one Stable Queue Manager (SQM) thread for each stable queue accessed by the Replication Server, whether inbound or outbound.

Stable Queue Transaction (SQT) interface A thread that reassembles transaction commands in commit order. A Stable Queue Transaction (SQT) interface thread reads from inbound stable queues, puts transactions in commit order, then sends them to the Distributor (DIST) thread or a DSI thread, depending on which thread required the SQT ordering of the transaction.

stable queues Store-and-forward queues where Replication Server stores messages destined for a route or database connection. Messages written into a stable queue remain there until they can be delivered to the destination Replication Server or database. Replication Server builds stable queues using its disk partitions. See also **inbound queue**, **outbound queue**, and **materialization queue**.

standalone mode A special Replication Server mode used for initiating recovery operations.

standby database In a warm standby application, a database that receives data modifications from the active database and serves as a backup of that database. See also **warm standby application**.

stored procedure A collection of SQL statements and optional control-of-flow statements stored under a name in a Adaptive Server database. Stored procedures supplied with Adaptive Server are called system procedures. Some stored procedures for querying the RSSD are included with the Replication Server software.

subscription A request for Replication Server to maintain a replicated copy of a table, or a set of rows from a table, in a replicate database at a specified location. You can also subscribe to a function replication definition, for replicating stored procedures.

subscription dematerialization See **dematerialization**.

subscription materialization See **materialization**.

subscription migration See **row migration**.

Sybase Central A graphical tool that provides a common interface for managing Sybase and Powersoft products. Replication Server uses Replication Server Manager as a Sybase Central plug-in. See also **Replication Server Manager (RSM)**.

symmetric multiprocessing (SMP)
On a multiprocessor platform, the ability of an application's threads to run in parallel. Replication Server supports SMP, which can improve server performance and efficiency.

synchronous command
A command that a client considers complete only after the completion status is received.

system function
A function that is predefined and part of the Replication Server product. Different system functions coordinate replication activities, such as rs_begin, or perform data manipulation operations, such as rs_insert, rs_delete, and rs_update.

system-provided classes
The error class rs_sqlserver_error_class and the function-string classes rs_sqlserver_function_class, rs_default_function_class, and rs_db2_function_class that Replication Server provides. Function strings are generated automatically for the system-provided function-string classes and for any derived classes that inherit from these classes, directly or indirectly. See also **error class** and **function-string class**.

system version
The version number for a replication system that represents the version for which new features are enabled, for Replication Servers of release 11.0.2 or earlier, and below which no Replication Server can be downgraded or installed. For a Replication Server version 11.5, your use of certain new features requires a site version of 1150 and a system version of at least 1102. See also **mixed-version system**, **site version**, and **software version**.

table replication definition
See **replication definition**.

table subscription
A subscription to a table replication definition.

thread
A process running within Replication Server. Built upon Sybase Open Server, Replication Server has a multi-threaded architecture. Each thread performs a certain function such as managing a user session, receiving messages from a Replication Agent or another Replication Server, or applying messages to a database. See also **Data Server Interface (DSI)**, **Distributor**, and **Replication Server Interface (RSI)**.

transaction
A mechanism for grouping statements so that they are treated as a unit: either all statements in the group are executed or no statements in the group are executed.

Transact-SQL
The relational database language used with Adaptive Server. It is based on standard SQL (Structured Query Language), with Sybase extensions.

truncation point

An Adaptive Server database that holds primary data has an active truncation point, marking the transaction log location where Adaptive Server has completed processing. This is the primary truncation point.

The RepAgent for an Adaptive Server database maintains a secondary truncation point, marking the transaction log location separating the portion of the log successfully submitted to the Replication Server from the portion not yet submitted. The secondary truncation point ensures that each operation enters the replication system before its portion of the log is truncated.

user-defined function

A function that allows you to create custom applications that use Replication Server to distribute replicated functions or asynchronous stored procedures between sites in a replication system. In replicated function delivery, a user-defined function is automatically created by Replication Server when you create a function replication definition.

variable

See **function-string variable**.

version

See **mixed-version system**, **site version**, **software version**, and **system version**.

warm standby application

An application that employs Replication Server to maintain a standby database for a database known as the active database. If the active database fails, Replication Server and client applications can switch to the standby database.

wide-area network (WAN)

A system of local-area networks (LANs) connected together with data communication lines.

wide columns

Columns in a replication definition containing char, varchar, binary, varbinary, unichar, univarchar, or Java inrow data that are wider that 255 bytes. Wide columns are supported by Replication Server version 12.5 and later.

wide data

Wide data rows, limited to the size of the data page on the data server. Adaptive Server supports page sizes of 2K, 4K, 8K, and 16K. Wide data is supported by Replication Server version 12.5 and later.

wide messages

Messages larger than 16K that span blocks. Wide messages are supported by Replication Server version 12.5 and later.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

Z