

SYBASE®

Administration Guide: Volume 1

## **Replication Server®**

15.1

DOCUMENT ID: DC32511-01-1510-01

LAST REVISED: May 2008

Copyright © 1992-2008 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the [Sybase trademarks page](http://www.sybase.com/detail?id=1011207) at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

# Contents

<b>About This Book .....</b>	<b>xiii</b>	
<b>CHAPTER 1</b>	<b>Introduction .....</b>	<b>1</b>
	About Replication Server .....	1
	Asynchronous transaction replication.....	2
	Advantages of replicating local data.....	3
	Replication Server and distributed database systems .....	4
	Replication Server basic primary copy model .....	6
	Other distributed data models .....	9
	Replication Server and heterogeneous data servers .....	16
	Warm standby applications .....	17
	Mixed-version replication systems .....	18
	Restrictions in mixed-version systems .....	19
	Mixed versions of Adaptive Server.....	19
	Replication system security.....	20
	Replication Server security features.....	20
	Network-based security features.....	21
	Replication Server roles and responsibilities.....	21
	Replication system administrator .....	22
	Database administrator .....	22
	Replication Server tasks and responsibilities .....	22
<b>CHAPTER 2</b>	<b>Replication Server Technical Overview .....</b>	<b>25</b>
	Replication system components.....	25
	Replication Server .....	26
	Adaptive Server or other data server .....	28
	Client applications .....	31
	Sybase Central.....	31
	Replication Manager (RM) plug-in to Sybase Central .....	32
	Replication Monitoring Services (RMS).....	32
	Specifying data for replication .....	32
	Replication definitions and subscriptions for tables.....	33
	Replication definitions for database objects .....	33

- Replication definitions for stored procedures ..... 34
- Publications ..... 35
- Overview of replicating tables ..... 36
- Commands for managing replicated data ..... 37
- Establishing Replication Server connections ..... 38
  - Interfaces file ..... 38
  - LDAP server ..... 39
  - Making Replication Server connections ..... 40
- Specifying database operations ..... 42
  - Function strings ..... 42
  - Function-string classes ..... 43
- Transaction handling with Replication Server ..... 43
  - Stable queues ..... 44
  - Distributed concurrency control ..... 48
  - Transaction processing by the Replication Agent ..... 50

**CHAPTER 3                    Managing Replication Server with Sybase Central ..... 53**

- Using Replication Manager from Sybase Central ..... 53
  - Starting and stopping Sybase Central ..... 53
  - Getting started ..... 54
  - Using online help ..... 54
  - Using the Replication Manager GUI ..... 55
- Setting up a replication environment ..... 64
  - Preparing for a two-tier solution ..... 64
  - Creating an environment ..... 65
  - Connecting to and disconnecting from  
    a replication environment ..... 66
  - Setting up a replication environment  
    using Replication Manager ..... 67
  - Managing Replication Server objects ..... 70
- Monitoring a replication environment using RMS ..... 75
  - Preparing for a three-tier solution ..... 75
  - Connecting to RMS ..... 75
  - Adding and dropping servers through RMS ..... 76
  - Viewing managed objects ..... 76
  - Adding event triggers ..... 76

**CHAPTER 4                    Managing a Replication System ..... 79**

- Setting up a replication system ..... 79
  - Creating connections and routes ..... 80
  - Setting permissions and security ..... 80
  - Verifying the replication system ..... 81
  - Creating replication definitions ..... 81

Creating subscriptions .....	82
Performing Replication Server tasks .....	82
Using rs_init.....	83
Managing Replication Server with Sybase Central .....	83
Using isql.....	84
Starting Replication Server .....	86
Replication Server executable program .....	86
Replication Server configuration file.....	87
Shutting down Replication Server .....	87
Adding a Replication Server.....	88
Adding a replication system domain .....	89
Guidelines for adding replication system domains .....	89
Setting Replication Server configuration parameters.....	90
About configuration parameters .....	91
Changing Replication Server parameters .....	94
Managing the RSSD .....	95
Enabling Failover support for an RSSD connection .....	96
Managing Embedded Replication Server	
System Database .....	97
Overview .....	97
Before you start.....	98
Configuring ERSSD.....	98
Configuration parameters and command .....	99
ERSSD routing.....	100
Moving ERSSD files .....	100
ERSSD users .....	100
Backup .....	101
Recovery instructions.....	101
Quiescing Replication Server.....	104
Quiescing a replication system.....	104
Removing a Replication Server.....	105
Removing an active Replication Server .....	105
Removing an inactive Replication Server.....	107

<b>CHAPTER 5</b>	<b>Setting Up and Managing RepAgent.....</b>	<b>111</b>
	Setting up RepAgent .....	112
	Defining the local Adaptive Server .....	112
	Enabling RepAgent on Adaptive Server.....	113
	Enabling RepAgent for the database .....	113
	Configuring RepAgent.....	114
	Starting RepAgent.....	118
	Stopping RepAgent.....	119
	Disabling RepAgent .....	119
	Checking log files for information and error messages .....	120

- Configuring RepAgent for network security..... 120
- Handling extended limits ..... 121
- Support for longer identifiers ..... 122
- Adaptive Server shared-disk cluster support ..... 123
- Reviewing status and configuration information..... 124
  - Viewing RepAgent information ..... 124
  - Viewing configuration parameter values ..... 125
  - Viewing RepAgent thread information..... 125
- Managing log transfer activity ..... 126
  - Using the log transfer commands..... 126
  - Using alter connection and the set log transfer option ..... 128
- Using counters to monitor RepAgent performance ..... 128
  - Invoking sp\_sysmon..... 130
  - RepAgent counter activity ..... 130

**CHAPTER 6**                      **Managing Routes..... 137**

- Overview ..... 137
- Before you begin ..... 138
  - Routing preparations ..... 139
- Routing schemes ..... 140
  - Direct routes ..... 140
  - Indirect routes..... 141
  - Unsupported routing schemes ..... 144
- Creating routes..... 144
  - Using the create route command ..... 145
  - Configuring a Replication Server to manage primary tables. 148
- Suspending and resuming routes ..... 150
  - Using the suspend route command..... 150
  - Using the resume route command ..... 150
- Changing routes ..... 151
  - Changing route topology ..... 151
  - Changing the password for the RSI user for a direct route ... 154
  - Changing parameters affecting direct routes ..... 155
  - Routing modification example ..... 156
- Dropping routes..... 158
  - Using the drop route command ..... 159
  - Using the sysadmin purge\_route\_at\_replicate command .... 160
- Upgrading routes..... 160
- Monitoring routes ..... 161
  - Displaying RSI thread status using admin who ..... 161
  - Using the rs\_helproute stored procedure ..... 162

**CHAPTER 7**                      **Managing Database Connections ..... 163**

Preparing databases for replication .....	163
Steps in preparing databases for replication .....	164
Upgrading an existing Adaptive Server database .....	165
Managing maintenance user login names .....	165
Finding the current maintenance user .....	166
Granting permissions in the database .....	166
Creating database connections.....	167
Information for adding a database connection .....	168
Using the create connection command .....	169
Altering database connections .....	170
Suspending database connections .....	171
Setting and changing parameters affecting physical connections .....	172
Resuming database connections .....	183
Changing replicate databases to primary databases .....	184
Changing primary databases to replicate databases .....	186
Dropping database connections.....	187
Dropping a database from the ID Server.....	187
Monitoring database connections .....	188
Viewing current database connections.....	188
Listing databases managed by a Replication Server .....	188
Displaying DSI thread status .....	189

## CHAPTER 8

<b>Managing Replication Server Security.....</b>	<b>191</b>
Overview .....	191
Managing Replication Server system security .....	192
RSSD login names and passwords .....	193
Replication Server login name and password for the RepAgent .....	194
ID Server login name and password .....	194
Replication Server login name and password for Replication Servers .....	195
Maintenance user Adaptive Server login name and password	195
Sending encrypted passwords for Replication Server client connections .....	196
Existing Encrypted Password Migration .....	196
Extended password encryption support .....	197
Sybase Central dependencies .....	198
Replication Server object creation dependencies .....	198
Managing Replication Server user security.....	199
Managing Replication Server login names and passwords...	200
Enabling and disabling password encryption in sysattributes	201
Managing Replication Server permissions .....	202
Examining users, passwords, and permissions .....	208

- Managing network-based security ..... 210
  - How security services work ..... 211
  - Requirements and restrictions..... 212
  - Setting up network-based security ..... 213
  - Modifying configuration parameters and environment variables ..... 213
  - Configuring objectid.dat..... 215
  - Configuring the interfaces file..... 215
  - Setting environment variables (Kerberos) ..... 216
  - Establishing the principal user..... 216
  - Identifying principal users to Replication Server ..... 218
  - Activating network-based security..... 218
  - Starting server and clients..... 219
  - Configuring security services for Replication Server..... 219
  - Maintaining network security ..... 232
- Managing SSL security ..... 237
  - SSL overview ..... 237
  - SSL on Replication Server ..... 238
  - Setting up SSL security ..... 239
  - Enabling SSL security ..... 239

**CHAPTER 9**

- Managing Replicated Tables ..... 241**
  - Introduction ..... 242
  - Planning a replication system ..... 243
    - Design considerations ..... 243
    - Restrictions on data replication ..... 244
    - Preparing a replication system ..... 244
  - Summarizing the process..... 245
    - Replication procedure ..... 246
    - Commands for managing table replication definitions..... 249
  - Creating replication definitions ..... 250
    - Replication definition settings..... 250
    - Using the create replication definition command..... 251
    - Creating replication definitions using extended limits..... 263
    - Creating multiple replication definitions per table..... 265
    - Replication definitions and function strings ..... 267
    - Replication definition restrictions in mixed-version systems . 268
  - Marking tables for replication ..... 270
    - Using the sp\_setreptable system procedure ..... 270
  - Replicating Java columns ..... 273
    - Restrictions..... 273
    - Upgrade considerations ..... 273
    - Java datatypes in Replication Server ..... 274
    - Creating replication definitions for Java columns ..... 274



Function strings for Java columns .....	275
Replicating text, unitext, image, and rawobject columns .....	277
Replicating large objects to non-ASE servers using DirectConnect Anywhere.....	279
Creating a text, unitext, image, or rawobject replication definition .....	279
Marking tables with text, unitext, image, or rawobject columns	280
Changing column status for text, unitext, image, or rawobject columns .....	281
Altering replication status for text, unitext, image, and rawobject columns .....	283
Resolving inconsistencies in replication status .....	284
Subscription issues for replicate_if_changed status .....	286
Function strings for replicating text, unitext, and image data	287
Replicating new large-object (LOB) datatypes .....	287
Replicating computed columns .....	288
Replicating encrypted columns .....	289
Working with special datatypes .....	291
Using the rs_address datatype.....	291
Replicating identity columns.....	291
Replicating timestamp columns.....	292
Modifying replication definitions .....	293
Maintaining table schema.....	294
Viewing existing replication definitions .....	298
Altering replication definitions.....	298
Dropping replication definitions .....	304
Modifying replicated data .....	305
Adding a new table .....	305
Renaming replicated tables.....	305
Dropping a replicated table .....	305
Adding columns in source and destination tables .....	306
Deleting columns in a source or destination table .....	306
Changing searchable columns .....	307
Changing column datatypes in a source or destination table	307
Using publications .....	308
Using publications to replicate data at the command line .....	309
Translating datatypes using HDS.....	317
Overview .....	317
Getting started .....	318
Creating class-level translations.....	319
Creating column-level translations .....	322
Using class-level and column-level translations together.....	326
Verifying translations .....	327

<b>CHAPTER 10</b>	<b>Managing Replicated Functions.....</b>	<b>329</b>
	Prerequisites and restrictions .....	330
	Replicated function prerequisites .....	330
	Replicated function restrictions .....	331
	Commands for managing function replication definitions.....	333
	Using replicated functions .....	334
	Applied functions .....	335
	Request functions.....	336
	Implementing an applied function .....	337
	Implementing a request function .....	340
	Marking stored procedures for replication .....	344
	Subscribing to replicated functions .....	345
	Modifying or dropping replicated functions.....	345
	Before modifying a function replication definition .....	345
	Altering a function replication definition.....	346
	Modifying a function replication definition.....	346
	Dropping a function replication definition .....	347
	Creating or modifying a function string for a replicated function .....	348
	Using publications for stored procedures.....	349
<b>CHAPTER 11</b>	<b>Managing Subscriptions .....</b>	<b>351</b>
	Overview .....	351
	Subscription materialization methods .....	353
	Atomic materialization .....	354
	Nonatomic materialization .....	355
	No materialization.....	357
	Bulk materialization .....	357
	Dematerialization processing .....	365
	Dematerializing and purging rows .....	365
	Dematerialization without purging rows.....	366
	Monitoring materialization and dematerialization .....	366
	Before you create subscriptions .....	368
	Using subscription commands .....	370
	Using the where clause .....	371
	Enabling replication of truncate table .....	373
	Using the create subscription command .....	374
	Using the define subscription command .....	377
	Using the activate subscription command.....	378
	Using the validate subscription command.....	379
	Using the check subscription command.....	379
	Using the drop subscription command .....	380
	Subscription example.....	382
	Description of replication system.....	382

Procedures for replicating tables ..... 383

Materializing text, untext, image, and rawobject data ..... 386

    Nonatomic materialization ..... 386

    Row migration ..... 386

Subscriptions for columns with heterogeneous datatypes ..... 387

Bitmap subscriptions ..... 388

Obtaining subscription information ..... 390

    Displaying subscription information ..... 391

    Verifying subscription consistency ..... 391

Using publication subscriptions ..... 395

    Commands for creating and managing  
        publication subscriptions ..... 396

    Creating publication subscriptions ..... 397

    Dropping subscriptions for publications and articles ..... 401

    Viewing publication subscription information ..... 402

**CHAPTER 12**

**Managing Replicated Objects Using Multisite Availability ..... 405**

Overview ..... 406

Setting up an MSA system ..... 408

    Replicating the database ..... 408

    Replicating tables and functions ..... 410

    Using replicate databases as warm standby databases ..... 411

Marking data for replication ..... 414

Managing database replication definitions ..... 415

    Altering database replication definitions ..... 415

    Dropping database replication definitions ..... 416

    Using database replication filters ..... 417

Viewing information about database replication definitions ..... 418

Using database, table, and function replication  
    definitions concurrently ..... 418

    Altering database replication definitions ..... 419

    Altering table and function replication definitions ..... 420

Managing database subscriptions ..... 420

    Materialization ..... 421

    Altering database subscriptions ..... 422

    Dropping database subscriptions ..... 422

Viewing information about database subscriptions ..... 423

Using database, table, and function subscriptions concurrently .. 423

    Creating and dropping subscriptions ..... 424

Replicating the master database in an MSA environment ..... 424

Replicating DDL and system procedures ..... 426

Replicating user stored procedures ..... 427

Customizing function strings ..... 427

**Index ..... 429**

# About This Book

Sybase® 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, *Replication Server Administration Guide*, provides an overview of how Replication Server works, and describes Replication Server administrative tasks.

## Audience

The *Replication Server Administration Guide* is for replication system administrators, who manage the routine operation of their Replication Servers. Any user who has been granted the sa permission can be a replication system administrator, although each Replication Server usually has just one.

## How to use this book

This book contains the following chapters:

- Chapter 1, “Introduction” introduces you to Replication Server, describing the role it plays in a distributed database system and its concepts and components.
- Chapter 2, “Replication Server Technical Overview” provides a technical overview of the replication system, giving you the background necessary to maintain and troubleshoot the system.
- Chapter 3, “Managing Replication Server with Sybase Central” describes using Sybase Central’s Replication Manager plug-in, which is a graphical tool for managing Replication Server.
- Chapter 4, “Managing a Replication System” describes basic operations such as starting, stopping, and configuring Replication Server.
- Chapter 5, “Setting Up and Managing RepAgent,” describes how to set up, configure, and manage RepAgent.
- Chapter 6, “Managing Routes” describes how to create and manage routes between source and destination Replication Servers.

- 
- Chapter 7, “Managing Database Connections” describes how to prepare databases for replication and how to create and manage connections between databases and Replication Servers.
  - Chapter 8, “Managing Replication Server Security” describes how to create and modify login names, passwords, and permissions and how to set up network-based security.
  - Chapter 9, “Managing Replicated Tables” describes how to set up and manage replicated tables.
  - Chapter 10, “Managing Replicated Functions” describes how to copy the execution of user stored procedures to remote sites in a replication system using replication definitions.
  - Chapter 11, “Managing Subscriptions” describes how to create and manage subscriptions, which allow Replication Server to replicate data between databases.
  - Chapter 12, “Managing Replicated Objects Using Multisite Availability,” describes how to create and manage database replication definitions and database subscriptions.

Volume 2 of the System Administration Guide contains these chapters:

- Chapter 1, “Verifying and Monitoring Replication Server” describes checking error logs, verifying that the components of a replication system are running, and monitoring the status of system components and processes.
- Chapter 2, “Customizing Database Operations” describes how to use functions, function strings, and function-string classes to customize data replication with Adaptive Server® Enterprise and data servers from other vendors.
- Chapter 3, “Managing Warm Standby Applications” describes how to create and manage warm standby applications.
- Chapter 4, “Performance Tuning” describes how to manage resources effectively and optimize the performance of individual Replication Servers.
- Chapter 5, “Using Counters to Monitor Performance” describes Replication Server counters and how to use them.
- Chapter 6, “Handling Errors and Exceptions” discusses error conditions and failed transactions and how to customize data server responses to errors.

- Chapter 7, “Replication System Recovery” describes replication system failure conditions and provides procedures for recovering from them.
- Appendix A, “Asynchronous Procedures” describes a method for replicating stored procedures associated with table replication definitions.
- Appendix B, “High Availability on Sun Cluster 2.2,” provides background and procedures for configuring Sybase Replication Server for high availability (HA) on Sun Cluster 2.2.
- Appendix C, “Pre-15.1 Request Function Replication” provides information about request function replications with versions earlier than 15.1.

**Related documents**

The Sybase Replication Server documentation set consists of:

- The 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 release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Product Manuals at <http://www.sybase.com/support/manuals/>.

- *Installation Guide* for your platform – describes installation and upgrade procedures for all Replication Server and related products.
- *New Features Guide* – describes the new features in Replication Server version 15.1 and the system changes added to support those features.
- *Administration Guide* (this book) – 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.
- *Heterogeneous Replication Guide* – describes how to use Replication Server to replicate data between databases supplied by different vendors.

- 
- *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 Manager plug-in help, which contains information about using Sybase Central™ to manage Replication Server.

#### **Other sources of information**

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

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

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

Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

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



**Sybase certifications on the Web**

Technical documentation at the Sybase Web site is updated frequently.

**❖ 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

This section describes the style and syntax conventions, RCL command formatting conventions, and icons used in this book.

**Style conventions** Syntax statements that display the syntax and options for a command are printed as follows:

```
alter user user
set password new_passwd
[verify password old_passwd]
```

See “Syntax conventions” on page xviii for more information.

Examples that show the use of Replication Server commands are printed as follows:

```
alter user louise
set password somNific
verify password EnnuI
```

Command names, command option names, program names, program flags, keywords, functions, and stored procedures are printed as follows:

Use `alter user` to change the password for a login name.

Variables, parameters, and user-supplied words are in italics in syntax and in paragraph text, as follows:

The `set password new_passwd` clause specifies a new password.

Names of database objects such as databases, tables, columns, and datatypes, are in italics in paragraph text, as follows:

The `base_price` column in the `Items` table is a money datatype.

Names of replication objects, such as function-string classes, error classes, replication definitions, and subscriptions, are in italics.

**Syntax conventions** Syntax formatting conventions are summarized in the following table. Examples combining these elements follow.

**Table 1: Syntax formatting conventions**

Key	Definition
{ }	Curly braces mean you must choose at least one of the enclosed options. Do not include braces in the command.
[ ]	Brackets mean you may choose or omit enclosed options. Do not include brackets in the command.
	Vertical bars mean you may choose no more than one option (enclosed in braces or brackets).
,	Commas mean you may choose as many options as you need (enclosed in braces or brackets). Separate your choices with commas, to be typed as part of the command. Commas may also be required in other syntax contexts.
( )	Parentheses are to be typed as part of the command.
...	An ellipsis (three dots) means you may repeat the last unit as many times as you need. Do not include ellipses in the command.

**Obligatory choices**

- Curly braces and vertical bars – choose only one option.  
`{red | yellow | blue}`
- Curly braces and commas – choose one or more options. If you choose more than one, separate your choices with commas.  
`{cash, check, credit}`

**Optional choices**

- One item in square brackets – choose it or omit it.  
`[anchovies]`
- Square brackets and vertical bars – choose none or only one.  
`[beans | rice | sweet_potatoes]`
- Square brackets and commas – choose none, one, or more options. If you choose more than one, separate your choices with commas.  
`[extra_cheese, avocados, sour_cream]`

**Repeating elements**

An ellipsis (...) means that you may repeat the last unit as many times as necessary. For the alter replication definition command, for example, you can list one or more columns and their datatypes for the add clause or the add searchable columns clause:

```
alter replication definition replication_definition
{add column datatype [, column datatype]... |
add searchable columns column [, column]... |
replicate {minimal | all} columns}
```

---

RCL command  
formatting

RCL commands are similar to Transact-SQL® commands. The following sections present the formatting rules.

Command format and  
command batches

- You can break a line anywhere except in the middle of a keyword or an identifier. You can continue a character string on the next line by typing a backslash (\) at the end of the line.
- Extra spaces are ignored, except after a backslash. Do not enter any spaces after a backslash.
- You can enter more than one command in a batch unless otherwise instructed.
- RCL commands are not transactional. Each command is executed independently and is not affected by the completion status of other commands in the batch. However, syntax errors in a command prevent Replication Server from executing subsequent commands in a batch.

Case sensitivity

- Keywords in RCL commands are not case sensitive. You can enter them in any combination of uppercase or lowercase letters.
- Case sensitivity in identifiers and character data depends on the sort order that is in effect.
  - If you use a case-sensitive sort order such as “binary,” you must enter identifiers and character data in the correct combination of uppercase and lowercase letters.
  - If you use a sort order that is not case sensitive, such as “nocase,” you can enter identifiers and character data in any combination of uppercase or lowercase letters.

Identifiers

Identifiers are names you give to servers, databases, variables, parameters, database objects, and replication objects. Database object names include names for tables, columns, and views. Replication object names include names for replication definitions, subscriptions, functions, and publications.

- Identifiers can be 1 – 255 bytes long (equivalent to 1 – 255 single-byte characters) and must begin with a letter, the @ sign, or the \_ character. See “Support for longer identifiers” on page 122 for a list of identifiers that have been extended to 255 bytes.
- Replication Server function parameters are the only identifiers that can begin with the @ character. Function parameter names can include 255 characters *after* the @ character.
- After the first character, identifiers can include letters, digits, and the #, \$, or \_ characters. Spaces are not allowed.

**Parameters in function strings**

Parameters in function strings have the same rules as identifiers, except that:

- They are enclosed in question marks (?). This allows Replication Server to locate them in the function string. Use two consecutive question marks (??) to represent a literal question mark in a function string.
- The exclamation point (!) introduces a parameter modifier that indicates the source of the data to be substituted for a parameter at runtime. Refer to the *Replication Server Reference Manual* for a list of modifiers.




**Data support** Replication Server supports all Adaptive Server datatypes.



User-defined datatypes are not supported. The double precision, nchar, and nvarchar datatypes are indirectly supported; they are mapped to other datatypes.

For more information about the supported datatypes, including how to format them, see “Datatypes,” in Chapter 2, “Topics” of the *Replication Server Reference Manual*.

**Icons**

Illustrations in this book use icons to represent the components of a replication system.

	<b>Description</b>
	<p>This icon represents Replication Server, the Sybase server program maintains replicated data on a local-area network (LAN) and processes data transactions received from other Replication Servers on wide-area network (WAN).</p>
	<p>This icon represents Adaptive Server, the Sybase data server. Data servers manage databases containing primary or replicated data. Replication Server also works with heterogeneous data servers, so, unless otherwise noted, this icon can represent any data server in a replication system.</p>
	<p>This icon represents Replication Agent™, a replication system process or module that transfers transaction log information for primary database to a Replication Server. The Replication Agent for Adaptive Server is RepAgent. Sybase provides Replication Agent products for Adaptive Server® Anywhere, DB2, Microsoft SQL Server, and Oracle data servers.</p> <p>Except for RepAgent, which is an Adaptive Server thread, all Replication Agents are separate processes. In general, this icon only appears when representing a Replication Agent that is a separate process.</p>

	Description
	<p>This icon represents client application. A client application is a user process or application connected to a data server. It may be a front-end application program executed by a user or a program that executes as an extension of the system.</p>
	<p>This icon represents the Sybase Central Replication Manager plug-in (RM), a management utility that lets a replication system administrator develop, manage, and monitor a Sybase Replication Server environment.</p>

## 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 HTML documentation has 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.

## 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.

# Introduction

This chapter introduces you to Replication Server and its role in a distributed database system. It also discusses the benefits and features of Replication Server, methods and concepts for replicating data, and Replication Server support for heterogeneous data servers, as well as defining user roles in maintaining a replication system.

<b>Topic</b>	<b>Page</b>
About Replication Server	1
Replication Server and distributed database systems	4
Warm standby applications	17
Mixed-version replication systems	18
Replication system security	20
Replication Server roles and responsibilities	21

## About Replication Server

Replication Server maintains replicated data in multiple databases while ensuring the integrity and consistency of the data. It provides clients using databases in the replication system with local data access, thereby reducing load on the network and centralized computer systems.

The Replication Command Language (RCL) enables you to customize replication functions and to monitor and maintain the replication system. For example, you can request subsets of data for replication at the table, data row, or column level. This feature further reduces overhead by allowing you to replicate only the data that is needed at the replicate site.

Replication Server supports heterogeneous data servers. You can build a replication system from existing databases and applications without having to convert them. As your enterprise grows and changes, you can add data servers to your replication system to meet your needs.

Replication Server uses a basic publish-and-subscribe model for replicating data across networks. Users “publish” data that is available in a primary database, and other users “subscribe” to the data for delivery in a replicate database. Users can replicate both changes to the data (update/insert/delete operations) and stored procedures using this method.

Instructions to publish and subscribe to data are given at Replication Servers that control, or have a **connection** to, each database. The user creates a **replication definition** at a primary Replication Server, which controls the primary database containing the data to be published. The replication definition specifies information such as which columns are to be replicated, or in the case of a **database replication definition**, of the database objects to be replicated. The user creates a **subscription** at a replicate Replication Server, which controls the replicate database that will receive the information.

Replication Servers communicate with each other via user-defined *routes*. Most commonly, a primary Replication Server sends data to a replicate Replication Server through one or more routes set up to transmit data from the primary database to the replicate database. Users may also transmit stored procedures from the replicate to the primary to request updates of the primary data; in this case, data flows through one or more routes from the replicate Replication Server to the primary Replication Server.

Connections and routes define the structure of the replication system. They allow Replication Servers to send messages to each other and to send commands to databases. A connection transfers messages from a Replication Server to a database. A route transfers requests from a source Replication Server to a destination Replication Server.

## Asynchronous transaction replication

Replication occurs asynchronously—that is, updates to data at the primary are transferred to replicate databases in transactions separate from the update itself. While asynchronous replication provides important advantages, system designers should remain aware of the latency between initial and replicated updates.



## Advantages of replicating local data

Replicating tables on local data servers provides clients with local access to enterprise data, which results in improved performance and greater data availability.

### Improved performance

In a typical Replication Server system, data requests are completed on the local data server without accessing the WAN. Therefore, local clients gain improved performance because:

- Data transfer rates are faster on a LAN than they are on a WAN.
- Local access remains unaffected by network traffic over the WAN. Local clients that share local data server resources do not compete with the enterprise-wide user community for central resources.

### Greater data availability

Because data is replicated at local and remote databases in a Replication Server system, clients can operate in a fault-tolerant environment so that:

- When a failure occurs at a remote database, clients can use local copies of replicated data.
- When a WAN failure occurs, clients can use local copies of replicated data.
- When the local data server fails, clients can use replicated data at another site.

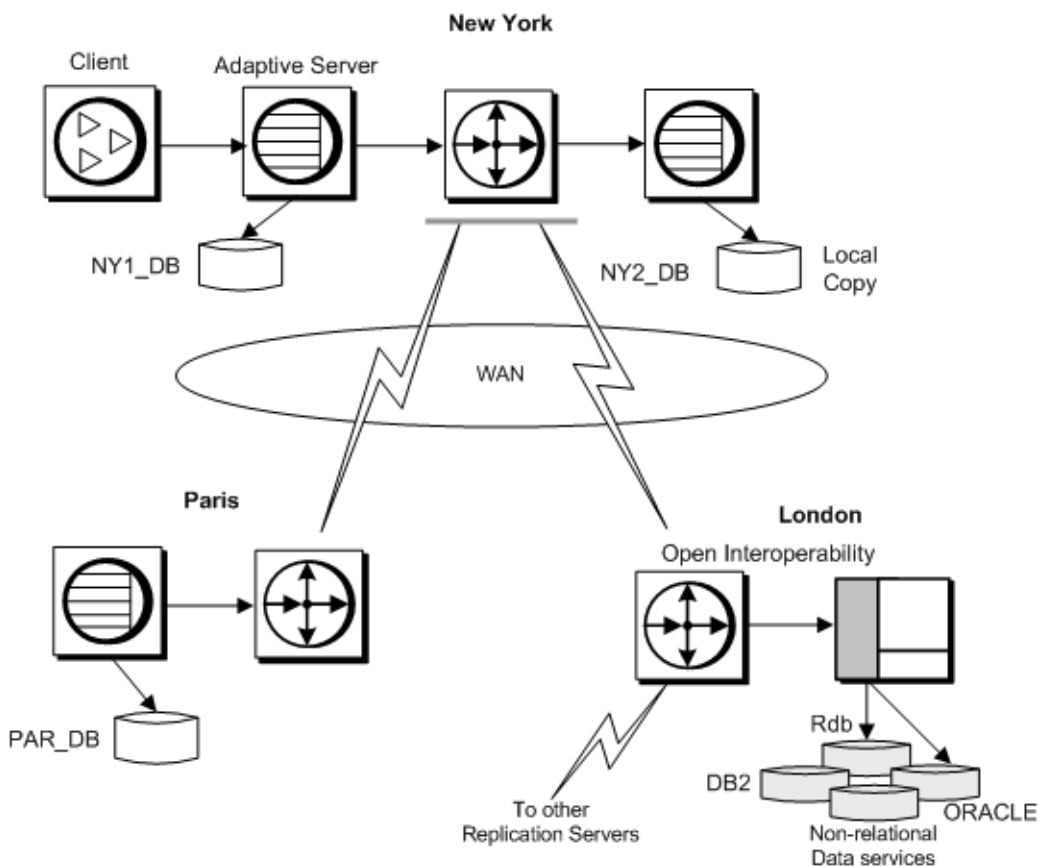
Network failure or database failure at other locations do not halt work at the local database. When WAN communications fail, Replication Server stores operations on replicated tables in **stable queues** (disk storage). The replicated tables at the unavailable databases are updated when communications resume. If a local data server fails, clients can continue working by temporarily accessing a replicate copy of the data.

## Replication Server and distributed database systems

Distributed database systems allow client applications to access data on multiple database servers throughout an enterprise—even geographically dispersed enterprises. Replication Server ensures that data on replicate databases stays updated while off-loading processing responsibilities from the source database.

As Figure 1-1 illustrates, these enterprises may consist of many LANs and one or more WANs.

**Figure 1-1: Replication system in a distributed environment**



Replication Server minimizes performance and data-availability problems typically associated with remote access in distributed systems. Since Replication Server provides multiple copies of data, clients can rely on their own local data instead of remote, centralized databases. In addition, you can copy only the data you want to destination databases. Replication Server allows you to create a replication definition that identifies all or part of a table to replicate. You can then subscribe to only the rows you want. You can create a database replication definition that identifies the database objects—tables, functions, system procedures, transactions, and data definition language (DDL)—to replicate. You can also create a replication definition of a stored procedure (called **function replication definition**) to facilitate rapid replication of large amounts of data and to replicate updates from replicate databases back to the primary database. If your application requires it, you can consolidate or “roll up” replicated data from primary tables into a centralized database.

You can group replication definitions, both table replication definitions and functions replication definitions, in a **publication** and subscribe to them all at once. Publications allow you to organize subscriptions and then monitor them with a single command.

**A Replication Agent**—RepAgent for sites running Adaptive Server—transfers transaction information from a database to a Replication Server for distribution to replicate databases. Sybase also offers Replication Agent for Microsoft SQL Server, DB2, and Oracle. You can make Replication Agents for IMS and VSAM using the Sybase Replication Toolkit™ for MVS. RepAgent is an Adaptive Server thread; all other Replication Agents are separate processes.

Several models for replicating data in distributed systems exist in Replication Server. Consult the *Replication Server Design Guide* to help you determine which model best suits your application. The model that you choose determines how you set up your system.

Setting up a replication system based on your distribution model involves:

- Creating tables to store primary and replicate data
- Setting up routes and connections between Replication Servers and establishing permissions that control access to primary data
- Creating replication definitions that identify the data you want replicated
- Creating subscriptions from replicate databases to those replication definitions

See Chapter 2, “Replication Server Technical Overview” for a discussion of Replication Server components, concepts, and terminology. See Chapter 4, “Managing a Replication System” for a more detailed overview of setting up a Replication Server system.

## **Replication Server basic primary copy model**

The simplest approach Replication Server uses to copy data is to distribute updates from one source (primary) database to one or more destination (replicate) databases. To ensure consistency, a source table is designated as the primary table. All other versions of the table are replicates. In this approach, replicate tables are read-only and used for operations that do not modify the data.

As updates occur at the primary table, Replication Server captures the updates and sends them to replicate data servers. In this model, clients at remote sites can also update primary data, either directly by accessing the primary database over the network or indirectly through replicated stored procedures.

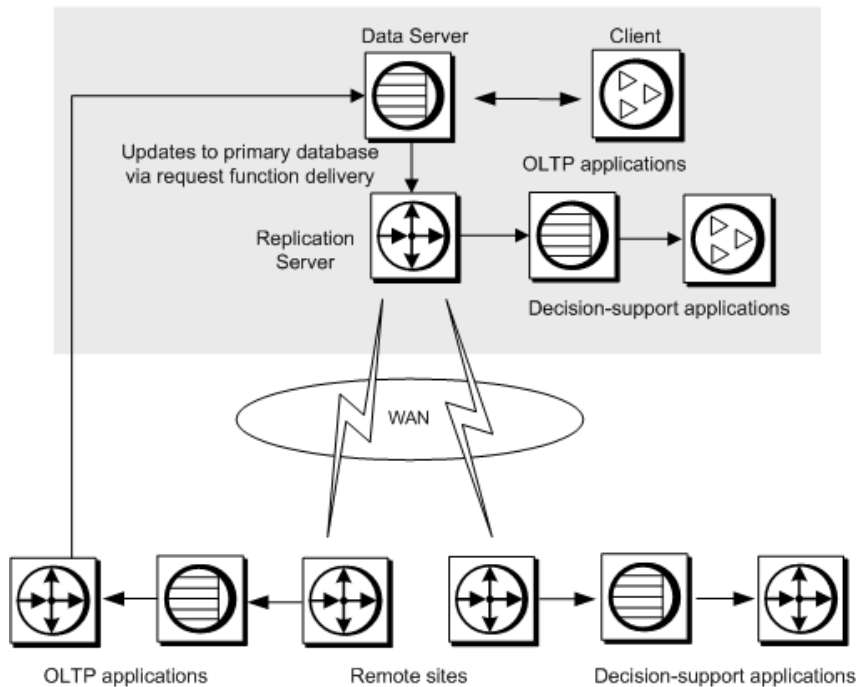
For more information, see “Specifying data for replication” on page 32, and Chapter 10, “Managing Replicated Functions.”

If communication between the primary and destination databases fails, operations executed in the primary database are stored in Replication Server stable queues until they can be delivered to replicate sites. Likewise, operations executed remotely are held in stable queues until they can be delivered to the primary database.

This arrangement lets remote client applications take advantage of Replication Server fault tolerance while preserving the basic primary copy model. See “Transaction handling with Replication Server” on page 43 for more information about stable queues.

Figure 1-2 illustrates Replication Server configurations using the primary copy method of replicating data.

Figure 1-2: Replication Server basic primary copy model



## Replication system processing

This section describes a typical replication system, according to the basic primary copy model, in which a primary Replication Server and a data server are separated across a WAN from replicate Replication Servers. It does not cover the case where primary data is updated at the replicate database.

**Figure 1-3: Replication system overview**

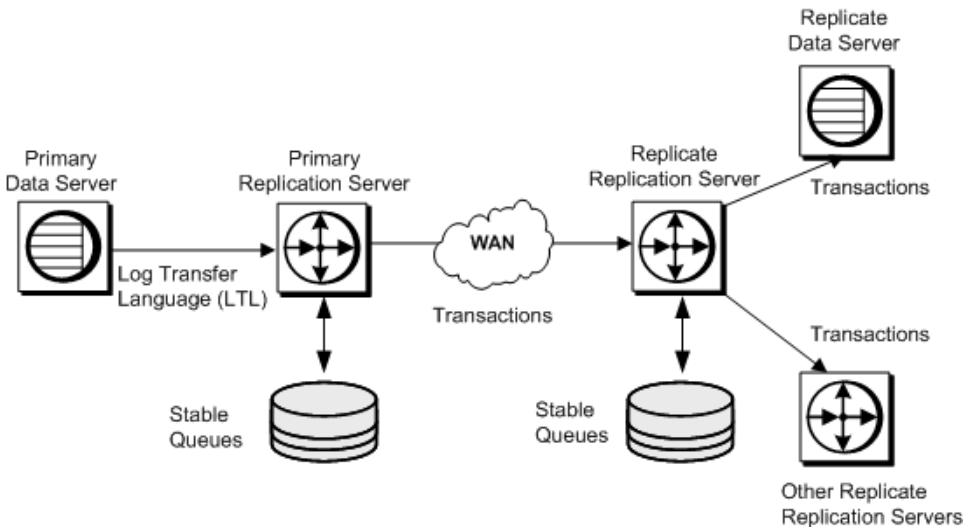


Figure 1-3 illustrates how data is replicated from a primary database to replicate databases. The following actions take place:

- 1 RepAgent reads the primary database log and converts transactions for tables or stored procedures that are marked for replication into commands that are sent to Replication Server.

The Replication Server stores the transactions in a stable queue (see “Distributed concurrency control” on page 48).

- 2 The primary Replication Server:
  - a Determines which Replication Servers manage replicate databases with subscriptions for the data
 

The primary Replication Server may have a direct route to a subscribing Replication Server or an indirect route, with one or more intermediate Replication Servers in between.
  - b Forwards the transaction to the appropriate replicate Replication Server, where it is stored in a stable queue
  - c Applies the transaction to any local replicate database for which there is a subscription for the data
- 3 The replicate Replication Server performs one or both of the following actions:

- Routes the transaction to another Replication Server
- Applies the transaction to replicate databases that it manages

## Setting up a primary copy model system

In order to set up a system according to the basic primary copy model, you need to:

- Set up routes and connections between Replication Servers.

For information on these topics, see Chapter 6, “Managing Routes” and Chapter 7, “Managing Database Connections.”

- Create the table in the primary and replicate databases. The table should have the same structure in each database.
- Create indexes and grant appropriate permissions on the tables.

For information on setting permissions for a Replication Server system, see Chapter 8, “Managing Replication Server Security.”

- Allow replication on the tables using the `sp_setreptable` system procedure.
- Create a replication definition for the table at the primary site.

For information about creating replication definitions, see Chapter 9, “Managing Replicated Tables.”

- At each site, create a subscription for the table replication definition at the primary site.

For information about creating subscriptions, see Chapter 11, “Managing Subscriptions.”

## Other distributed data models

Besides the basic primary copy model, Replication Server also lets you design your system based on other distributed data models, including:

- Distributed primary fragments
- Corporate rollup
- Redistributed corporate rollup

These models are discussed briefly in this section. For complete information about these distributed data models, refer to Chapter 3, “Implementation Strategies,” in the *Replication Server Design Guide*.

Warm standby applications represent another type of application model. See Chapter 3, “Managing Warm Standby Applications” in the *Replication Server Administration Guide Volume 2* for more information.

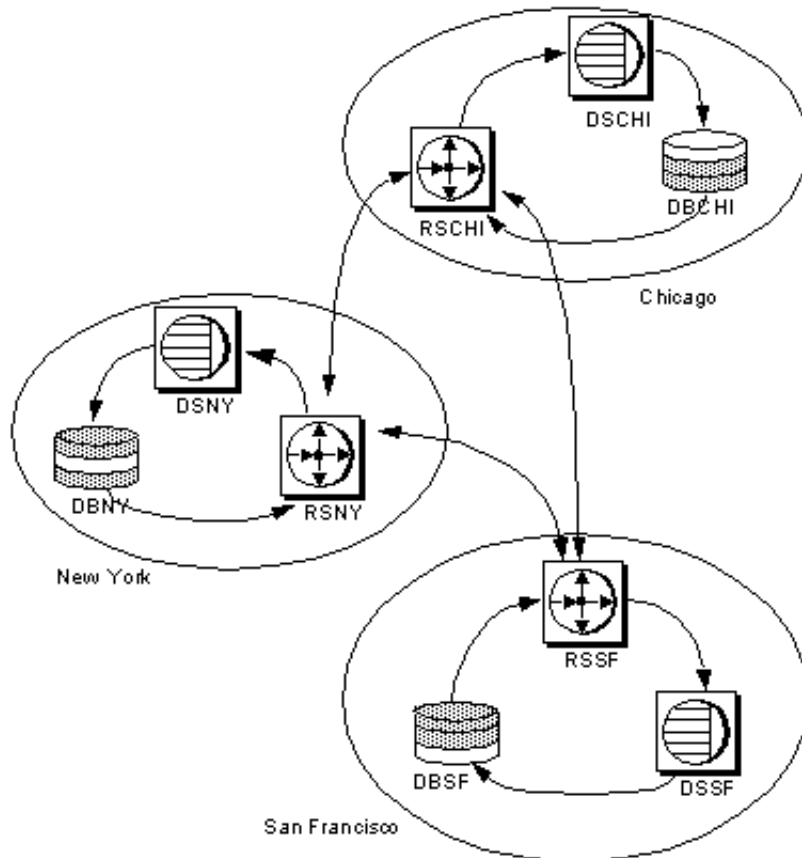
## **Distributed primary fragments**

Applications that use the distributed primary fragments model include distributed tables that contain both primary and replicated data. The Replication Server at each site distributes modifications made to local primary data to other sites and applies modifications received from other sites to the data that is replicated locally.

Figure 1-4 diagrams the flow of data for distributed primary fragments.



Figure 1-4: Distributed primary fragments model



The tasks needed to set up a distributed primary fragment system are similar to those for creating a basic primary copy system, with the following exceptions and additions:

- Your application should avoid or handle cases where multiple sites update the same data at the same time. Sybase recommends that each fragment have a single “owner” site.
- Databases can be both primary and replicate. Make sure that tables with the same structure exist at both primary and replicate sites.
- Create routes from each primary site to all sites that subscribe to its data.

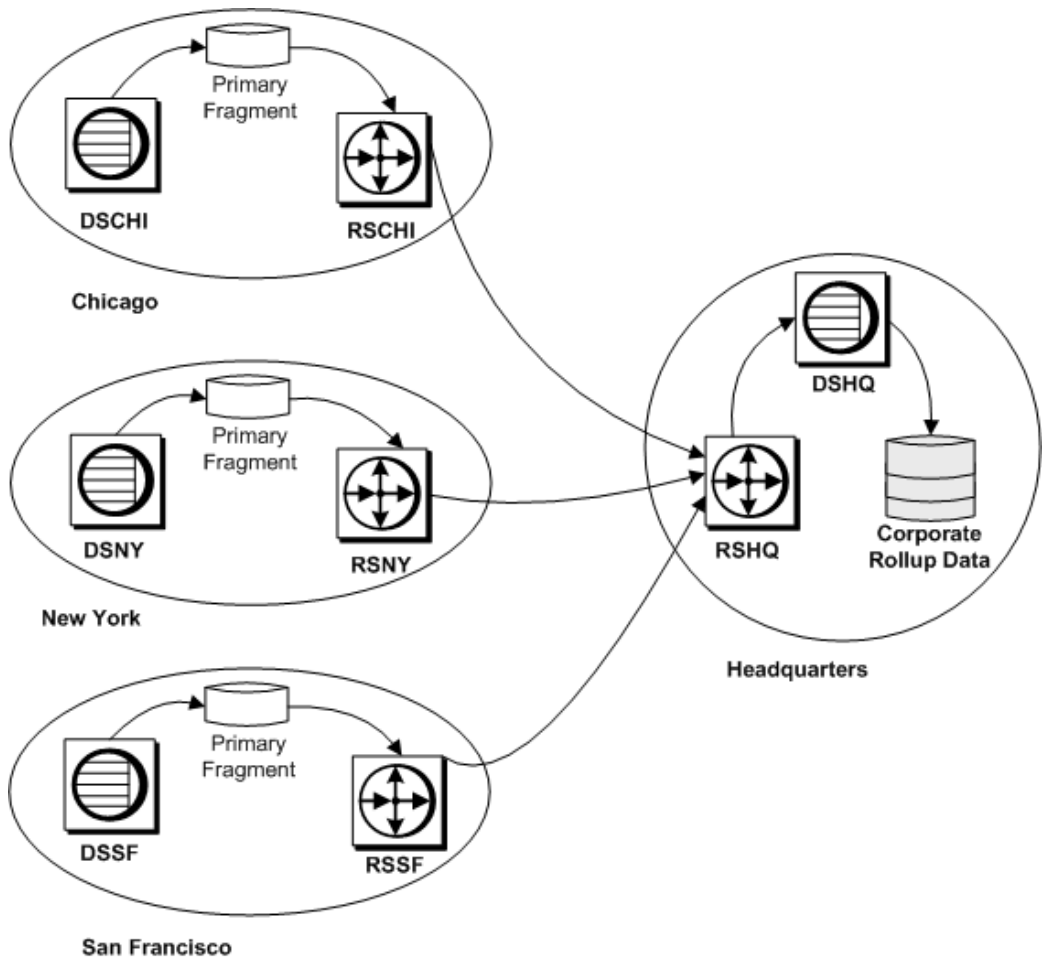
- Create a replication definition at any site where there is primary data, even if it is a “remote” site.
- Create subscriptions at each site for the replication definitions at the other sites. If  $n$  is the number of sites, create  $n-1$  subscriptions for each replication definition.

## **Corporate rollup**

The corporate rollup model has distributed primary fragments and a single, centralized consolidated replicate table. The table at each primary site contains only the data that is primary at that site. No data is replicated to these sites. The corporate rollup table is a “roll-up” of the data at the primary sites.

Figure 1-5 illustrates the flow of data for a corporate rollup application model:

Figure 1-5: Distributed primary fragments with corporate rollup



The corporate rollup model requires distinct replication definitions at each primary site. The site where the data is consolidated subscribes to the replication definition at each primary site.

To create a corporate rollup application from distributed primary fragments:

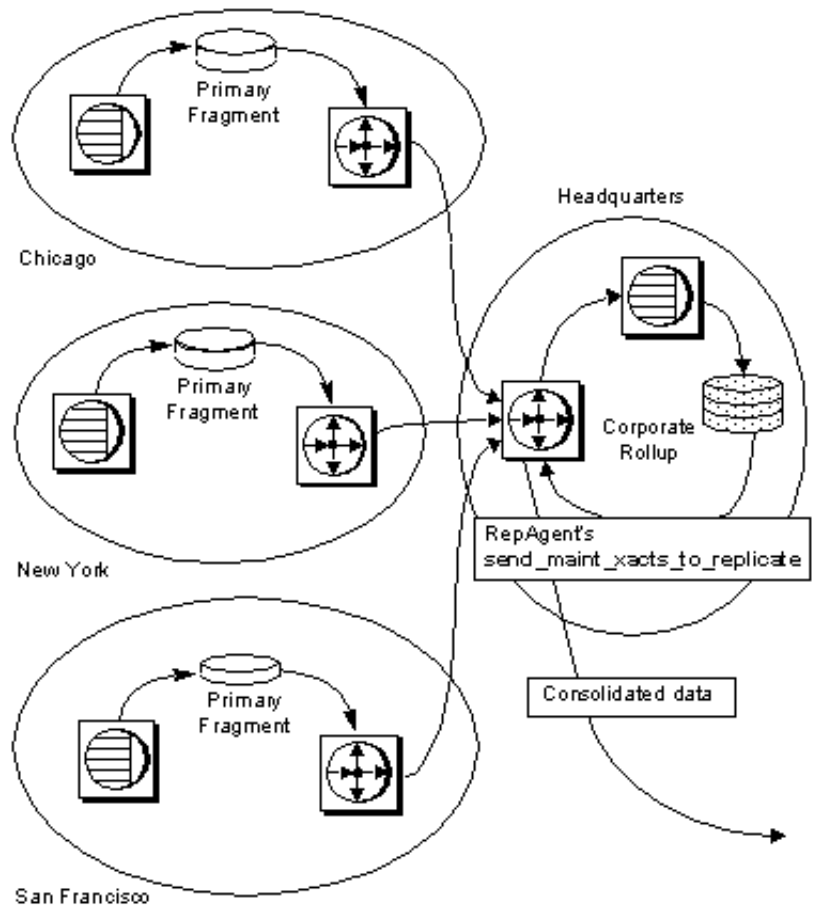
- Activate a Replication Agent at each primary site. However, you do not need to activate a Replication Agent at the central site, since data is not replicated from that site.
- Create tables in each primary database and in the database at the central site.
- Allow for replication on tables at each remote database where primary data is stored.
- Create replication definitions for tables at each remote site where primary data is stored.
- At the headquarters site, where the data is to be consolidated, create subscriptions for the replication definitions at the remote sites.

### **Redistributed corporate rollup**

The redistributed corporate rollup model is similar to the corporate rollup model. Primary fragments distributed at remote sites are rolled up into a consolidated table at a central site. At the site where the fragments are consolidated, however, a Replication Agent processes the consolidated table as if it were primary data. The data is then forwarded to Replication Server for distribution to subscribers.

Figure 1-6 illustrates the flow of data in an application based on the redistributed corporate rollup model:

Figure 1-6: Distributed fragments with redistributed corporate rollup



The consolidated table is described with a replication definition. Other sites can then subscribe to this table. Do not allow applications to update the corporate rollup table directly. All updates should originate from the primary sites.

The tasks associated with creating a redistributed corporate rollup replication system are identical to the corporate rollup model, except that:

- A Replication Agent must be activated at the headquarters site for the consolidated database so that all updates are submitted to the Replication Server as if they were made by a client application.

RepAgent must be configured with its `send_maint_xacts_to_replicate` option set to “true.” Otherwise, the Replication Agent filters will not redistribute replicated data as primary data.

For information about configuring RepAgent, see Chapter 4, “Managing a Replication System.”

- A Replication Agent is required for the headquarters Replication Server, since data will be redistributed from that site.
- At the headquarters site a replication definition must be created for each table. Other sites can create subscriptions to this replication definition, but the primary sites cannot subscribe to their own primary data.
- The headquarters Replication Server must have routes to the other sites that create subscriptions for the consolidated replicate table. If the primary sites create subscriptions, routes must be created to them from headquarters.
- Do not allow rollup sites to re-create subscriptions to their primary data. If they do, transactions could loop endlessly through the system.

## Replication Server and heterogeneous data servers

Replication Server supports heterogeneous data servers through an open interface. You can use any data-storage system as a data server if it supports a set of required basic data operations and transaction-processing directives.

Sybase Client/Server Interfaces (C/SI) include routines and protocols for client/server communication. Replication Server connects with data servers as a client using C/SI. If a data server does not support C/SI, you can create an Open Server™ gateway to allow Replication Server to access the data server or you can use a Sybase DirectConnect™ product, which provides access to other databases. When the data server returns results, the Open Server gateway can return them to the client using C/SI routines.

For detailed information about using Replication Server with databases from different vendors, see the *Replication Server Heterogeneous Replication Guide*.

Other open architecture components include:

- Replication Agents  
A Replication Agent detects modifications made to primary data and submits them to Replication Server for distribution to other databases.

The **RepAgent** thread in Adaptive Server is the Replication Agent for Adaptive Server databases.

Replication Agents for Microsoft SQL Server, SQL Anywhere™, DB2, Oracle, IMS, and VSAM databases are available from Sybase. If you use non-Adaptive Server data servers, you must provide a Replication agent for them. For details, see the *Replication Server Design Guide* and Sybase documentation for Replication Agents.

- Error classes and error processing actions

Error classes allow you to tailor your system to handle database errors for a type of data server. You can specify error actions in response to errors that a data server returns to Replication Server. Replication Server provides a default error class for Adaptive Server. See Chapter 6, “Handling Errors and Exceptions” in the *Replication Server Administration Guide Volume 2* for details.

- Functions, function strings, and function-string classes

Replication Server uses function strings to format replicated operations correctly for a type of destination database. To aid replication system administrators, Replication Server groups all the function strings for a particular type of database into a function-string class.

Replication Server provides default function-string classes for Adaptive Server, Oracle, Microsoft SQL Server, Adaptive Server Anywhere, IMS, VSAM, and DB2 databases. You can customize function strings to execute commands appropriate for your database and application. See Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2* for details.

## Warm standby applications

Warm standby applications are used to maintain a set of databases, one or more of which functions as standby copies of an active database. As clients update the active database, Replication Server copies transactions to the standby databases, maintaining consistency between them. Should the active database fail for any reason, you can switch to a standby database, making it the active database, and resume operations with little interruption.

Replication Server provides two methods for setting up a warm standby application. In both methods, the active and standby databases must be Adaptive Server databases. They can act as either a primary or replicate database with respect to other databases in the system.

- One method uses the multisite availability (MSA) feature to set up an active and one or more standby databases. See Chapter 12, “Managing Replicated Objects Using Multisite Availability,” for detailed information.
- The second method lets you set up an active and a single standby database, both of which must be managed by the same Replication Server. This warm standby application is considered a single logical unit in a Replication Server system. See Chapter 3, “Managing Warm Standby Applications” in the *Replication Server Administration Guide Volume 2* for detailed information.

## Mixed-version replication systems

A replication system can include Replication Servers or Adaptive Servers of different versions. Each program presents different issues.

You can use Replication Server version 11.5 and later with earlier versions of Replication Server. In earlier versions, all Replication Servers had to be at the same version before you could set the **system version** and enable certain features. This restriction has been relaxed for systems running Replication Server version 11.0.2 and later.

- When all Replication Servers are at least version 11.0.2 and the system version is set to 11.0.2, each Replication Server uses features according to its **site version**. Replication Servers running version 12.5 can use all 12.5 features among themselves, while Replication Servers running 11.0.2 can only use 11.0.2 features. Such a system is called a **mixed-version system**; each Replication Server can use all of its features.

See “Restrictions in mixed-version systems” on page 19 for more information.



- If the replication system includes Replication Servers prior to version 11.0.2, the system version number must be set to match the Replication Server with the earliest software version, for example 11.0.1 or 10.1.1. Certain new features that were introduced in later versions, including features of version 12.5, will not be available to any Replication Server. Such a replication system is not called a mixed-version system, because new feature use is restricted.

## Restrictions in mixed-version systems

Interaction between Replication Servers of different versions is restricted to the capabilities of the oldest version. Information associated with new features may not be available to Replication Servers of earlier versions.

See the documentation for each feature introduced in a new version, such as function-string inheritance or multiple replication definitions, for additional information about usage restrictions in mixed-version environments.

Refer to the installation and configuration guides and the release bulletin for your platform for more information about mixed-version systems and about setting the site version and system version.

## Mixed versions of Adaptive Server

You can use Replication Server version 15.0 or later with different versions of Adaptive Server. Although you can use data sources and destinations other than Adaptive Server, Replication Server requires either Adaptive Server or Adaptive Server Anywhere for warm standby databases and for Replication Server System Databases (RSSD).

---

**Note** Sybase does not support replication of Adaptive Server system databases, such as tempdb, model, sybssystemprocs, sybsecurity, and sybssystemdb. The replication of the Adaptive Server system database master is supported only if the Adaptive Server supports master database replication.

---

Some capabilities of Replication Server version 15.0.1 require you to use an Adaptive Server version 15.0.1 or later.

Refer to the installation and configuration guides and the release bulletin for your platform for more information about using Adaptive Server with Replication Server.

## Replication system security

Replication Server provides careful management of the login names, passwords, and permissions that are essential for system security. In addition, Replication Server supports third-party security mechanisms that safeguard data transmission across the network.

See Chapter 8, “Managing Replication Server Security” for more information about security.

## Replication Server security features

Replication Server enforces security using the following features:

- Replication Server login names  
Each Replication Server has its own set of login names, which are distinct from data server login names. This distinction gives the replication system administrator control over replicated data and other aspects of the replication system.
- Data server login names  
Data server login names are used with client applications to connect to data servers. Clients are generally given permission to update primary data. On replicate tables, however, clients are generally granted permission to select or view data, but are prohibited from making changes to data. These permissions are controlled in the data server, according to the application.
- Data server maintenance user login names  
Replication Server uses a special data server **maintenance user** login name for each local data server database that contains replicate tables. This allows Replication Server to maintain and update the replicate tables in the database.
- Password encryption  
You can encrypt passwords in sensitive areas of the replication system.
- Permission system  
Replication Server permissions are assigned to and cancelled from Replication Server login names using the `grant` and `revoke` commands.

See “Replication Server roles and responsibilities” on page 21 for more information about Replication Server and data server login names and roles.

## Network-based security features

Replication Server supports third-party, network-based security mechanisms that can:

- Establish unified logins to servers on the network

The security mechanism authenticates users at login. Each authenticated user is given a security credential that can be presented to remote servers as needed. As a result, users can seamlessly access different servers using a single login.

- Ensure secure data transmission across the network

A choice of different data protection services can:

- Encrypt and decrypt data transmissions
- Verify that a transmission has not been tampered with
- Verify the origin of each transmission
- Verify that a transmission has not been captured and re-sent
- Verify that transmissions are received in the order sent

See “Managing network-based security” on page 210 for more information about establishing network security.

## Replication Server roles and responsibilities

Administering the replication system is primarily the role of the replication system administrator. The database administrator plays a subsidiary role by supporting some Replication Server administration tasks. At some sites, role distinctions may not be clear-cut and some responsibilities can overlap. The following sections describe user roles and Replication Server tasks.

## Replication system administrator

The replication system administrator installs, configures, and administers the replication system. On a WAN, this role may be performed by different people at different locations. If this is the case, various tasks for administering Replication Server may require coordination between replication system administrators.

The replication system administrator has sa user permissions, which provide that person with the ability to execute nearly all commands in the replication system. In managing the system, the replication system administrator may need to coordinate with database administrators for both local and remote databases.

## Database administrator

The database administrator is responsible for:

- Administering local data servers, including login names and permissions.
- Managing data in a distributed database system. Various tasks may require coordination between Database Administrators for different databases.

## Replication Server tasks and responsibilities

Table 1-1 lists the tasks required to maintain the replication system.

**Table 1-1: Replication Server tasks and responsibilities**

<b>Task and reference</b>	<b>Roles</b>
Installing Replication Server. Refer to the Replication Server installation and configuration guides for your platform.	replication system administrator (RSA), database administrator (DBA)
Starting up and shutting down Replication Server. See Chapter 4, “Managing a Replication System.”	RSA
Quiescing Replication Server. See Chapter 4, “Managing a Replication System.”	RSA, DBA
Adding login names, database users, and administering appropriate permissions. See Chapter 8, “Managing Replication Server Security.”	RSA, DBA
Monitoring Replication Server. See Chapter 4, “Managing a Replication System.”	RSA
Configuring Replication Server. See Chapter 4, “Managing a Replication System.”	RSA

<b>Task and reference</b>	<b>Roles</b>
Adding replicated tables and stored procedures or changing table schemas. <ul style="list-style-type: none"> <li>• Creating and modifying replicated tables and stored procedures.</li> <li>• Creating and modifying table and function replication definitions.</li> <li>• Creating and materializing subscriptions at replicate sites.</li> </ul> See Chapter 9, “Managing Replicated Tables” Chapter 10, “Managing Replicated Functions” and Chapter 11, “Managing Subscriptions.”	RSA, DBA
Defining data server function-string classes and function strings. See Chapter 2, “Customizing Database Operations,” in the <i>Replication Server Administration Guide Volume 2</i> .	RSA, DBA
Maintaining routes. <ul style="list-style-type: none"> <li>• Creating and modifying routes.</li> </ul> See Chapter 6, “Managing Routes”	RSA
Maintaining and monitoring database connections. <ul style="list-style-type: none"> <li>• Suspending and resuming connections.</li> </ul> See Chapter 7, “Managing Database Connections.”	RSA
Creating a warm standby application. See Chapter 3, “Managing Warm Standby Applications” in the <i>Replication Server Administration Guide Volume 2</i> .	RSA, DBA
Localizing Replication Server. Refer to the <i>Replication Server Design Guide</i> .	RSA
Adding a primary or replicate database. See Chapter 7, “Managing Database Connections.”	RSA, DBA
Adding or removing a Replication Server. See Chapter 4, “Managing a Replication System.”	RSA
Processing rejected transactions. See Chapter 6, “Handling Errors and Exceptions” in the <i>Replication Server Administration Guide Volume 2</i> .	RSA, DBA
Administering local data server. <ul style="list-style-type: none"> <li>• Suspending or resuming data server.</li> </ul> See Adaptive Server or local server documentation.	DBA
Managing the RSSD. See Chapter 4, “Managing a Replication System.”	RSA, DBA
Managing Embedded Replication Server System Database (ERSSD). See Chapter 4, “Managing a Replication System.”	RSA, DBA
Creating, deleting, and modifying databases for replication. See Adaptive Server or local server documentation.	DBA
Setting up database user login names and passwords. See Adaptive Server or local server documentation.	DBA
Performing regular backups. See Chapter 1, “Verifying and Monitoring Replication Server” in the <i>Replication Server Administration Guide Volume 2</i> .	DBA
Applying database recovery procedures. See Chapter 7, “Replication System Recovery” in the <i>Replication Server Administration Guide Volume 2</i> .	RSA, DBA
Reconciling database inconsistencies. See Chapter 11, “Managing Subscriptions.”	RSA, DBA



# Replication Server Technical Overview

This chapter provides a technical overview of Replication Server and the replication system.

<b>Topic</b>	<b>Page</b>
Replication system components	25
Specifying data for replication	32
Establishing Replication Server connections	38
Specifying database operations	42
Transaction handling with Replication Server	43

This chapter introduces the components of a distributed database system based on Replication Server and illustrates the movement of transactions from a primary database to a replicate database. It identifies the aspects of Replication Server that play a role in receiving and distributing data at primary and replicate sites.

This chapter can aid in diagnosing and troubleshooting replication system problems.

## Replication system components

This section describes the components and resources that must be present or assembled before you can run Replication Server. Components in a Replication Server environment can include:

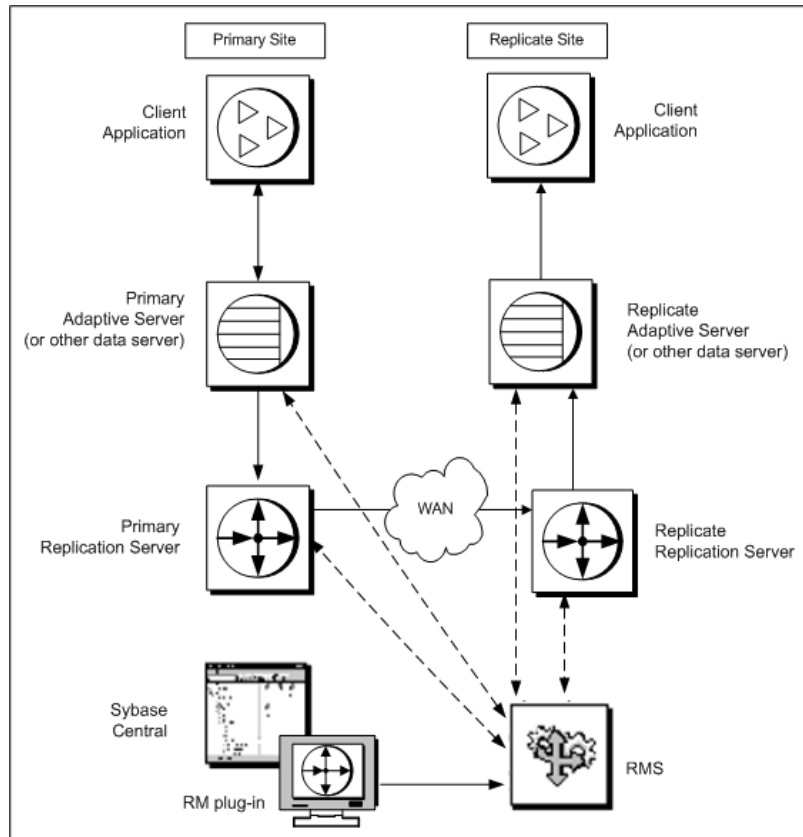
- Replication Server
- Adaptive Server or other data server
- Client applications
- Sybase Central
- Replication Manager (RM) plug-in to Sybase Central

- Replication Monitoring Services (RMS)

Each component uses the Open Client/Server™ Interface to communicate with other components.

Figure 2-1 illustrates a simple configuration for a WAN-based distributed database system based on Replication Server.

**Figure 2-1: Replication Server Domain**



## Replication Server

Replication Server coordinates data replication activities for local databases and exchanges data with Replication Servers that manage data at other sites. A Replication Server:



- Receives transactions from primary databases and distributes them to subscribing replicate databases
- Receives requests for data updates from a replicate database and applies them to a primary database

See “Replication Server internal processing” on page 123 in the *Replication Server Administration Guide Volume 2* for more information about the internal elements of the Replication Server.

## ID Server

The ID Server is a Replication Server that registers all Replication Servers and databases in the replication system. The ID Server must be running each time a:

- Replication Server is installed
- Route is created
- Database connection is created or dropped

Because of this requirement, *the ID Server is the first Replication Server that you start when you install a replication system.*

The ID Server must have a login name for Replication Servers to use when they connect to the ID Server. The login name is recorded in the configuration files of all Replication Servers in the replication system by the `rs_init` configuration program.

---

**Note** Once you have selected a login name for the ID Server, you cannot change to a different Replication Server. Sybase does not support any procedures that change the login name of the ID Server in the configuration files.

---

See Chapter 4, “Managing a Replication System” for information about the configuration file.

## Replication system domain

**Replication system domain** refers to all replication system components that use the same ID Server. You can set up multiple replication system domains, with the following restrictions:

- Replication Servers in different domains cannot exchange data. Each domain must be treated as a separate replication system with no cross-communication between them. You cannot create a route between Replication Servers in different domains.
- A database can be managed by only one Replication Server in one domain. Any given database is in one, and only one, ID Server's domain. This means that you cannot create multiple connections to the same database from different domains.

See “Adding a replication system domain” on page 89 for guidelines and restrictions on adding multiple system domains.

## Adaptive Server or other data server

Adaptive Server manages databases containing either primary or replicate data. Client applications use Adaptive Server to store and retrieve data and to process transactions and queries.

Each Replication Server requires an Adaptive Server database for its Replication Server System Database (RSSD) or an Adaptive Server Anywhere database for its Embedded Replication Server System Database (ERSSD), which contains the Replication Server system tables.

Replication Server also supports heterogeneous data servers through an open interface. You can use any system for storing data if it supports a set of required basic data operations and transaction processing directives. For data servers that contain primary databases, you must use a compatible Replication Agent program.

See the *Replication Server Heterogeneous Replication Guide* for details on heterogeneous data server support.

## Replication Agent

A Replication Agent notifies Replication Server of actions in a primary database that must be copied to other databases. The Replication Agent reads the database transaction log and transfers log records for replicated tables and stored procedures to the Replication Server managing the database, which distributes the modifications to databases that subscribe to the data.

A Replication Agent is needed for every database that contains primary data and for every database where stored procedures that need to be replicated are executed. A database that contains replicated data and no stored procedures marked for replication does not require a Replication Agent.

Replication Agents communicate with Replication Server by executing commands in Log Transfer Language (LTL).

Refer to Chapter 5, “Introduction to Replication Agents,” in the *Replication Server Design Guide* for more information about LTL commands.

### **Which Replication Agent for your system?**

The Replication Agent you use depends on the data servers in your replication system. Supported Replication Agents are:

- RepAgent – for Adaptive Server data servers. RepAgent, a thread in Adaptive Server, is the Replication Agent described in this book.
- Replication Agents for non-Sybase data servers:
  - SQL Anywhere
  - DB2
  - Oracle
  - IMS
  - VSAM

You also can create a Replication Agent to replicate data from a foreign data server. Refer to Chapter 5, “Introduction to Replication Agents,” in the *Replication Server Design Guide* for details.

### **Replication Server System Database (RSSD)**

The Replication Server System Database (RSSD) is a database that contains the Replication Server system tables. Each Replication Server requires an RSSD or an ERSSD, which holds the system tables for one Replication Server. The RSSD is managed by the Adaptive Server. The ERSSD is managed by Adaptive Server Anywhere.

### **System tables**

The Replication Server system tables are loaded into the RSSD during Replication Server installation. System tables hold information that Replication Server requires to send and receive replicated data, and include:

- Descriptions of replicated data and related information
- Security records for Replication Server users
- Routing information for other sites
- Access methods for the local databases
- Other administrative information

Refer to Chapter 8, “Replication Server System Tables,” in the *Replication Server Reference Manual* for a comprehensive list of system tables.

System table contents are modified during Replication Server activities, such as the execution of RCL commands or Sybase Central procedures. Only the replication system administrator, or members of the `rs_systabgroup` group, can alter the system tables.

To query the system tables and find status information:

- Use Sybase Central to view replication system details and properties.
- Use Replication Server system information or system administration commands. See “System Information Commands” and “System Administration Commands” in Chapter 1, “Introduction to the Replication Command Language,” in the *Replication Server Reference Manual*.
- Use Adaptive Server stored procedures to display information about the replication system. Refer to Chapter 5, “Adaptive Server Commands and System Procedures” in the *Replication Server Reference Manual*.

---

**Warning!** RSSD tables are for internal use by Replication Server only. You should never modify RSSD tables directly unless directed by Sybase Technical Support.

---

## RSSD and Replication Agent specifications

The RSSD is dedicated to the Replication Server that it supports; do not use it to store user data. However, a single data server may contain the RSSD and user databases. The database device space for the RSSD must be at least 20MB (10MB for data and 10MB for the log). It is best to put the database and the database log on separate devices.

A Replication Agent is needed for the RSSD if the Replication Server is the source for any route. If this is true, Replication Server distributes some of the information in its RSSD to other Replication Servers. See Chapter 6, “Managing Routes” for more information.

## Client applications

A client application is a program that accesses a data server. When the data server is Adaptive Server, applications can be programs created with Open Client Client-Library™ or DB-Library™, Embedded SQL™, or any other front-end development tool that is compatible with the Open Client/Server Interfaces such as PowerBuilder®. Open Client/Server includes routines and protocols for client/server communications.

In a simple replication system, clients update primary databases and Replication Server updates replicate databases. By replicating stored procedures, clients can update primary data from any replicate database.

## Sybase Central

Sybase Central is a graphical management tool for Sybase products. It implements the Sybase enterprise management strategy, which calls for a single management console, seamlessly integrated, across all server and middleware products. It connects to and manages Sybase products that are running on any Sybase-supported platform.

The Replication Manager is a plug-in to Sybase Central, which allows you to develop, manage, and monitor a replication environment.

With its easy-to-use interface, Replication Manager allows you to perform many administrative tasks that you would otherwise use RCL commands to perform, including:

- Creating, altering, and deleting Replication Server objects.
- Managing, monitoring, and troubleshooting replication system components.
- Monitoring the availability of servers and the state of connections and routes.
- Generating the RCL scripts for all Replication Server objects. Providing a script editor window that allows users to submit RCL or SQL to a server.
- Managing a replication domain, including configuration parameters for Replication Servers, Replication Agents, RepAgent threads, connections, and routes.
- Controlling the flow of data by suspending and resuming connections and routes.
- Displaying transactions in Replication Server stable queues.

- Displaying transactions in Replication Server exception log and allowing the user to edit and resubmit transactions.
- Managing a warm standby environment.

---

**Note** Sybase Central is available on Windows 2000, Windows 2003, and all UNIX platforms that Replication Server supports.

---

## Replication Manager (RM) plug-in to Sybase Central

Replication Manager is a management utility for developing, managing, and monitoring replication environments. It allows you to create replication objects such as connections, routes, replication definitions, and subscriptions.

## Replication Monitoring Services (RMS)

Replication Monitoring Services is a monitoring tool that you can use if your replication environment is fairly complex, involving ten or more servers. RMS allows you to monitor various servers and components in your environment, acting as a middle layer between the Replication Manager and the servers in the replication environment. RMS also provides the ability to control the flow of data and set the configuration parameters.

## Specifying data for replication

Replication Server uses the relational database model to represent data in tables that have a fixed number of columns and a varying number of rows. Each table you want to replicate must have one or more columns that can be used as a **primary key** to uniquely identify each row.

Replication Server lets you define the data and stored procedures that you want to replicate at remote databases, as well as letting you specify the destination databases themselves. As part of design and planning, you designate source and destination databases for your replication system and create the routes that replicated data follows from one Replication Server to another.

In general, a source database contains primary data and may be called the **primary database**, while a destination database contains replicate data and may be called the **replicate database**. Depending on your implementation, the same database may contain both primary and replicate data. Transactions or stored procedure executions are replicated from primary to replicate databases. Stored procedure executions may also be replicated from replicate to primary databases.

See “Replication Server basic primary copy model” on page 6 for details.

## Replication definitions and subscriptions for tables

You create one or more **replication definitions** to describe each primary (source) table. A replication definition lists a table’s columns and datatypes, the columns that make up the primary key, the columns that can be used in subscribing to the primary data, and specifies the location of the primary version of the table.

A replication definition may include additional settings to let you customize how you will use it. For example, you can create a replication definition just for replicating into a standby database in a warm standby application. Or, see Chapter 9, “Managing Replicated Tables” for more information.

You then create **subscriptions** for transactions on the data defined in the replication definition. A subscription instructs Replication Server to copy transactions for all rows or for qualifying rows only. Copies of a table can be limited to only the rows or columns needed.

Typically, creating a subscription causes Replication Server to copy the initial requested data from the primary database to the replicate database. This process is called **subscription materialization**. Once the subscription is created and materialized, Replication Server begins distributing database operations for the primary data as they occur. See Chapter 11, “Managing Subscriptions” for details.

## Replication definitions for database objects

Using multisite availability (MSA), you create a single **database replication definition** to describe the data to be sent to the replicate database. The database replication definition describes the database objects that are to be replicated. You can choose to replicate, or not replicate, individual tables, transactions, functions, system stored procedures, and DDL.

You then create a single **database subscription** at each subscribing database for the data described in the database replication definition. Database subscriptions cannot limit the data copied.

MSA provides a simple replication methodology that requires only one replication definition for the primary database and only one subscription for each subscribing database. If you want to transform the data, replicate minimal columns, or use primary keys to improve performance, you must add table and function replication definitions.

See Chapter 12, “Managing Replicated Objects Using Multisite Availability,” for more information.

## Replication definitions for stored procedures

For certain operations, replication of stored procedures may offer significant performance improvements over table replication. In addition, you can replicate stored procedures to update data from a replicate database to a primary database. A replication definition of a stored procedure is called a *function replication definition*.

## Benefits of replicated functions over normal replication

Adaptive Server logs a record for each row modified by a Transact-SQL command. When a single Transact-SQL command modifies multiple rows, Replication Server treats each log record received from the Replication Agent as a separate command in the transaction. For example, to replicate the results of a single update command that modifies 1000 rows in the primary database, Replication Server may execute 1000 update commands in each replicate database.

Commands that modify many rows can affect performance of replicate Adaptive Servers and the replication system. The volume of rows delivered through the replication system may use all available space in stable queues.

If an application updates multiple rows in a primary table, you can use replicated stored procedures to maintain data in destination databases. Because commands in stored procedures can modify multiple rows, using stored procedures allows you to update rows in replicate databases without passing images of the rows through the replication system. Only a single record reflecting stored-procedure execution and its parameters replicates through the system.



## Using replicated functions

A function replication definition describes a replicated stored procedure and includes:

- The parameters and datatypes
- The location of the primary data that the stored procedure may modify
- Parameters that can be used in subscribing to stored-procedure executions
- The name of the stored procedure to execute at the destination database

There are two types of **replicated function delivery**:

- **Applied** – executed at primary databases first and affect primary data. Replication Servers propagate the stored procedure and its parameters, applying data changes asynchronously at replicate sites that have subscriptions for an applied function replication definition. The maintenance user executes the applied function at the replicate sites.
- **Request** – executed at primary databases first and affect primary data. Replication Servers propagate the stored procedure and its parameters, applying data changes asynchronously at replicate sites that have subscriptions for a request function replication definition. The same user who executes the stored procedure at the primary databases executes the request function at the replicate sites.

Typically, the request function delivery is used to modify the remote data asynchronously at databases on other sites. The changes are replicated back to the originating site via either normal data replication or applied function delivery.

See Chapter 10, “Managing Replicated Functions” and Chapter 11, “Managing Subscriptions” for details.

## Publications

A **publication** lets you collect replication definitions for related tables and stored procedures and then subscribe to them as a group. You create publications at the primary Replication Server and subscribe to them at the destination Replication Server.

When you use publications, you create and manage these objects:

*Article* – identifies a replication definition, primary database, and publication. It may also limit the number of rows or parameters sent to the replicate database.

*Publication* – a collection of articles from a primary database.

*Publication subscription* – a subscription to a publication. When you create a publication subscription, Replication Server creates a subscription for each article in the publication.

Publications allow you to group replication definitions and subscriptions in a manner that makes sense for your system. It also allows you to create and check the status of only one subscription for a set of tables and procedures.

## Overview of replicating tables

This section summarizes how to replicate transaction data between a primary (source) and destination table. For more details, see “Marking tables for replication” on page 270 and “Subscription example” on page 382.

- *At the destination data server:* Create a copy of a table into which data will be replicated from the primary table. The copy may contain all or a subset of the columns from the primary table.

*At the primary Replication Server:* Create a replication definition to identify the table data you want to replicate. You can create one or more replication definitions per table that can be replicated into different destination databases. You can also create replication definitions for stored procedures. See Chapter 10, “Managing Replicated Functions” for details.

Once you have created a replication definition, transactions are available for replication to qualifying destination Replication Servers that subscribe to the replication definition.

You can create a set of articles that reference replication definitions and group them in a publication. If you want to limit the transactions sent to the destination database to those that affect certain rows, use a *where* clause in the article.

- *At the primary Adaptive Server:* Use the `sp_setreptable` system procedure to mark a table as replicated.

When you mark a table as replicated in the primary data server, the Replication Agent for the primary database can forward the table’s transactions to the primary Replication Server.

If you want to replicate text, unitext, or image columns, you may also need to use the `sp_setrepcol` system procedure.

If you use a different data source with a Replication Agent, refer to your Replication Agent documentation for information about marking primary objects for replication.

- *At destination Replication Servers:* Create a subscription for replication definitions that were created in primary Replication Servers. A subscription allows the destination table to receive the initial data from the primary (source) table through a process known as *materialization*, and to begin receiving subsequent replicated data updates.

You can create multiple subscriptions for each replication definition, but a replicate table can subscribe to only one replication definition. You can set up a subscription to receive all transactions for a destination table, or use a where clause to receive just the transactions that affect certain rows.

Create publication subscriptions for publications created at the primary Replication Server. When you do so, Replication Server creates an article subscription for each article in the publication.

Creating subscriptions completes the process of replicating data. See Chapter 11, “Managing Subscriptions” for details.

## Commands for managing replicated data

Refer to the following resources for detailed information about each command used to manage replicate data:

- Replication Manager plug-in help lists tasks and concepts for working with table replication definitions, function replication definitions, and subscriptions in Sybase Central.
- Table 9-1 on page 249 lists the Replication Server commands for working with table replication definitions.
- Table 10-1 on page 333 lists the Replication Server commands for working with function replication definitions.
- Table 9-3 on page 309 lists the Replication Server commands for working with publications.
- Table 11-3 on page 370 lists the Replication Server commands for working with subscriptions.

- Table 11-5 on page 396 lists the commands for working with publication subscriptions.

## Establishing Replication Server connections

Replication Server uses the Open Client/Server Interfaces to communicate between client applications and servers.

Server programs, including Replication Servers, Adaptive Servers, and gateway software for other data servers, are registered in a directory service—either an **interfaces file** or a Lightweight Directory Access Protocol (LDAP) server—so that client applications and other server programs can locate them.

---

**Note** If you are using network-based security, use the directory services of your network security mechanism to register Replication Servers, Adaptive Servers, and gateway software. Refer to the documentation that comes with your network-based security mechanism for details.

---

### Interfaces file

The interfaces file contains network definitions for servers in the replication system, including Replication Servers and data servers.

Generally, one interfaces file at each site contains entries for all local and remote Replication Servers and data servers. The entry for each server includes its unique name and the network information that other servers and client programs need to connect with it. The interfaces file at a site requires entries for these components:

- ID Server (if Replication Server is not also the ID Server)
- Replication Server
- RSSD Adaptive Server or ERSSD Adaptive Server Anywhere for this Replication Server
- ERSSD Replication Agent if a route is to be created from the current site
- Data servers with databases managed by this Replication Server
- Backup Server to back up Adaptive Server databases, including RSSDs

- Replication Servers at other sites that manage databases containing primary data that is replicated to this site
- Replication Servers at other sites with subscriptions for primary data maintained at this site
- Other Replication Servers to which this Replication Server has a route with no intermediate Replication Servers

You can use the default interfaces file or you can specify an alternative interfaces file at the command line when you start Replication Server. The interfaces file is usually located in the Sybase release directory. Use a text editor to modify the interfaces file. Refer to the Replication Server installation and configuration guides for your platform for more information.

## LDAP server

An LDAP server provides global directory services for sharing component information such as server names and connection properties. LDAP directory services allow components to look up directory information in a network-based system.

Any type of LDAP service or gateway is an LDAP server. An LDAP driver calls LDAP client libraries to establish connections to an LDAP server. The LDAP driver and client libraries define the communication protocol and content of messages exchanged between clients and servers. LDAP runs directly over the Transmission Control Protocol (TCP).

When the LDAP driver connects to the LDAP server, the server establishes the connection based on one of two authentication models:

- Anonymous access – which does not require any authentication information, and is used typically for read-only privileges, or
- User name and password access – which is different from the user name and password used to access Replication Server.

Replication Server uses the access information as an extension to the LDAP URL. Access information is taken from this file:

- `$SYBASE/$SYBASE_OCS/config/libtcl.cfg`
- `%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg` (Windows 2000, 2003)

Replication Server uses Open Client/Server libraries to connect to LDAP servers and Open Client/Server configurations and procedures to set up and maintain LDAP services. See the *Replication Server Configuration Guide* for your platform for directions on how to set up an LDAP directory. For detailed information about Open Client/Server LDAP support, see the *Open Client-Library/C Reference Manual*.

## Making Replication Server connections

To connect data servers and Replication Servers at the sites on a LAN or WAN, the replication system administrator at each site defines connections and routes.

Organizing connections and routes is fundamental in planning replication. The connections and routes you establish determine the number of Replication Server components you need. In addition, how you map replication between source and destination databases can impact system performance and data availability.

To specify where data is copied requires that you create the following paths or message streams between Replication Servers and between Replication Servers and databases in the system:

- A **connection** from a Replication Server to a database  
Replication Servers distribute transactions received from primary databases through connections to the replicate databases they manage. A Replication Server may have connections to several databases, but each database can have only one connection from a Replication Server.  
Warm standby applications also use a **logical connection**, which represents both a database and its standby database.
- A **route** from a Replication Server to another Replication Server  
From each source Replication Server that manages databases containing primary data, you must specify a route to each destination Replication Server that subscribes to the data.  
You can specify a direct route from a source Replication Server to a destination Replication Server, or an indirect route, with intermediate Replication Servers between the source and destination Replication Servers.

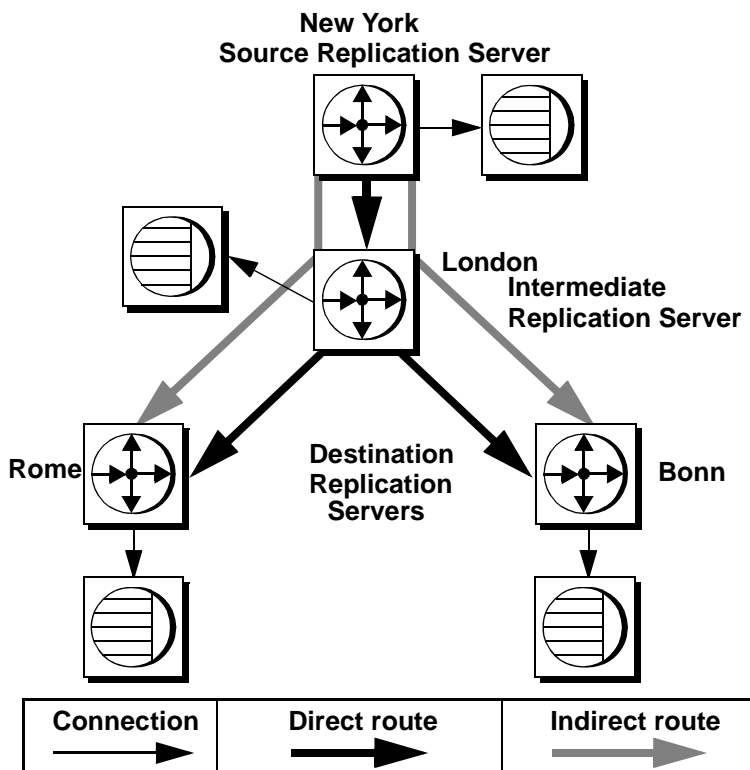
Figure 2-2 depicts an enterprise with several locations in Europe. A New York Replication Server routes all information for Europe through the London Replication Server. This arrangement reduces the number of direct connections the New York Replication Server makes and reduces WAN traffic. Data is sent once from New York to London, rather than from New York to each European location. The London Replication Server distributes the replicated data to the other European locations.

Refer to the *Replication Server Design Guide* for details and rules on designing routes and connections for a replication system.

See Chapter 6, “Managing Routes” and Chapter 7, “Managing Database Connections” for guidelines and procedures on when and how to create routes and connections.

See Chapter 3, “Managing Warm Standby Applications” in the *Replication Server Administration Guide Volume 2* for more information about logical connections.

**Figure 2-2: Routes and connections**



## Specifying database operations

Replication Server distributes database operations from a primary database to destination Replication Servers as *functions* that consist of a name and a set of data parameters. The destination Replication Server then uses **function strings** to map functions to the commands recognized by the destination data server. These commands represent transaction-control directives (begin transaction or commit transaction) or data-manipulation instructions (insert, update, or delete). The function string serves as a template or meta-command that transforms a function to a data-server-specific command. The use of function strings makes it possible for a primary site to replicate data to multiple heterogeneous data servers. Function strings are categorized into **function-string classes** according to data server type.

For example, a primary Replication Server transmits the `rs_insert` function to a destination Replication Server, which uses the appropriate function string to translate the function into the insert command specific for the data server in use at that site, whether the database is Adaptive Server, DB2, or another database.

There are two types of functions:

- *System functions* – represent data-server operations with function strings supplied by Replication Server or available when you install a new database to the replication system.
- *User-defined functions* – allow you to customize Replication Server applications to distribute stored procedures.

See Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2* for details.

## Function strings

Function strings for functions can be automatically generated for function-string classes that come with Replication Server. Function strings must be customized for any function-string class that the user creates that does not inherit its functions strings from one of the provided classes. To customize a function string, you modify an existing function string with data-server-specific commands or by invoking a remote procedure call (RPC). A customized function string can also contain **function string variables** that represent the values of columns, procedure parameters, system-defined information, and user-defined variables. Replication Server replaces the variables with actual values before sending function strings to the data server.



See Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2* for details.

## Function-string classes

A function-string class comprises all of the function strings used with a type of database. Replication Server provides three function-string classes: two for Adaptive Server and one for DB2. Although function strings may contain data-server-specific instructions, they can often be used with several databases maintained by the same data server type. You can create classes with all new function strings or create a **derived class** that inherits function strings from an existing **parent class**.

## Transaction handling with Replication Server

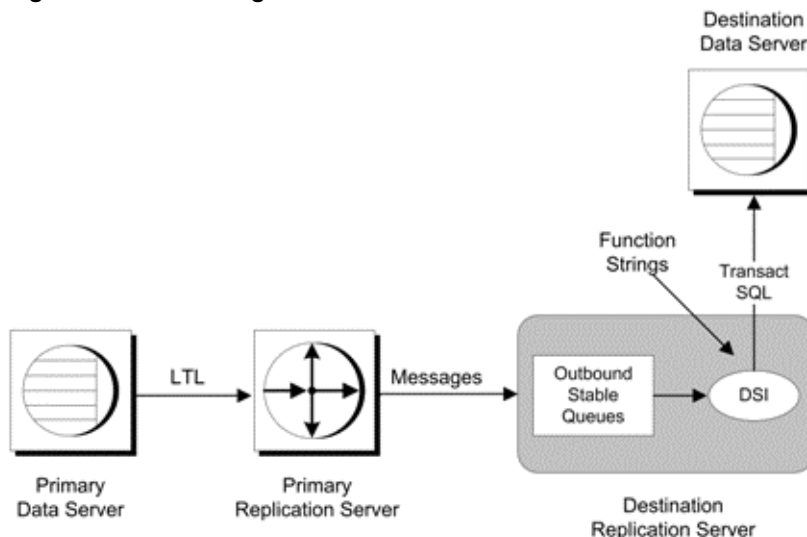
Replication Server depends on data servers to provide the transaction-processing services needed to protect stored data. To ensure the integrity of distributed data, data servers must comply with the following transaction-processing conventions:

- A transaction is one unit of work. Either all operations in the transaction are performed or none are performed.
- Transaction results are permanent. A transaction cannot be arbitrarily undone after it is committed.

Replication Server copies committed transactions from primary sites to destination sites. It distributes transactions in the order they are committed so that copied data passes through the same states as the primary (source) data.

Figure 2-3 illustrates Replication Server method for translating transactions.

Figure 2-3: Translating transactions



Once the primary Replication Server sends transactions to subscribing sites, destination Replication Servers store the transactions in the outbound Data Server Interface (DSI) stable queue.

## Stable queues

When you install Replication Server, you set up a disk partition that Replication Server uses to establish **stable queues**. During replication operations, Replication Server temporarily stores updates in these queues. You can add more partitions later if your replication system requires more space for stable queues.

There are three types of stable queues, each of which stores a different type of data:

- **Inbound queue** – holds messages only from a Replication Agent. If the database you add contains primary data, or if request stored procedures are to be executed in the database for asynchronous delivery, Replication Server creates an inbound queue and prepares to accept messages from a Replication Agent for the database.
- **Outbound queue** – holds messages for a replicate database or a replicate Replication Server. There is one outbound queue for each of these destinations:

- For each replicate database managed by a Replication Server, there is a Data Server Interface (DSI) outbound queue.
- For every Replication Server to which a Replication Server has a route, there is a Replication Server Interface (RSI) outbound queue.
- Subscription materialization queue – holds messages related to newly created or dropped subscriptions. This queue stores a valid transactional “snapshot” from the primary database during subscription materialization or from a replicate database during dematerialization.

See “Partitions for stable queues” on page 47 for physical queue requirements.

See the *Replication Server Troubleshooting Guide* for information on how to examine queue contents for troubleshooting purposes.

## Queue management

Each queue is managed by a Stable Queue Manager (SQM) thread. Threads are subprocesses that manage specific tasks, such as receiving messages. Some queues also have an additional Stable Queue Transaction (SQT) thread. See “Processing in the primary Replication Server” on page 124 in the *Replication Server Administration Guide Volume 2* for details on the SQM and SQT threads.

When transactions are ready to leave the stable queue, one of these threads submits the transactions in the queue:

- **Data Server Interface (DSI)** thread – manages the connection with the data server.
- **Replication Server Interface (RSI)** thread – manages the connection with the replicate Replication Server.

The enhanced queue dump commands, gives you flexibility in identifying the stable queues, controlling the stable queue contents to dump, and supporting additional output file options. Replication Server also introduces commands that allow you to delete and restore specific transactions from the SQM.

These are the commands that you can use for managing stable queues:

- `sysadmin dump_queue`
- `sysadmin sqt_dump_queue`
- `resume connection`
- `sysadmin log_first_tran`

- `sysadmin sqm_zap_tran`
- `sysadmin sqm_unzap_tran`
- `sysadmin dump_tran`

See the *Replication Server Reference Manual* for detailed information about these commands.

## DSI thread

The DSI thread translates the transaction modifications into RPCs or the language as specified by the function strings in the function-string class assigned to the destination database.

Replication Server starts DSI threads to submit transactions to a replicate database to which it has a connection.

The DSI thread performs the following tasks:

- Collects small transactions into groups by commit order.
- Maps functions to function strings according to the function-string class assigned to the database connection.
- Executes the transactions in the replicate database.
- Takes action on any errors returned by the data server; depending on the assigned error actions, also records any failed transactions in the exceptions log.

To improve performance in sending transactions from a Replication Server to a replicate database, you can configure a database connection so that transactions are applied using multiple DSI threads. See “Using parallel DSI threads” on page 151 in the *Replication Server Administration Guide Volume 2* for a description of this feature.

The DSI thread may apply a mixture of transactions from all data sources supported by the Replication Server. The transactions are processed in the single outbound stable queue for the destination data server.

## RSI thread

RSI threads send messages from one Replication Server to another. There is one RSI thread for each destination Replication Server.

The primary Replication Server processes transactions, causing those destined for other Replication Servers to be written to RSI outbound queues. An RSI thread logs in to each destination Replication Server and transfers messages from the stable queue to the destination Replication Server.

When a direct route is created from one Replication Server to another, an RSI thread in the source Replication Server logs in to the destination Replication Server. When an indirect route is created, Replication Server does not create a new stable queue and RSI thread. Messages for indirect routes are handled by the RSI thread for the direct route. For more information, see “Establishing Replication Server connections” on page 38.

## Partitions for stable queues

Replication Server stores messages destined for data servers or other sites on partitions. It allocates the space in partitions to stable queues and operates in 1MB chunks called segments. Each stable queue holds messages to be delivered to another Replication Server or to a database.

The `rs_init` program assigns the initial partition to the Replication Server. Refer to the Replication Server installation and configuration guides for more information about working with partitions in `rs_init`.

The minimum initial partition is 20MB. You may need additional partitions, depending on the number of databases the Replication Server manages and the number of remote sites to which the Replication Server distributes messages. Larger partitions may also be necessary when subscriptions are initiated or when there are long-running transactions.

A Replication Server can have any number of partitions of varying sizes. The sum of the partition sizes is the Replication Server capacity for queued transactions.

Use the `create partition` command to assign additional partitions. See the *Replication Server Reference Manual* for details.

When choosing a partition for Replication Server, consider these guidelines:

- Replication Server partitions should be operating system raw partitions.
- Do not mount the partition for use by the operating system.
- Do not use the partition for any other purpose, such as storing file systems, maintaining swap space, or locating Adaptive Server devices.

- Allocate the entire partition to Replication Server. If you allocate just a portion of a partition for Replication Server, you cannot use the remainder for any other purpose.
- Do not allow any users read/write permissions on the partition unless the user is going to start Replication Server.

You can choose how Replication Server allocates queue segments to partitions or you can use the default mechanism. The default mechanism assigns queue segments to the next partition in an ordered list. Use the `alter connection` or `alter route` command to choose a different allocation mechanism. See “Allocating queue segments” on page 182 in the *Replication Server Administration Guide Volume 2* for more information.

## Using disk files for stable queues

Partitions can be either raw disk partitions, which is preferable, or operating system files. Where a choice is available, raw disk partitions provide the best recoverability, since disk writes to raw disk partitions are not buffered by the operating system.

To use a disk file for a partition, create the file before you execute the `create partition` command. You can create an empty file and set its permissions so that Replication Server can read and write to the file. Replication Server extends the file to the size you specify.

## Distributed concurrency control

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 primary version of a table, the primary Replication Server does not distribute the update to other sites.

When a transaction succeeds in updating primary data, the Replication Server distributes the changes. Unless a failure occurs, the update succeeds at all sites with subscriptions to the data.

## Transactions that modify data in multiple databases

A transaction that modifies primary data in more than one data server may require additional concurrency control. According to the transaction processing requirements, either all of the operations in the transaction must be performed, or none of them. If a transaction fails on one data server, it must be rolled back on all other data servers updated in the transaction.

If a multi-database transaction is replicated, updates to each database flow to replicate databases as independent transactions because there is one Replication Agent per database.

## Failed replicate table updates

A modification to primary data may fail to update a copy of the data at a subscribing site. The primary version is the “official” copy and updates that succeed there are expected to succeed at subscribing sites with copies.

If the updates do not succeed, one of the following reasons may explain why:

- Replicate and primary versions are out of sync following a system recovery and a loss has been detected.

See Chapter 7, “Replication System Recovery” in the *Replication Server Administration Guide Volume 2* for more information.

- The data server storing the copy of the table has constraints that are not enforced by the data server storing the primary version.
- The data server storing the copy of the table rejects the transaction due to a system failure, such as lack of space in the database or a full transaction log.

When a transaction fails, Replication Server records the transaction in an exceptions log for handling that is appropriate to the application. Replication Server offers error handling flexibility through its error action feature. This feature allows responses to data server errors based on your own defined configuration settings. For example, you can specify that transactions be retried at the site where they failed.

A client at each site must resolve transactions in the exceptions log, because the appropriate resolution is application-dependent. In some cases, you can automate the resolution by encapsulating the logic for handling rejected transactions in an intelligent application program.

## Transaction processing by the Replication Agent

The Replication Agent scans the database transaction log and sends transaction information to the Replication Server for distribution to subscribing databases.

This section describes transaction processing by Adaptive Server RepAgent thread. Other Replication Agents may work differently.

## Coordinating Adaptive Server log truncation

As long as there is space in the Adaptive Server database transaction log, Adaptive Server continues to process transactions. To prevent the log from filling up, it must be emptied (“truncated”) periodically. You can use the Adaptive Server dump transaction command or set the Adaptive Server trunc log on chkpt option to “on” so that the log truncates automatically.

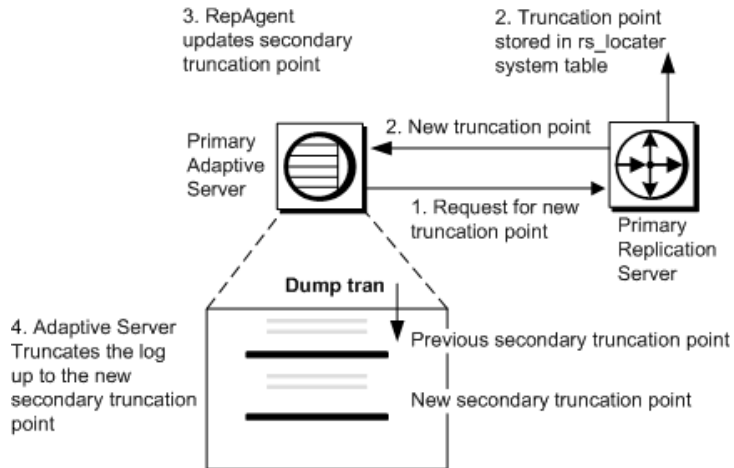
Each primary database maintains primary and **secondary truncation points** in its database log. The primary truncation point marks the last log record Adaptive Server has finished processing. The secondary truncation point normally marks the log record that contains the begin transaction command for the oldest open transaction not yet fully applied by Replication Server. Replication Server stores a copy of the latest secondary truncation point in the rs\_locator table of the RSSD.

RepAgent requests a new secondary truncation point when it has scanned a predetermined number (batch) of records or has reached the end of the log and there is no new activity. Replication Server acknowledges receipt of a batch of transaction records by giving RepAgent the information that allows it to move the secondary transaction point.

Adaptive Server makes sure that only transactions already processed and passed to the Replication Server are deleted by never truncating the log past the secondary truncation point.

RepAgent updates the secondary truncation point as shown in Figure 2-4.



**Figure 2-4: Adaptive Server log truncation**

- 1 RepAgent requests a new secondary truncation point from the primary Replication Server.
- 2 The primary Replication Server returns the latest secondary truncation point to the RepAgent and also writes it into the rs\_locator system table.
- 3 RepAgent updates the secondary truncation point in the transaction log.
- 4 At the next checkpoint or dump transaction command, the log is truncated up to the new secondary truncation point.

**Schema** information describes the structure of the database. Each time you change the schema of a database object—such as dropping a table, creating a clustered index, or renaming a column—Adaptive Server records current schema information for that object. Thus, when RepAgent scans the transaction log, it can always retrieve the correct schema for a table or procedure—even if the original database object has been changed or no longer exists. You do not need to drain the transaction log before executing schema changes at the primary site.



# Managing Replication Server with Sybase Central

This chapter describes how you can manage your replication environment using the Replication Manager (RM) plug-in and Replication Monitoring Services (RMS). Sybase integrates its systems management tools into one desktop product, called Sybase Central. Each server product, such as Replication Server or Adaptive Server, can be managed from Sybase Central.

Topic	Page
Using Replication Manager from Sybase Central	53
Setting up a replication environment	64
Monitoring a replication environment using RMS	75

See “Replication system components” on page 25, for a detailed description of Replication Manager and Replication Monitoring Services.

## Using Replication Manager from Sybase Central

This section describes how to use the Replication Manager plug-in with Sybase Central.

### Starting and stopping Sybase Central

This section explains how to start and stop Sybase Central.

#### Starting Sybase Central

On Windows

Start Sybase Central using any of the standard methods for your Windows operating system, such as:

- Select Start | Programs | Sybase | Sybase Central v6.0.

- Create a shortcut on your desktop for Sybase Central.
- Double-click *scjview.exe* in the to *%SYBASE%\Shared\Sybase Central 6.0.0/win32* and double-click .
- Add Sybase Central to your Startup program group.

On UNIX

Start Sybase Central by going to *\$SYBASE/shared/sybccentral600* and executing *scjview.sh*.

## Stopping Sybase Central

To stop Sybase Central on Windows or UNIX, select File | Exit.

## Getting started

After installing and configuring Replication Server:

- 1 Start Sybase Central.
- 2 Use RM to create a new replication environment that includes all data servers, Replication Servers, and Replication Agents that participate in the replication.

Replication Manager displays a two-pane window that contains icons for the servers managed in the environment. Use this window to monitor the status of the servers and to execute menu commands to diagnose and manage the servers and other components of your replication system.

## Using online help

The Replication Manager plug-in provides online help that presents information on a broad range of help topics. The instructions for performing specific tasks using the Replication Manager through Sybase Central are detailed in the online help.

There are two types of help in the Replication Manager plug-in:

- Topic help
- Tooltips and status bar messages

## Topic help

Topic help describes how to use Sybase Central to manage your replication system. To display topic-level help, select Help from the Sybase Central main menu, then select Replication Manager Online Help.

The Replication Manager help browser opens with two panes. The left pane displays the table of contents, and the right pane displays contents of the selected topic.

Click the Contents tab to browse through the topics by category.

- *Book icons* represent headings. Double-click a book icon to see the sub entries under that heading. Sub entries can be other book icons or page icons.

Topic headings are organized around Replication Server concepts (for example, Managing Users, Managing Database Connections, and others) for easy reference.

- *Page icons* represent topics that describe tasks, or concepts that correspond to the heading under which they are listed. Topics are generally organized in the order that you would perform procedures under that heading. Double-click a page icon to display a topic.

## Tooltips

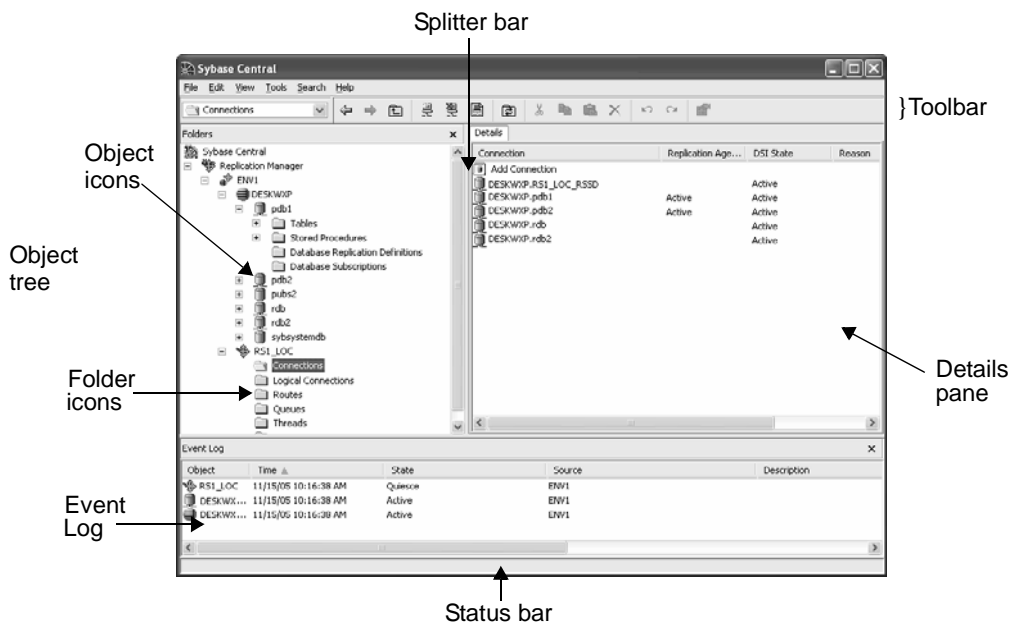
Tooltips are small pop-up windows that provide a description of a control (that is, a toolbar button or menu option) when a pointer is moved over that control.

## Using the Replication Manager GUI

The Replication Manager user interface displays within the Sybase Central framework. The main window allows you to access replication environment and server objects.

Figure 3-1 shows Replication Manager in the Sybase Central main window.

**Figure 3-1: Replication Manager in Sybase Central main window**



The main window is divided into left and right panes. When you are connected to a replication environment:

- The left pane displays a hierarchical list, or *object tree*, which shows:
  - Icons for folders and objects in the replication environment
  - Icons for other plug-ins, such as the Sybase Central plug-in for Adaptive Server Enterprise, if they are installed
- The right pane displays the contents of the folder or object selected in the left pane.

To adjust the size of the panes, use the mouse pointer to drag the splitter bar to the left or right.

## Selecting folder and object icons

The main window includes folder icons and object icons:

- Each *folder icon* contains all objects of its type within the replication environment. For example, the Connections folder shows all connections. Folder icons can appear in either the left or right pane.

---

**Note** A Replication Server connection is associated with one database in an Adaptive Server.

---

- Each *object icon* represents one server object, such as a database, a table, a replication definition, or a connection. Some objects contain other objects, such as a database containing tables.

To select an object, click its icon. The type of object selected determines the range of commands available. For most activities, you must select an object before you can perform any operation on it.

## Using the Details list

When you select an object in the left pane, one or more tabs appear in the right pane. For most objects, a single tab called “Details” is displayed, containing a list of general information about the object.

The Details list displays:

- Sub components, which are replication or database objects that are contained in another object.
- Function components, which invoke a wizard when double-clicked.

---

**Note** When you select a thread object, several tabs appear in the right pane.

---

## Moving through the Sybase Central object tree

To see different parts of the object tree, use the following techniques:

- To move vertically through the current display, use the scroll bar on the left or right pane.
- To expand or collapse the list to show different levels of detail, do one of the following:
  - Click the plus or minus buttons. A plus button next to an icon indicates that the list of objects for that icon can be expanded. A minus button indicates that the list of objects for the icon is fully expanded.

- Double-click a folder icon, which expands the list in the right pane and changes the view to a list of objects in the folder. For most objects, double-clicking an object icon in the right pane opens a property sheet that displays information.

## Customizing the display

To hide the folders, toolbar, status bar, or event log, select Folders, Toolbar Status Bar, or Event Log from the View menu. To redisplay, repeat the procedure.

To move the display of the right pane tabs from top to bottom or from left to right, select Tools | Options.

## Using keyboard shortcuts

In addition to using a mouse, you can use keyboard shortcuts to choose menu commands and navigate through dialog boxes.

Every menu title and menu command has an underlined letter, called a *mnemonic*. To select a menu, press **Alt+mnemonic**. To choose a menu command, press the mnemonic key. You can execute some commands directly by pressing **Ctrl** plus another key, or by pressing a function key. These shortcuts are listed on the menus.

To navigate to different *controls* (for example, fields, lists, and buttons) in a dialog box or property sheet, use the **Tab** key. To select different tabs in a property sheet, use the **Tab** key to select the current tab, then use the left and right arrow keys to select other tabs.

## Using menus and toolbars

This section describes the Sybase Central menus and toolbar.

### Context menu

To activate a *context menu*, right-click an object icon. From the menu that appears, choose the appropriate command.

A context menu is specific to the selected object and contains commands that are executed against the selected objects. You can execute some commands against multiple objects at once.



### **Toolbar and status bar**

The Sybase Central toolbar provides quick alternatives for executing frequently used menu commands. The status bar provides information about the selected menu item.

To display or hide the toolbar or status bar, toggle the **Toolbar** or **Status Bar** command on the **View** menu.

#### **Toolbar**

The toolbar has the following controls:

- A drop-down list box that displays the hierarchy of the currently selected object. You can select an object higher up the hierarchy to change the focus of the main window.
- Buttons, which provide a quick way to execute menu commands.

#### **Status bar**

The status bar is an information display bar located at the bottom of the application window. In Sybase Central, the status bar displays a brief description of the menu command at which the cursor is currently pointed. The help line appears on the left side of the status bar.

### **Viewing events in the log pane**

The Replication Manager displays the Event Log pane, which shows events that occur in the replication environment. These events can be:

- Component state changes for connections, routes, and queues
- Server availability changes
- Background thread completion
- Background processes status
- RMS event trigger execution

### **Background processing**

Several tasks performed by the Replication Manager can be very time-consuming, such as creating a subscription that also materializes the table. These tasks are performed in the background, allowing Replication Manager to perform other tasks. When you start a time-consuming task, Replication Manager displays a message window to indicate a running process. Click **Stop Process** to cancel the background process or click **Close** to close the window, to continue running the process in the background.

When a background task is completed, Replication Manager puts an event entry in the event log.

To see the status of a background process at a later time, open the Background Processes dialog, which displays a list of all the currently running processes.

To access the Background Processes dialog, select Search | Background Processes. The Background Processes dialog opens, displaying the following:

- Process – the name of the process.
- Start time – the start time of the process.
- Status – the status of the process.

## Using script editors

Replication Manager provides two script editors; the Replication Command Language (RCL) script editor and the Structured Query Language (SQL) script editor. Both editors operate in the same way, except the RCL script editor highlights the RCL keywords while the SQL script editor highlights the SQL keywords.

You can use the script editors to view generated RCL commands, which include syntax to create any objects such as connection, routes, and replication definitions.

### ❖ Accessing the script editor

- 1 Select the Replication Server object for which you want to generate RCL.
- 2 Right-click that object.
- 3 Select Generate RCL from the context menu. The selected script editor window opens and contains the RCL needed to create the selected object.

## Monitoring of status

The status of an environment is the state of its components. The status of a server or component includes its current state and a list of reasons that describe the state.

Replication Manager graphically displays the status of the servers and components in the environment. It shows an object icon, which changes depending on the state of an object. The status of the servers, connections, routes, and queues also displays in the Properties dialog.

As you work, information in open dialog boxes and the Sybase Central window may become unsynchronized. To update the contents of the main window, select View | Refresh Folder or Refresh All, or press F5.

## Using Hide options for connection status

You can hide (or filter out) the connection status if you do not want to see the it either on the individual connection icon or as part of the rollup status for Replication Server.

The options for hiding connection status are as follows:

- Hide the State of the Replication Agent — hides the state of the Replication Agent thread in the Details list, on the Connection Properties dialog box, and in the rollup status for the Replication Server to which that Replication Agent thread is connected.
- Hide the State of the DSI Thread — hides the state of the DSI thread in the Details list, on the Connection Properties dialog box, and in the rollup status for the Replication Server to which the DSI thread is associated.

### ❖ Hiding connection status

- 1 Right-click the connection for which to hide the status.
- 2 Select Hide Connection Status from the drop-down menu.

A dialog box displays which shows options for hiding the connection status.

- 3 Select an option.

The state for the connection now reads “Hidden.” The state on the Connection Properties dialog box and in the rollup status for the Replication Server is also hidden. The Event Log records this change.

## Filtering connection status different instances of Replication Manager

The filtering state of a connection status is stored locally by the Replication Manager, therefore, different instances of Replication Manager do not share filtering states. For example, if you create a connection using one instance of Replication Manager, and then set the Replication Status to hide for that connection, another Sybase Central plug-in instance monitoring the same environment does not filter the connection status; filtering information is available only to the original Replication Manager instance.

In addition, any connection created outside of Sybase Central (by `rs_init` or from the command line) is not filtered automatically by the Replication Manager. You must set the filtering manually from within Sybase Central.

### Filtering connection status in warm standby environments

If you are creating a warm standby environment, the Replication Manager automatically sets the filtering state for the active Data Server Interface (DSI) thread and standby RepAgent thread connections. You must set filtering for the physical connection manually by selecting one of the connection status hide options from the context menu.

### Performing common tasks

This section describes how to do these tasks, which are common to most Sybase Central objects:

- Create an object
- View an object's properties
- Delete an object

### Creating an object

You can create new replication definitions, subscriptions, connections, and other Replication Server objects in Sybase Central.

To create an object:

- 1 Select the folder for the type of object you want to create.
- 2 Select File | New.
- 3 From the cascading menu, choose the object name.

One of the following occurs:

- If a wizard exists to help you create the object, the wizard opens. Respond to the wizard prompts.
- If no wizard exists, a property sheet displays. Fill in the information for the new object.

### Viewing an object's properties

After you create an object, it is represented by an icon in any pane of the Sybase Central window. You can display or update the object by opening its Properties dialog.

A Properties dialog contains information about the object and how it relates to other objects in the replication environment. The Properties dialog can also provide a direct navigation path to its related objects. It enables you to enter information for a new object on tabbed pages.

The Properties dialog box generally has three tabs.

- General tab — displays all status.
- Communications tab — displays information on how Replication Manager communicates with the server.
- Parameters tab — displays and allows modification to configuration parameters for servers and components.

---

**Note** Some Properties dialogs may have different tabs. For example, a connection has General, Security, and Parameters tabs.

---

To view an object's properties:

- 1 Select the object icon.
- 2 Select File | Properties.

### **Deleting an object**

To delete an object:

- 1 Select the object icon.
- 2 Select Edit | Delete.
- 3 Confirm the deletion in the confirmation dialog.

### **Naming data servers in RM**

Data servers in RM must have unique names, and the names of the non-Sybase data servers must match the Replication Agent configuration parameter `rs_source_ds`. If an existing environment uses the same name for the Replication Agent and the configuration parameter, change the name of the agent by manually adding the server name, host, and port number in page 3 of the Add Server wizard.

## Setting up a replication environment

A replication environment includes replication objects such as Replication Servers, data servers, and Replication Agents. Before any replication activities can be performed, you must create and configure an environment.

Depending upon the scale and complexity of your replication environment, you can set up either a two-tier or a three-tier solution for your environment.

### Two-tier management solution

In a two-tier management solution, Replication Manager (RM) connects directly to the servers in the environment without communicating through a management layer.

This lets you manage small, simple replication environments with fewer than 10 servers, and provides you the ability to create, alter, and delete components in the replication environment.

### Three-tier management solution

In a three-tier management solution, Replication Manager can monitor larger and complex replication environments with the help of Replication Monitoring Services (RMS). RM connects to the servers in the environment through RMS.

RMS provides the monitoring capabilities for the replication environment. In this solution, RMS monitors the status of the servers and other components in the replication environment, and RM provides the client interface that displays the information provided by the RMS.

## Preparing for a two-tier solution

To prepare for a two-tier solution:

- 1 Install the Replication Server and Sybase Central software. See the release bulletin and *Replication Server Installation Guide* for your platform.
- 2 Identify the data servers to be used in your replication system. If the data servers are not yet installed, do so using the installation guide for your specific data server.
- 3 Use `rs_init` to configure the Replication Server. See the *Replication Server Configuration Guide* for your platform.
- 4 Start Sybase Central. See “Starting Sybase Central” on page 53.

- 5 In Sybase Central, create a replication environment and add the data servers and Replication Servers.

---

**Note** Replication Manager does not require a Sybase interfaces file, but you have the option to use it.

---

## Creating an environment

Creating the replication environment involves giving a name to your environment, creating an environment object, setting up permissions, and adding servers.

### ❖ Creating a replication environment object

- 1 In Sybase Central, click the Replication Manager icon in the left pane of the Replication Manager window. The Add Replication Environment icon displays on the Details pane.
- 2 Double-click the Add Replication Environment icon in the right pane.
- 3 Enter the name of your environment and click Next.
- 4 Enter a user name and password that will let you access the environment. Click Next.
- 5 From the list of servers, select the ones to add to your environment, then add a user name and password for each. If you are adding a Replication Server, enter a user name and password for the RSSD. Click Next.

When adding servers, you must provide a user name and password that have been granted certain permissions:

- Replication Server – sa permissions.
- Adaptive Server Enterprise – the sa\_role and the sso\_role.
- Replication Server RSSD – the database owner.

---

**Note** You can either select a server from the list or enter a server name, host, and port number. This list is from the interfaces file found in the \$SYBASE directory.

---

- 6 Check the summary page to make sure you have added all the servers you need. Then click Finish.

---

**Note** You are not required to add all servers when you create the environment. You can add new servers to an existing environment by using the Add Server wizard.

---

The new environment object displays in the left pane under the Replication Manager object with the name you assigned.

---

**Note** If you update the *sql.ini* or interfaces file while Sybase Central is running, you need only restart the wizard, or reopen any dialog box in progress. You need not restart Sybase Central for the changes to take effect.

---

❖ **Dropping a server from a replication environment in Sybase Central**

- 1 Select the server you want to drop.
- 2 Do one of the following:
  - Click the Delete icon from the toolbar.
  - Right-click the selected server, select Delete.

---

**Note** Although Sybase Central removes the server from the replication environment's server list and removes the server icon from the environment, the server is not removed from your replication system. If there are routes or database connections still associated with a deleted server, the server name may still appear in dialog boxes.

---

## Connecting to and disconnecting from a replication environment

The Replication Manager saves the environment information so that you do not have to re-create it when you restart Sybase Central.

❖ **Connecting to an existing replication environment**

- 1 Select the environment to which you want to connect.
- 2 In the login dialog box, enter your user name and password.



3 Click OK. This allows you to start managing the environment.

You can be connected to more than one environment or RMS domain at a time.

❖ **Disconnecting from a replication environment**

1 Select the environment from which you want to disconnect.

2 Select Tools | Disconnect. The Details view displays the state of the environment you disconnected from.

## Setting up a replication environment using Replication Manager

The Replication Manager provides a wizard to help you quickly set up one of several types of working replication environments, including:

- warm standby environment
- An environment consisting of one primary and multiple replicates
- A bidirectional replication environment

After you have created the replication environment, use the Configure Replication wizard to create connections, database replication definitions, and subscriptions for your replication tasks.

❖ **Configuring replication**

1 Select the environment object you created.

2 In the right pane, double-click Configure Replication. The Configure Replication wizard appears.

3 In the Configure Replication wizard, select the type of environment you want to create:

- Standard Replication Server warm standby environment. To configure, go to “Configuring a standard warm standby environment” on page 68.
- An environment where the primary database is replicated to several replicate sites. To configure, go to “Configuring an environment with one primary and multiple replicates” on page 69.
- Bidirectional replication environment. To configure, go to “Creating a bidirectional replication environment” on page 70.

❖ **Configuring a standard warm standby environment**

---

**Note** If the active server you want to configure does not appear in the list, click Add Server to open the Add Server wizard. Follow the procedure to add servers in “Creating a replication environment object” on page 65.

---

- 1 After selecting the type of environment to create from the Configure Replication wizard, click Next.
- 2 Select the active server and active database.
- 3 Select the Replication Server that will manage the database connections.
- 4 Select the standby server and standby database.
- 5 Enter the name of the logical connection.

When using an existing connection to create a warm standby logical connection, you must use the existing data server and database names of the active database for the logical connection name. See the *Replication Server Administration Guide Volume 2* for more information.

- 6 Enter the user name and password of the maintenance user. If the maintenance user does not exist, the wizard creates one for you. Accept the defaults, or enter your own values.
- 7 Select the user name and password that the RepAgent will use to connect to the Replication Server. If the RepAgent user does not exist, the wizard creates one for you. Accept the defaults, or enter your own values.
- 8 Select the materialization method.
- 9 Review the summary information about the replication environment.
- 10 If everything looks correct, click Finish. Otherwise, click Back to return to an earlier window and change the replication environment information. Then, return to the final wizard window and click Finish.

Replication Manager creates the following replication objects:

- Logical connections
- Physical connections
- Maintenance user in both active and standby Adaptive Server Enterprise servers.

**❖ Configuring an environment with one primary and multiple replicates**

Use this procedure to set up a warm standby environment where data is replicated from one site to many replicate sites using multisite availability (MSA).

---

**Note** If the active server you want to configure does not appear in the list, click Add Server to open the Add Server wizard. Follow the procedure to add servers in “Creating a replication environment object” on page 65.

---

- 1 Select the primary server and the primary database. Click Next.
- 2 Select the Replication Server that will manage the database connections. Click Next.
- 3 Select a replicate server, then a corresponding database, and click Add.  
The corresponding *server.database* connection displays in the Connections list.  
Repeat this step for each replicate server and replicate database pair you need in your environment. Click Next.
- 4 Enter the user name and password of the maintenance user. If the maintenance user does not exist, the wizard creates one for you. Accept the defaults, or enter your own values. Click Next.  
All selected connections will use this maintenance user login.
- 5 Select the user name and password that the RepAgent will use to connect to Replication Server. If the RepAgent user does not exist, the wizard creates one for you, assigning a default name and password. Accept the defaults, or enter your own values. Click Next.  
All connections will use this RepAgent login.
- 6 Specify how replicated tables will materialize:
  - Create Subscription Without Materialization – use this method if the primary data is already loaded at the replicate and updates are not in progress.
  - Define Subscription for Bulk Materialization – in this method, a subscription is initialized by a user-specified mechanism outside the replication system.

- 7 If you chose Define Subscription for Bulk Materialization, click Use Dump Marker in the Transaction Log to use dump and load coordination. Click Next.
- 8 Click Finish if everything looks correct in the information summary of replication environment. Otherwise, click Back to return to an earlier window and change the replication environment information. Then, return to the final wizard window and click Finish.

At the end of the configuration, Replication Manager creates the following replication objects:

- Physical connections
- A database replication definition for the primary database
- One or more database subscriptions for each of the replicate databases
- Maintenance user in Adaptive Server Enterprise servers

❖ **Creating a bidirectional replication environment**

Use this procedure to define an environment where data is updated at multiple locations and replicated on each site.

- 1 Identify the servers and databases that will be part of the bidirectional replication environment.
- 2 Follow steps 3 through 8 in configuring an environment with one primary and multiple replicates.

## Managing Replication Server objects

The Replication Server objects include connections, replication definitions, subscriptions, and queues.

For non-Sybase data servers, RM uses DirectConnect to communicate with the data servers and to act as an interface for RM. The status of DirectConnect is reflected in the status of the non-Sybase data server.

---

**Note** RM does not support database replication definition, database subscription, and creation of logical connection for non-Sybase data servers.

---

## Connections

Connections go from a database to a Replication Server, or from a Replication Server to a database. Replication Servers distribute transactions received from primary databases through connections to the replicate databases they manage.

### ❖ **Creating a connection**

- 1 Select the Connection folder from the Sybase Central object tree.
- 2 Double-click the Add Connection icon on the Details pane. The Add Database Connection wizard opens. Click Next.
- 3 Select an active server and database from the drop-down list. Click Next.
- 4 Enter a user name or accept the default value.
- 5 Enter a password. Then, click Next.
- 6 Select from the given options. Click Next.
- 7 Click Finish after checking the summary of information.
- 8 If you created a replicate connection through DirectConnect to a non-Sybase data server, manually execute the script that generates the tables and procedures required for replication.

## Replication definitions

A replication definition describes the source table to Replication Server, specifying the columns you want to copy. It may also describe attributes of the destination table. Destination tables that match the specified characteristics can subscribe to the replication definition.

Replication Server provides replication at the database, table, and stored procedure levels. RM allows you to create a replication definition for a database, a table, or a stored procedure. A replication definition for a stored procedure is called a “function replication definition”. You can create, edit, and delete function replication definitions and function subscriptions.

This procedure describes how to create a replication definition for a database.

### ❖ **Creating a replication definition on the primary database**

- 1 In the object tree, double-click the database where you want to create a replication definition. The Database Replication Definitions folder displays.
- 2 Double-click the Database Replication Definitions folder. The Add New Database Replication Definition window appears.

- 3 In the General tab, enter a replication definition name.

---

**Note** You can specify other replication definition settings on the other given tabs.

---

- 4 Click Replicate all DDL if you want the DDL that is executed at the primary database to be replicated to the replicate database.
- 5 Click OK.

## Subscriptions

A subscription is created on the replicate database to subscribe to a specific replication definition. It identifies the primary database, which contains the data to be replicated.

You can create a subscription for any type of replication definition: databases, tables, and stored procedures.

### ❖ Creating a subscription for a database replication definition

- 1 In the object tree, double-click the database where you want to create a subscription.
- 2 Double-click the Database Subscriptions folder.
- 3 In the Details pane, double-click the Add Subscription icon.
- 4 Enter the name of the subscription.
- 5 Under Primary, select the Connection and the database replication definition that you want to subscribe to.
- 6 Select a materialization method from the drop-down list (optional).
- 7 Specify whether to Subscribe to truncate table (optional).
- 8 Click OK.

## Queues

Data that is passed between servers (Adaptive Server, Replication Server, and so on) is stored in stable queues within Replication Server. Replication Manager displays the statistics of queue usage and the content of the queues.

### Using the View Queue Data dialog box

The View Queue Data dialog box lets you filter and sort the data from a queue as an aid in troubleshooting transactions in the queue. You can also edit, delete, or undelete a given command, or purge the first transaction in the queue.

The View Queue Data dialog box contains these options:

- Filter fields, which let you select the type of filters that RM uses to display data from the queue. These filters include:
  - Column
  - Column value
  - Segment
  - Number of blocks displayed
  - Number of rows displayed
  - Whether to start at the first segment
  - Whether to include all data to the end of the segment
  - Whether to include all rows
  - Whether to show deleted data
  - Whether to view all data to the end of the queue
- General buttons, which let you:
  - Display the queue data with the current filters
  - Close the dialog box
  - Purge the first transaction from the queue
  - Edit transactions
  - Delete transactions
  - Undelete transactions
  - Group transactions, which returns the Queue Data scrolling list display back to grouped transactions

- Queue Data scrolling list, which contains rows of data from the current queue. Each column contains specific information about the command and transaction contained in each row. For example, to sort the queue data by a specific column, select the column name. The Queue Data scrolling list refreshes, sorting the data according to that column. An arrow displays next to the column name to show that you have sorted the data by that column. The columns you can sort by include:
  - Segment
  - Transaction Name
  - Command
  - Origin Site
  - Origin Commit Time
  - Origin User
  - Transaction ID
  - Origin QID

---

**Note** You can delete, undelete, or purge queue transactions only when Replication Server is in standalone mode.

---

❖ **Viewing queue data**

- 1 In the object tree, click the Queues folder. Queues display in the Details pane.
- 2 In the Details pane, right-click the queue whose data you want to view.
- 3 Select View Data from the context menu. The View Data dialog box opens.
- 4 To filter data shown, select one of the filter fields. See “Using the View Queue Data dialog box” on page 73 for more information.
- 5 To sort the data, select segment, transaction, origin, size, status, commit time, or user.



## Monitoring a replication environment using RMS

To monitor a replication environment, the Replication Manager connects to the servers in the environment through the Replication Monitoring Services—a middle-management layer that provides monitoring capabilities for replication environment. RMS is an optional component of a replication system. It is used to monitor large or complex environments. It also provides the ability to control the flow of data and set the configuration parameters.

In a three-tier solution, you set up an RMS server to help you monitor your replication environment. In this solution, RMS monitors the health and availability of the servers and other components in your replication environment.

### Preparing for a three-tier solution

When you are creating a three-tier environment, you must connect to an RMS server. In this environment, you must edit the interfaces file (UNIX) or the *sql.ini* file (Windows) with the host name, port number, and server name. You can edit this file using a text editor or the *dsedit* utility. You can use the same interfaces file that the other servers in the replication environment use.

RM does not need an entry in the interfaces file for RMS. You can provide the host name and port number of RMS directly to RM. The servers that are managed by RMS must be in the RMS's interfaces file.

See the *Replication Server Configuration Guide* for your platform for information about configuring RMS.

### Connecting to RMS

To connect to RMS:

- 1 Click the Connect icon from the toolbar; The Connect to a Replication Domain window opens.
- 2 Select RMS Server.
- 3 Enter the user name and password needed to connect to RMS.
- 4 Select RMS from the list of servers in the drop-down list or click the Options button to provide the connection information for the RMS.
- 5 Enter a server name, host, and port number.

6 Click OK. The RMS server is added to your object tree.

See the *Replication Server Configuration Guide* for your platform for information about starting/stopping RMS.

## Adding and dropping servers through RMS

Servers in a three-tier environment are added and dropped in the same way as in the two-tier environment. The difference is in the properties that display if you select the object and view its properties. RMS is designed to monitor servers and components, thus you can only view properties that RMS uses to monitor and troubleshoot the replication environment.

## Viewing managed objects

In the object tree, double-click or expand the RMS folder to view replication objects managed by RMS. Under RMS, you can still see the connections, routes, queues, and threads. When you select a replication object such as the Routes folder, you can view the list of created routes. You can manage these replication objects using the Replication Manager.

Viewing objects in the Replication Manager for RMS is exactly the same as viewing objects in a two-tier environment.

## Adding event triggers

Replication Monitoring Services is designed to monitor the replication environment. When something happens in your environment, the server and component status changes. These changes are displayed in the event log. You can use RMS to create event triggers to monitor these changes.

Event triggers notify you when some event occurs in the replication environment. RMS executes a script when the specified event occurs. For example, a user can set up a script to request an e-mail message when a connection becomes suspended. This capability lets you specify a method of notification when an event occurs. You can create an event trigger for any server or component that the RMS monitors.

### ❖ Creating an event trigger for a Replication Server

1 In the object tree, select the Replication Server.

- 2 On the right side of the desktop, select the Event Log pane.
- 3 Double-click the Add Server Event Trigger icon.
- 4 Select the status change that will trigger the event.
- 5 Enter a “Wait before executing” value (optional). This causes RMS to wait for the event to change before executing the trigger.
- 6 Select “Execute at each interval” (optional). This causes RMS to execute the trigger at each monitoring interval instead of just once.
- 7 Enter the name of the script for RMS to execute when the event occurs.
- 8 Click OK. The new event displays in the Event Log pane.



# Managing a Replication System

This chapter tells you how to perform various Replication Server operations, including starting and shutting down Replication Server, and monitoring, maintaining, and configuring the replication system.

Topic	Page
Setting up a replication system	79
Performing Replication Server tasks	82
Starting Replication Server	86
Shutting down Replication Server	87
Adding a Replication Server	88
Adding a replication system domain	89
Setting Replication Server configuration parameters	90
Managing the RSSD	95
Managing Embedded Replication Server System Database	97
Quiescing Replication Server	104
Removing a Replication Server	105

## Setting up a replication system

This section lists the basic steps in setting up a replication system. This process requires planning and careful attention to the replication needs of your organization. If you are new to Replication Server, refer first to the *Replication Server Design Guide* for information that can help you plan your replication system.

You can perform some of these steps in `rs_init`, Replication Server configuration utility, which allows you to configure Replication Servers and add databases to your system. You can use Sybase Central to perform most of the tasks listed here, including adding databases, creating replication definitions, and creating subscriptions.

### Installing the software

Install your Sybase software according to the Replication Server installation guide for your platform.

### Configuring the replication system

After you install Replication Server, use the `rs_init` utility program to start and configure the replication system and to add databases.

Refer to the *Replication Server Configuration Guide* for more information about `rs_init`.

## Creating connections and routes

To replicate data from one database into another, you must first establish the routes and connections that allow Replication Server to move the data from its source to its destination.

- Create connections

When you use Sybase Central or `rs_init` to add a database to your replication system, the program creates the connection for you. You never have to create a connection using the command-line option `create connection` unless you are replicating data into a database that is not an Adaptive Server database.

Refer to the *Replication Server Configuration Guide* for information about using `rs_init`.

Refer to Chapter 7, “Managing Database Connections” for detailed information about connections.

- Create routes

Create routes using Sybase Central or the `create route` command at the source Replication Server.

Refer to Chapter 6, “Managing Routes” for more information about how to create routes.

## Setting permissions and security

Set up login names, passwords, and permissions to establish Replication Server security for the replication system. Replication Server login names and specific permissions are required for:

- Users who are setting up replicated data or monitoring and managing the Replication Server. You can create these users in Sybase Central or at the command line.

See “Managing Replication Server user security” on page 199 for information about creating users at the command line.

- Components of the replication system, such as the data server and the Replication Server. You can create system users in `rs_init` or at the command line.

Refer to the installation and configuration guides for your platform for information about `rs_init`. See “Managing Replication Server system security” on page 192 for information about creating system users at the command line.

If network-based security is enabled at your site, you can set up secure pathways and choose message protection options for Replication Server to Replication Server communications. See “Managing network-based security” on page 210 for detailed information about setting up network-based security.

## Verifying the replication system

You must verify that the entire replication system is working before you create replication definitions or subscriptions or perform system diagnostics.

See “Verifying a replication system” on page 2 of *Replication Server Administration Guide Volume 2* for a detailed description of verifying the replication system.

## Creating replication definitions

To set up a table for replication, mark it as replicated in Adaptive Server and define a replication definition for it in Replication Server. The replication definition describes the table and contains information about the columns to be replicated.

- If you plan to replicate stored procedures, create the stored procedure in both the primary and replicate database.
- If you are replicating the procedure from the primary to replicate database, mark the stored procedure for replication in the primary database.
- If you are replicating the procedure from the replicate to the primary database, mark the stored procedure for replication in the replicate database.

Create a function replication definition for the stored procedure at the primary Replication Server, even if you are replicating the stored procedure from a replicate database to the primary database.

Refer to Chapter 9, “Managing Replicated Tables” for more information about creating replication definitions. Refer to Chapter 10, “Managing Replicated Functions” for more information about replicated function delivery.

## Creating subscriptions

A subscription instructs Replication Server to copy data from primary tables to specified replicate databases. If you create a replication definition for a table at Replication Server, you must create a subscription for that table replication definition at the replicate database. Similarly, if you create a function replication definition for a stored procedure, you must create a subscription for that function replication definition at the replicate database. However, you do not need to create subscriptions for table or function replication definitions that update primary databases.

Refer to Chapter 11, “Managing Subscriptions” for more information.

## Performing Replication Server tasks

This section describes several tools that you use when interacting with Replication Server.

`rs_init` allows you to set up a new Replication Server and add new databases to the system.

You execute RCL commands by connecting to Replication Server using a client application. You can use a utility program such as Sybase Central or `isql`, or you can use a custom application program that you create with Open Client Client-Library.

Sybase Central Replication Manager plug-in component provides a graphical user interface for performing many of the tasks associated with managing a Replication Server system.

Since many of the commands described in this book are used on an as-needed basis, `isql` is a convenient way to execute them.



RCL commands are similar to Transact-SQL commands. Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax for all RCL commands.

## Using *rs\_init*

Use the *rs\_init* utility to configure a new Replication Server and to add databases to your replication system. If you have an existing Replication Server, you can use *rs\_init* to upgrade to a new version or downgrade to a previous version. *rs\_init* is installed with the Sybase software. You can use it interactively or with a resource file.

Refer to the *Replication Server Configuration Guide* for your platform for complete instructions on using *rs\_init*.

## Managing Replication Server with Sybase Central

Sybase Central is a Replication Server system management tool. It provides a graphical user interface that allows you to monitor the components of the replication system and perform Replication Server tasks.

With Sybase Central, you can view a graphical representation of the topology of the replication system, which allows you to group objects and view status information. Sybase Central also provides menus for performing tasks and monitoring objects.

With Sybase Central, you can:

- Perform many of the tasks available from the Replication Server command line and *isql*, often more quickly than using equivalent Transact-SQL or RCL commands. For example, you can manage users, create routes and connections, create replication definitions and subscriptions, and manage warm standby applications.
- Display multiple Replication Server connections and selectively view the contents of queues.

Refer to Chapter 3, “Managing Replication Server with Sybase Central” for information about navigating Sybase Central.

## Using *isql*

You can use the *isql* utility to execute:

- ERSSD (Embedded Replication Server System Database) commands, using the primary user name and password from the Replication Server configuration file.
- RCL commands interactively
- Scripts stored in text files

For simple operations, using *isql* interactively may be easiest.

For more complex operations, Sybase recommends using *isql* to execute scripts, so you can keep a record of the RCL commands you have executed to set up a Replication Server. You can edit scripts and resubmit them whenever necessary. Scripts are also useful when you are verifying a new system or investigating the cause of a failure.

You can use *isql* to log in to Replication Server or Adaptive Server. This section describes both the interactive and script methods for using *isql* with Replication Server. For information about using *isql* with Adaptive Server, refer to the Adaptive Server utility programs manual for your operating system.

### Using *isql* interactively

To use *isql* interactively:

- 1 If necessary, start the Replication Server, as described in “Starting Replication Server” on page 86.
- 2 Log in to the Replication Server using the following command:

```
isql -User_name -Ppassword -Sserver_name
```

Specify the name of the Replication Server using the *-S* flag.

If your login is accepted, *isql* displays a prompt:

```
1>
```

- 3 Enter the RCL command you want to execute.

When you press the Return key at the end of a line, *isql* increases the line number. Some commands require more than one line.

- 4 To execute the command, enter “go” (on a line by itself, with no blanks) and press Return.

To cancel the command, enter “reset” and press Return. The prompt’s line number is reset to 1.

Some RCL commands display immediate results. Others execute asynchronously, that is, they return a system prompt without necessarily having completed the desired action and report only syntax errors.

- 5 To exit isql, enter “quit” at the beginning of a line.

---

**Note** You can check the status of asynchronous commands by executing RCL commands that display status or by querying the RSSD system tables at the affected sites. Refer to Chapter 8, “Replication Server System Tables,” in the *Replication Server Reference Manual* for more information on system tables and the stored procedures you can use to query them.

---

## Using *isql* to execute scripts

You can create scripts of RCL commands and execute them using *isql*. This procedure is useful when you need to execute the same set of commands in Replication Servers at multiple sites.

To create and execute a script for *isql*:

- 1 As necessary, start the Replication Server, as described in “Starting Replication Server” on page 86.
- 2 Create a text file for your script, and enter into it the RCL commands you want to execute. As with the interactive method, separate each command with the word “go” on a line by itself.
- 3 Execute the script using the following *isql* syntax:

```
isql -User_name -Ppassword -Sserver_name  
      -iscript_name
```

The *isql* utility displays the results from the script’s commands on your screen (standard output). Or, you can redirect the output to a file:

```
isql -User_name -Ppassword -Sserver_name  
      -iscript_name > output_file
```

## Starting Replication Server

Normally, you need to restart Replication Server only if you are reconfiguring system files or if your system experienced a failure that brought down Replication Server. Initially, the installation process starts the replication system for you.

To bring up a Replication Server site, start system components in this sequence:

- 1 Start the data server containing the databases that Replication Server manages.
- 2 If Replication Server uses Adaptive Server Enterprise for the RSSD, start the RSSD. For more details, see the *Replication Server Installation Guide* for your platform.
- 3 Start Replication Server by running the `repsvr` command on UNIX systems, or the `repsrvr.exe` command on Windows 2000 or 2003 systems, or by executing the Replication Server run file.

See “Replication Server executable program” on page 86.

- 4 Start RepAgent for the data server and for the RSSD if RepAgent has not been configured to start automatically at server startup.
- 5 To ensure that Replication Server started with no errors:
  - Check the `repsvr.log` file for error messages (indicated with the letter “E” on the left), as described in “Replication Server error log” on page 204 in the *Replication Server Administration Guide Volume 2*.
  - Use `isql` to log in to each Replication Server, or use a script that logs in to each server. See “Verifying server status” on page 4 in the *Replication Server Administration Guide Volume 2*.

## Replication Server executable program

You use the `repsvr` or `repsrvr.exe` command at the operating system prompt to run the Replication Server program.

For example, to run `repsvr`, log in to the operating system as the “sybase” user, and execute `repsvr` using the following syntax:

```
repsvr [-C config_file] [-i id_server] [-S rs_name]  
      [-l interfaces_file] [-E errorlog_file] [-M] [-v]  
      [-K keytab_file]
```

Refer to Chapter 7, “Executable Programs,” in the *Replication Server Reference Manual* for complete information about each of the parameters of the `repsvr` command.

The `rs_init` program creates the run file “`RUN_name`,” where *name* is the name of the Replication Server. The run file specifies the `repsvr` command with parameters set for the installed Replication Server. Normally, you start Replication Server by executing the run file.

The Replication Server executable program and the Replication Server run file are located in the *bin* subdirectory of the Sybase release directory. Refer to your platform’s Replication Server installation and configuration guides for more information.

## Replication Server configuration file

Replication Server finds the startup information it needs in a configuration file. The file is created by the `rs_init` program, but it can be edited with a text editor. If it contains encrypted passwords, however, you must modify them using `rs_init`. Refer to your platform’s Replication Server installation and configuration guides for more information. The default name for the Replication Server configuration file is the Replication Server name with “.cfg” appended.

## Shutting down Replication Server

To shut down a Replication Server, log in to it and enter this command at the `isql` prompt:

```
shutdown
```

When you shut down a Replication Server, it refuses additional connections, terminates threads, and exits.

## Adding a Replication Server

The first Replication Server you install must be the ID Server. It must be running when you install new Replication Servers or add databases to the replication system.

To add a Replication Server to a replication system, use the `rs_init` program, as described in your platform's installation and configuration guides. Always conduct a careful review and analysis of how the additional Replication Server will fit into your system. Determine the other processes that are required for the server and designate required names and accounts for these processes.

See “Creating an environment” on page 65 for instructions on adding Replication Server in Sybase Central.

When you install each Replication Server, `rs_init` performs the following tasks:

- Creates a configuration file for the Replication Server
- Creates an executable run file to start the Replication Server
- Sets RepAgent parameters at Adaptive Server
- Creates and initializes the RSSD or the ERSSD.
- Starts the Replication Server and RepAgent for the RSSD, as necessary

After you have executed `rs_init` for each Replication Server you are adding:

- 1 Determine the routing for the Replication Server, and modify the routes in the existing system to accommodate the new Replication Server.

See Chapter 6, “Managing Routes” for details.

- 2 If you want to add a new database, prepare that database for replication.

See Chapter 7, “Managing Database Connections” for details.

- 3 Grant users the appropriate permissions for Replication Server commands.

See Chapter 8, “Managing Replication Server Security” for details.

- 4 If applicable, add replication definitions, subscriptions, function-string classes, and error classes for the Replication Server.

See Chapter 9, “Managing Replicated Tables” and Chapter 11, “Managing Subscriptions”. See also Chapter 2, “Customizing Database Operations” and Chapter 6, “Handling Errors and Exceptions” in the *Replication Server Administration Guide Volume 2* for more information.

## Adding a replication system domain

A replication system domain includes all replication system components that use the same ID Server. Most replication systems should be set up as a single domain with a single ID Server. However, you may require replicates of two separate data environments in the following situations if:

- Your enterprise requires data management by separate groups, sites, or independent organizations.
- You need to eliminate an ID Server as a single point of failure, thereby creating a fault-tolerant system.

An ID Server failure in a domain results in system degradation. New Replication Servers and databases cannot be added to a domain as long as the ID Server is shut down.

If you do use multiple replication system domains, be sure to have completely independent data environments. For example, assume you have one data environment tracking personnel, and another tracking inventory. As long as there is no data sharing or relationship between these two groups, you can create two separate domains, one for each data environment.

## Guidelines for adding replication system domains

When creating multiple ID Servers for multiple replication system domains, observe these guidelines:

- Make sure all Replication Server and data server names are globally unique across domains.  
  
By using unique names, you simplify your administration and prevent confusion, especially in the interfaces files, which contain network access information for servers.
- Maintain unique names and distinct ID numbers to accommodate the future possibility of data transfer between domains (that is, merging of domains).
  - Provide a different range of database and Replication Server ID numbers for each domain.
  - Make sure the ID numbers of any additional domains are large enough so that they do not overlap with the ranges of the first domain. See “Example of assigning ID numbers” on page 90.

### Example of assigning ID numbers

- Make sure that replication definition names are globally unique within and between ID Server domains.

The ID number is increased each time a Replication Server or database is added to the replication system. By default, your first ID number for a data server is 101. For a Replication Server it is 16,777,317. The last possible ID number for a data server is 16,777,316. For a Replication Server it is 33,554,431.

If you are creating two domains, you could assign ID numbers according to Table 4-1.

**Table 4-1: Suggested ID numbers for multiple ID Servers**

Component	First ID number	Last ID number
1st domain data server	101	99,999
2nd domain data server	100,000	16,777,316
1st domain Replication Server	16,777,317	17,777,316
2nd domain Replication Server	17,777,317	33,554,431

When you are installing an ID Server using the `rs_init` program, you can specify the Starting Replication Server ID and the Starting Database ID.

---

**Note** Make sure your ranges do not overlap from one domain to another. Make your ranges large enough so that ID numbers can never increase to the next range. For example, a range of 99,999 accommodates nearly all possible cases.

---

## Setting Replication Server configuration parameters

You can configure Replication Server or specific objects within the replication system by using one of several methods that update configuration parameters in the `rs_config` system table in the RSSD or the ERSSD. You can also check configuration status information in this table. This section includes:

- An overview of configuration parameters and the objects that these parameters affect



- Information about displaying parameters related to the current Replication Server

---

**Note** Replication Server startup information is stored in a configuration file created by `rs_init`. The default name for the Replication Server configuration file is the Replication Server name with “.cfg” appended. Refer to Chapter 7, “Executable Programs,” in the *Replication Server Reference Manual* for more information about the configuration parameters stored in this file.

---

## About configuration parameters

Replication Server reads configuration parameters from the `rs_config` system table in the RSSD or ERSSD.

All configuration parameters have default values, which are inserted in the table when you use the `rs_init` utility to create the RSSD or ERSSD. The default values are sufficient for most replication systems. Normally, you change default values only for unusual environments or special situations. For example, you may need to adjust parameters for performance tuning if your system has a large number of replication definitions or subscriptions. You can change default values using the `configure replication server` command. See “Configuration parameters that affect performance” on page 131 in the *Replication Server Administration Guide Volume 2* for information on these parameters.

Parameters governing the names and version numbers of the Replication Server and its components can also be set with `rs_init`. Table 4-2 describes these basic parameters.

---

**Warning!** Do not change the values for the parameters listed in Table 4-2. The values are set when you run `rs_init` and should only be modified by the `rs_init` program when you upgrade or downgrade Replication Server.

---

**Table 4-2: Basic configuration parameters**

Configuration parameter	Description
<code>current_rssd_version</code>	The Replication Server version supported by this RSSD. The Replication Server checks this value at startup.
<code>id_server</code>	The name of the ID Server for this Replication Server.

Configuration parameter	Description
minimum_rssd_version	The minimum version of the Replication Server that can use this RSSD. When the current_rssd_version is greater than the version of the Replication Server, this value is checked when the Replication Server is started.
oserver	The name of the current Replication Server.
prev_min_rssd_version	Following an rs_init installation upgrade, this value contains the previous value of minimum_rssd_version.
prev_rssd_version	Following an rs_init installation upgrade, this value contains the previous value of current_rssd_version.
rssd_error_class	Error class for the RSSD. Default: “rs_sqlserver_error_class”
send_enc_pw	Ensures that Replication Server makes client connections to the RSSD with an encrypted password. Values are “on” and “off” (the default).  See “Sending encrypted passwords for Replication Server client connections” on page 196.

rs\_init also sets the password encryption configuration parameter. For information about it, refer to “Enabling and disabling password encryption in sysattributes” on page 201.

Many configuration parameters also have values for specific objects. You set these values after installation when your system requires the fine-tuning that these parameters allow. For example, default route parameters affect all routes that originate at the current Replication Server. If necessary, you change the default settings for these parameters with `configure replication server`. You also can set parameter values for individual routes by using the `alter route` command.

Setting some configuration parameters requires a technical understanding of the replication system. See Chapter 2, “Replication Server Technical Overview” in this book and Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2* for information on how the replication system works.

It is important to back up the RSSD periodically, and whenever you do anything to change its state. ERSSD is already configured for daily automated backup. See “Managing the RSSD” on page 95 for more information, or “Managing Embedded Replication Server System Database” on page 97.

## Different types of configuration parameters

Configuration parameters in the `rs_config` system table affect the Replication Server and different database objects. The method for changing a parameter also varies according to the object that the parameter affects. The different types of configuration parameters are as follows:

- Local Replication Server – parameters whose effects are restricted to the current Replication Server. These parameters are listed in Table 4-2 on page 91 and Table 4-2 on page 132 in the *Replication Server Administration Guide Volume 2*. See “Changing Replication Server parameters” on page 94 for instructions about setting Replication Server parameter values with `configure replication server`.
- Route – parameters that affect routes from the current Replication Server to other Replication Servers. See “Changing routes” on page 151 for information about setting default and per-target values for route parameters.
- Database connection – parameters that affect database connections originating with the Replication Server. See “Setting and changing parameters affecting physical connections” on page 172 for information about setting default and per-target values for database connection parameters. See also “Parallel DSI parameters” on page 152 in the *Replication Server Administration Guide Volume 2* for information about parameters for database connection parameters for parallel DSI.
- Logical database connection – Replication Server parameters that apply to logical database connections for warm standby applications. See “Changing parameters affecting logical connections” on page 99 in the *Replication Server Administration Guide Volume 2* for information about setting default and per-target values for logical connection parameters.
- Network-based security services – parameters that affect network security. See “Managing network-based security” on page 210 for information about setting security parameters.
- Performance – parameters that affect the performance of a Replication Server. See “Configuration parameters that affect performance” on page 131 and “Parallel DSI parameters” on page 152 in the *Replication Server Administration Guide Volume 2*.

## Changing Replication Server parameters

You can modify configuration parameters that affect the current Replication Server by using the `configure replication server` command at the Replication Server.

To change default configuration parameters using `configure replication server`, log in to Replication Server and execute `configure replication server` at the `isql` prompt.

Use the following syntax where *config\_param* is a character string that corresponds to the configuration parameter name and *value* is a character string representing the setting you want for the parameter:

```
configure replication server
    set config_param to 'value'
```

The *config\_param* string must match an entire parameter name. You must restart Replication Server for the new parameters to take effect.

### Example 1

For example, to change the maximum number of messages allowed in the Open Server message queue to 5, log in to the source Replication Server and:

- 1 Execute the `configure replication server` command:

```
configure replication server set num_msgs to '5'
```

- 2 Restart the Replication Server.

Refer to “Starting Replication Server” on page 86 for information about starting Replication Server.

### Example 2

This example uses `configure Replication Server` to change the `ha_failover` parameter to enable **Failover** support for all non-RSSD connections from a Replication Server to Adaptive Servers.

- 1 Execute `configure replication server`. Log in to the Replication Server for which you want to enable Failover support and enter:

```
configure replication server
    set ha_failover to 'on'
```

See “Configuring the replication system to support Sybase Failover” in Chapter 7, “Replication System Recovery,” in the *Replication Server Administration Guide Volume 2*.

- 2 Restart the Replication Server.

Refer to “Starting Replication Server” on page 86 for information about starting Replication Server.

Configuration changes take effect after you restart Replication Server.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about using configure replication server.

Parameters affecting security are covered in Chapter 8, “Managing Replication Server Security” Parameters affecting performance are discussed in Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2*.

### Configuring dynamic parameters

Several Replication Server configuration parameters are changed to dynamic, allowing you to change their values using the configure replication server command. You no longer need to restart the Replication Server for the new values to take effect. Table 4-3 lists the dynamic configuration parameters.

**Table 4-3: Dynamic configuration parameters**

init_sqm_write_delay	init_sqm_write_max_delay
memory_limit	num_concurrent_subs
queue_dump_buffer_size	sqm_recover_segs
sqm_warning_thr_ind	sqm_warning_thr1
sqm_warning_thr2	sqt_max_cache_size
sqt_init_read_delay	sqt_max_read_delay
sts_cachesize	sts_full_cache_system_table_name

Use the new admin config command to retrieve the values of these parameters.

The admin config syntax is:

```
admin config [,"connection" |,"logical_connection" |,"route" ] [,server
[,database]] [,configuration_name]
```

See the *Replication Server Reference Manual*, for a detailed information in using this new command.

## Managing the RSSD

The data in each Replication Server RSSD is essential in keeping the replication system running.

The replication system administrator or Adaptive Server system administrator manages the RSSD by monitoring the condition of the database and performing regular dumps. In the event of disaster recovery, you need to rely on up-to-date backups of the RSSD for full system recovery. Therefore, it is critical that you perform periodic backups of the replication system.

It is also important to back up the RSSD after performing tasks that change its state, such as adding routes, replication definitions, and subscriptions, or altering function strings for databases to which you are connected.

The system tables are loaded into the RSSD during Replication Server installation. You can query the system tables to find the status of the system, but in general, you should not make changes to the tables directly. Refer to the *Replication Server Reference Manual* for detailed descriptions of the system tables.

## Enabling Failover support for an RSSD connection

Sybase **Failover** allows you to configure two version 12.0 or 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.

---

**Note** 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 7, “Replication System Recovery” in the *Replication Server Administration Guide Volume 2*.

---

To enable Failover support for an RSSD connection, use either of the following methods:

- Use `rs_init` when you install a new Replication Server.  
For instructions, refer to Chapter 2, “Configuring Replication Server and Adding New Databases,” in the *Replication Server Configuration Guide* for your platform.
- Edit the Replication Server configuration file after you have installed the Replication Server.

- a Use a text editor to open the Replication Server configuration file. The default file name is the Replication Server name with a “.cfg” extension.

The configuration file contains one line per entry.

- b Find the line “RSSD\_ha\_failover=no” and change it to:

```
RSSD_ha_failover=yes
```

- c To disable Failover support for an RSSD connection, change the “RSSD\_ha\_failover=yes” to:

```
RSSD_ha_failover=no
```

These changes take affect immediately; that is, you do not have to restart Replication Server to enable Failover support.

## Managing Embedded Replication Server System Database

This section talks about Embedded Replication Server System Database (ERSSD) enhancements and its new features.

### Overview

Replication Server can run on either an Adaptive Server Enterprise Replication Server System Database (RSSD) or on an Embedded RSSD (ERSSD). ERSSDs are designed for users who do not want to use Adaptive Server Enterprise to manage the Replication Server RSSD. Replication Server is easy to install and manage with ERSSD. ERSSD is automatically installed, configured, and started in the background if you specify that you want to use it. Backup procedures are automatic and preconfigured.

#### Limitations

You cannot migrate from ERSSD to RSSD.

To use ERSSD, you must select it when you install Replication Server. For more details, see the *Replication Server Installation Guide* for your platform.

---

**Note** Sybase provides ERSSD as an option in Replication Server, implemented in Adapted Server Anywhere. Sybase continues to support the traditional RSSD, implemented in Adaptive Server Enterprise. All the RSSD features discussed in this section pertain to ERSSD only; they do not affect the behavior of the traditional RSSD in Adaptive Server Enterprise.

---

## Before you start

ERSSD runs on three files:

- A database root file
- A transaction log file
- A transaction log mirror file

These are operating system files. When you start `rs_init`, provide the directories for these files, and make sure that the name of your ERSSD is in the `interfaces` file.

---

**Note** For better performance and better protection against disk failure, put each one of these files on a different physical device.

---

## Configuring ERSSD

ERSSD has a preconfigured backup time, backup interval, and backup directory. Unless you want to change these defaults, you need not configure ERSSD.

To check the current default values, enter:

```
sysadmin erssid
```

You can find the following information in the Replication Server configuration file:

- ERSSD database file path
- ERSSD transaction log file path



- ERSSD transaction log mirror file path
- Backup directory path

## Configuration parameters and command

You can configure the ERSSD backup time and directory by using the following Replication Server configuration parameters in Table 4-4, with this command:

```
configure replication server
set
    {erssd_backup_start_time |
    erssid_backup_start_date |
    erssid_backup_dir |
    erssid_backup_interval | erssid_ra}
to 'value'
```

Do not update these values directly in the `rs_config` table.

**Table 4-4: ERSSD configuration parameters**

Repserver configuration parameter	Value	Default
<code>erssd_backup_start_time</code>	Time the backup starts. Specified as “hh:mm AM” or “hh:mm PM”, using a 12-hour clock, or “hh:mm” using a 24-hour clock.	01:00 AM
<code>erssd_backup_start_date</code>	Date the backup begins. Specified as “MM/DD/YYYY”	Current date
<code>erssd_backup_interval</code>	Interval between backups of database and log. Specified as “nn hours” or “nn minutes” or “nn seconds”.	24 hours
<code>erssd_backup_dir</code>	Location of stored backup files. Should be a full directory path. Configuring this path causes an immediate, unscheduled backup.	Same directory as the transaction log mirror; initial value specified in <code>rs_init</code> .
<code>erssd_ra</code>	Its value should be a server name. It is only used when when a user creates a route from the current site	<code>erssd_name_ra</code> ; where <code>erssd_name</code> is the ERSSD name in the user’s replication system

## ERSSD routing

You can create a route from a Replication Server with ERSSD, as long as both the source and the destination servers are version 15.0 or later.

To create a route from Replication Server with ERSSD, use the `create route` command. Verify that the Replication Agent name is in the Replication Server interfaces file; an ERSSD Replication Agent is started as an open server during `create route`, and if its name does not appear in the interfaces file the command fails.

The default ERSSD Replication Agent name is `erssd_name_ra`. To replace the default name with that of your Replication Agent server, enter:

```
configure replication server
set erssd_ra to 'value'
```

---

**Note** Sybase provides ERSSD in Adaptive Server Anywhere (ASA) as an option, and continues to support the traditional RSSD in the Adaptive Server Enterprise.

---

## Moving ERSSD files

Use the command `sysadmin erssd` to move the ERSSD database file, transaction log, or transaction log mirror. Do not edit the configuration file itself. Moving the database file, transaction log, and transaction log mirror is an expensive operation. Only use it when you are sure it is necessary. For more information on `sysadmin erssd`, see the *Replication Server Reference Manual*.

## ERSSD users

There are only two users in the ERSSD, the primary user, who also acts as System Administrator, and the maintenance user. You can find their names and passwords in the configuration file. You can do the following to change the user password:

- Use the Replication Server `alter user` command to alter the primary user password.
- Use the Replication Server `alter connection` command to alter the maintenance user password.

Both these commands alter the password at Replication Server as well as at ERSSD, and update the Replication Server configuration file.

For more information on these commands see the *Replication Server Reference Manual*.

To add a user at the ERSSD, use `isql` to access the ERSSD as the primary user and execute the command `grant connect to username identified by password`.

To give a user permission to read the Replication Server system tables, execute the command `grant membership in group rs_systabgroup to username`.

To grant `sa` privileges to a user, execute the command `grant dba to username`.

## Backup

There are four files in the backup directory. This directory is specified when you install Replication Server with ERSSD.

**Table 4-5: Backup directory files**

File name	File definition
<code>erssd_name.db</code>	Backup database file
<code>erssd_name.log</code>	Backup transaction log
<code>erssd_name.db.pre</code>	Previous backup database file
<code>erssd_name.log.pre</code>	Previous backup transaction log

An automatic full backup, including both the database file and the transactional log file, is performed at the default or the configured time.

The transaction log is mirrored, providing extra protection for critical data, and enables complete recovery of the transaction log file.

To perform an unscheduled backup, use this Replication Server command:

```
sysadmin erssd, backup
```

## Recovery instructions

ERSSD automatically manages recovery from operating system crashes, database server crashes, and crashes caused by shutting down improperly. The instructions in these procedures are designed for recovering a database damaged by media failure.

## Before you start recovery

Before using the recovery commands, set the following environment variables.

On UNIX platforms:

- Set your environment variable `PATH` to include `$$SYBASE/$SYBASE_REP/ASA9/bin`:

```
setenv PATH $$SYBASE/$SYBASE_REP/ASA9/bin:$PATH
```

- Set your environment variable `LD_LIBRARY_PATH` (`SHLIB_PATH` on HP, `LIB_PATH` on AIX) to include `$$SYBASE/$SYBASE_REP/ASA9/lib`:

```
setenv LD_LIBRARY_PATH  
$$SYBASE/$SYBASE_REP/ASA9/lib:$LD_LIBRARY_PATH
```

On Windows:

- Set your environment variable `PATH` to include `%SYBASE%\%SYBASE_REP%\ASA9\win32`:

```
set PATH=%SYBASE%\%SYBASE_REP%\ASA9\win32;%PATH%
```

## Recovery procedures

Use these procedures to ensure clean recovery after media failure.

### ❖ Recovering after media failure of the database file

- 1 Make an extra backup copy of the current transaction log. If the database file is gone, the only record of changes since the last backup is in the transaction log.
- 2 Create a recovery directory to hold the files you use during the recovery process.
- 3 Copy the database file from the last full backup to the recovery directory. You can find the database file in the backup directory. It is named *erssd\_name.db*.
- 4 Copy the backup transaction log into the recovery directory. The backup transaction log, named *erssd\_name.log*, is in the backup directory.
- 5 Apply the transactions from the backup transaction log to the recovery database:

```
dbsrv9 erssd_name.db -a erssd_name.log
```

- 6 Copy the online transaction log into the recovery directory. The online transaction log, named *erssd\_name.log*, is in the log directory.

- 7 Apply the transactions from the online transaction log to the recovery database:

```
dbsrv9 erssid_name.db -a erssid_name.log
```

- 8 Make a post-recovery backup by making an extra copy of the database file.
- 9 Move the database file to the production directory and restart the database. Use the command `dbspawn` from the Replication Server error log.
- 10 Perform validity checks on the recovery database:

```
dbvalid -c
"uid=primary_user_name;
pwd=primary_user_password;eng=erssd_name
LINKS=tcPIP
(DOBROAD=NONE;HOST=localhost;PORT=port) "
```

- 11 Restart Replication Server.

#### ❖ Recovering from media failure on the database transaction log

- 1 Identify the corrupted file. You can do this by running the Log Translation utility on both the transaction log and its mirror to see which one generates an error message. In this example, the Log Translation utility, `dbtran`, translates a transaction log named `erssd_name.log`, saving the translated output in `db_name.sql`.

```
dbtran erssid_name.log
```

The Log Translation utility translates the intact file with no errors, but reports an error when translating the corrupt file.

- 2 Copy the correct file over the corrupted file, so that the two files are identical.
- 3 Restart the database, using command from the Replication Server error log.
- 4 Restart Replication Server.

## ERSSD command and options

For detailed information on the `sysadmin erssid` command, see the *Replication Server Reference Manual*.

## Quiescing Replication Server

To **quiesce** a replication system means to put the system in a state in which no Replication Servers have messages to send or receive. You may need to quiesce all Replication Servers in the system to recover databases, alter routes, and troubleshoot the system.

A Replication Server is quiescent when the following conditions are true:

- Subscription materialization queues do not exist.
- Replication Server has read all messages in all queues.
- Transaction caches for inbound queues contain no complete transactions.
- Messages in RSI queues have been sent and acknowledged.
- Messages in DSI queues have been applied and acknowledged.

### Quiescing a replication system

You can use Sybase Central or the procedure discussed in this section to quiesce a system consisting of several Replication Servers.

To quiesce a replication system:

- 1 Execute the suspend log transfer from all command at each Replication Server. This prevents RepAgent from connecting to the Replication Servers.
- 2 Execute `admin quiesce_force_rsi` at each Replication Server.

This command forces Replication Server to deliver all queued messages to other Replication Servers, then reports whether the system is successfully quiesced.

Quiescing occurs most efficiently if you follow the flow of the data. For example, if data flows from TOKYO\_RS to MANILA\_RS to SYDNEY\_RS, quiesce the Replication Servers in that order.

- 3 Check that the Replication Server is quiescent using `admin quiesce_check`. If necessary, repeat steps 2 and 3 until all Replication Servers are quiescent.
- 4 After all Replication Servers are quiescent, execute `admin quiesce_force_rsi` once more at each Replication Server. Check that each Replication Server is quiescent using `admin quiesce_check`. If necessary, repeat this step until all Replication Servers are quiescent.

This step is necessary because, although a Replication Server may be quiescent, it may have recently sent messages to another Replication Server. These messages may initiate more communication between these two Replication Servers or between several Replication Servers in the replication system. Repeating steps 2 and 3 ensures that you have quiesced the entire replication system.

## Removing a Replication Server

How you remove a Replication Server from a replication system depends on whether or not the Replication Server is active (running). Although this section contains procedures for both situations, it is easiest to remove a Replication Server that is active.

The procedures in this section also require that you drop routes and subscriptions. See Chapter 11, “Managing Subscriptions” and Chapter 6, “Managing Routes” for details.

## Removing an active Replication Server

This section tells you how to remove a running Replication Server from service:

- 1 Query the RSSD to determine what replication definitions are defined at the primary Replication Server (the server you are removing from service). You can use the `rs_helprep` stored procedure to do this. Refer to Chapter 8, “Replication Server System Tables,” in the *Replication Server Reference Manual* for information on the RSSD system tables.
- 2 Drop subscriptions and replication definitions.

This can be done using the following command:

- a For each replication definition defined at the primary Replication Server, execute the drop subscription command for each subscription on all Replication Servers that manage subscribing data.

To retain data at the replicate, execute the drop subscription command without purge.

To delete data at the replicate, execute the drop subscription command with purge.

See “Using the drop subscription command” on page 380 for more information about dropping subscriptions.

- b Drop all replication definitions for primary data managed by the Replication Server (determined in step 1).

Wait for the replication definitions to disappear from the RSSDs of Replication Servers that the Replication Server has a route to.

- c At the Replication Server you are removing, drop all subscriptions to replication definitions on other Replication Servers.

To retain data at the replicate, execute the drop subscription command without purge.

To purge data at the replicate, execute the drop subscription command with purge.

- 3 If the Replication Server is the primary Replication Server for a function-string class or error class, execute the move primary command at another Replication Server to change the primary Replication Server for each class.

During a move primary operation, routes must exist from the old primary site to the new primary site, and from the new primary site to the old primary site. The Replication Server assuming the role of the primary site also must have routes to all of the same Replication Servers as the old primary site.

- 4 Drop database connections.
  - a Stop all RepAgent connected to the Replication Server, using the `sp_stop_rep_agent` system procedure at Adaptive Server.
  - b Remove connections to all databases managed by this Replication Server, using the drop connection command.

---

**Note** If you want to continue to maintain the replicate data in databases previously managed by a Replication Server that has been removed from service, you must create connections to those databases from some other Replication Server and create new subscriptions.

---

- 5 Perform the following routing tasks:
  - a If the Replication Server is an intermediate site in a route, use the alter route command so it is no longer an intermediate site.
  - b Drop all routes *from* the Replication Server.



To do this, execute the drop route command for each route from the Replication Server to another Replication Server.

- c Drop all routes *to* the Replication Server.

To do this, execute the drop route command on each Replication Server that has a route to the Replication Server you are removing.

See Chapter 6, “Managing Routes” for more information about altering and dropping routes.

- 6 After all subscriptions and routes to and from the Replication Server are dropped, remove the Replication Server from the list maintained by the ID Server. To do this, execute the sysadmin droprs command on the ID Server:

```
sysadmin droprs, replication_server
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information on the sysadmin droprs command.

- 7 Remove all databases managed by the Replication Server from the database list maintained by the ID Server. Include the RSSD. To remove databases, run the sysadmin dropdb command on the ID Server, for each database:

```
sysadmin dropdb, data_server, database
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information on the sysadmin dropdb command.

## Removing an inactive Replication Server

An inactive Replication Server is one that is not running. To take an inactive Replication Server out of service, follow these steps:

- 1 Drop all routes to the Replication Server.

To do this, execute the drop route command with the with nowait option on each Replication Server that has a route to the Replication Server. For example:

```
drop route to OLD_RS with nowait
```

This command also deletes information about subscriptions created at OLD\_RS for data managed by this Replication Server.

- 2 If the Replication Server you are removing is primary for any function-string classes or error classes other than the system defaults, `rs_default_function_class` and `rs_sqlserver_error_class`, create a replacement for each class at a new primary. To do this:
  - Choose a Replication Server that has routes to all other Replication Servers that use the class.
  - Create a new class at that Replication Server containing the same function strings or error actions as the original class. See Chapter 2, “Customizing Database Operations” and Chapter 6, “Handling Errors and Exceptions” in the *Replication Server Administration Guide Volume 2* for details.
  - Alter each database connection that is using the original class to use the new class instead. See Chapter 7, “Managing Database Connections” for details.
- 3 On each Replication Server that has a route from the Replication Server, purge the Replication Server route.

To purge a route, execute the `sysadmin purge_route_at_replicate` command on each Replication Server to which the Replication Server had a route. For example:

```
sysadmin purge_route_at_replicate, OLD_RS
```

This command also removes:

- Subscription information for data originating at the Replication Server you are removing from service.
  - Function-string and error classes defined at the Replication Server you are removing from service. If the Replication Server is the primary site for `rs_default_function_class`, `rs_sqlserver_function_class`, or `rs_sqlserver_error_class`, these classes are not removed but are reset to have no primary Replication Server.
- 4 Remove the Replication Server from the list maintained by the ID Server. To do this, execute the `sysadmin droprs` command on the ID Server:

```
sysadmin droprs, replication_server
```

See the *Replication Server Reference Manual* for more information on the `sysadmin droprs` command.

- 5 Remove all databases managed by the Replication Server from the database list maintained by the ID Server. Include the RSSD. To remove databases, run the `sysadmin dropdb` command on the ID Server, for each database:

```
sysadmin dropdb, data_server, database
```

See the *Replication Server Reference Manual* for more information on the `sysadmin dropdb` command.

This completes the removal of an inactive Replication Server from a replication system.

Keep in mind these three additional points:

- If you want to continue to replicate any data in the databases previously managed by the Replication Server, you must reassign those databases to some other Replication Server.
- Since the subscriptions to the Replication Server data did not go through normal subscription dematerialization, replicate data has not been deleted from replicate Replication Servers.
- You may need to create additional routes to maintain the replication system—for example, if the Replication Server is an intermediate on an indirect route.



# Setting Up and Managing RepAgent

This chapter describes how to set up, configure, and manage RepAgent, the Replication Agent for Adaptive Server.

Topic	Page
Setting up RepAgent	112
Configuring RepAgent	114
Starting RepAgent	118
Stopping RepAgent	119
Disabling RepAgent	119
Checking log files for information and error messages	120
Configuring RepAgent for network security	120
Handling extended limits	121
Support for longer identifiers	122
Adaptive Server shared-disk cluster support	123
Reviewing status and configuration information	124
Managing log transfer activity	126
Using counters to monitor RepAgent performance	128

RepAgent is an Adaptive Server thread; it scans the database transaction log and sends transaction information to the Replication Server for distribution to subscribing databases.

See Chapter 2, “Replication Server Technical Overview” and Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2* for detailed information about how RepAgent processes transaction data.

## Setting up RepAgent

After Replication Server and Adaptive Server are installed on your system, you must enable a RepAgent for each database the Replication Server manages—if the database:

- Contains primary data, or
- Contains stored procedures marked for replication

In addition, if Replication Server is the source site for any route, you must enable RepAgent for the Replication Server RSSD.

There are three possible scenarios for setting up RepAgent. In some scenarios you use `rs_init`, in other scenarios you must use command line options.

- If you install a new Replication Server or add a new database, use `rs_init` to set up RepAgent. This process enables RepAgent, set default parameters, and start RepAgent. See the *Replication Server Configuration Guide* for your platform for information about `rs_init`.
- To change an existing replicate database to a primary database, you must use command line options.

### ❖ **Configuring RepAgent using command line options:**

These are the basic steps for configuring RepAgent from the command line.

- 1 Define the local Adaptive Server using `sp_addserver`.
- 2 Enable the RepAgent feature on Adaptive Server using `sp_configure`.
- 3 Enable the RepAgent feature for each database using `sp_config_rep_agent`.
- 4 Enable log transfer on Replication Server using `alter connection`.
- 5 Start the RepAgent on Adaptive Server using `sp_start_rep_agent`.

## Defining the local Adaptive Server

If you are starting Adaptive Server for the first time, you must execute the Adaptive Server system procedure `sp_addserver` to add an entry for the local server to Adaptive Server `sys.servers` table. Refer to the *Adaptive Server Enterprise Reference Manual* for information about using `sp_addserver`.

## Enabling RepAgent on Adaptive Server

Enable the RepAgent feature on the Adaptive Server using `sp_configure`. You need to perform this task only once at each Adaptive Server.

Log in to Adaptive Server and enter this command at the `isql` prompt:

```
sp_configure 'enable rep agent threads', 1
```

`sp_configure 'enable rep agent threads'` is a dynamic option. It takes effect immediately. However, you may want to restart Adaptive Server after enabling RepAgent so that Adaptive Server allocates a fixed number of dedicated static process structures for the thread. Otherwise, RepAgent borrows process structures from the pool dedicated to user connections.

## Enabling RepAgent for the database

For each primary database, you must:

- Enable the RepAgent for the database using `sp_config_rep_agent`
- Turn on log transfer for the connection using `alter connection`

## Enabling RepAgent

Execute `sp_config_rep_agent` to enable the RepAgent for the database and set default values for RepAgent configuration parameters. You can reset the default values at a later time.

Log in to Adaptive Server. At the `isql` prompt, enter:

```
use dbname  
go  
sp_config_rep_agent dbname, enable, 'repserver_name',  
    'repserver_username', 'repserver_password'
```

*dbname* is the name of the database for which you are enabling RepAgent, *repserver\_name* is the Replication Server to which RepAgent connects, and *repserver\_username* and *repserver\_password* are the name and password RepAgent uses to log in to Replication Server.

---

**Note** Make sure that *repserver\_username* is a valid Replication Server user and that it has Replication Server connect source permission. Try out the user name and password at the Replication Server before you use `sp_config_rep_agent`.

---

Refer to “Configuring RepAgent” on page 114 for information about setting RepAgent parameters with `sp_config_rep_agent`.

### Turning on log transfer

---

**Note** You must create a database connection between Replication Server and the data server using `rs_init` or `create connection` before you can turn on log transfer. See the *Replication Server Configuration Guide* for your platform for information about creating connections using `rs_init`; refer to Chapter 7, “Managing Database Connections” for information about using `create connection`.

---

Turn on log transfer for the database connection to the primary database using `alter connection`. For example, at the Replication Server, enter:

```
alter connection to TOKYO_DS.pubs2
    set log transfer on
```

## Configuring RepAgent

Enabling RepAgent (using `rs_init` or `sp_config_rep_agent`) sets default configuration parameters. You can change the default parameters using `sp_config_rep_agent`. You must restart RepAgent for the new parameters to take effect.

Table 5-1 describes the configuration parameters that affect RepAgent. These parameters are stored in the `sysattributes` table of the database for which RepAgent is enabled.



If your system supports network-based security, refer to “Managing network-based security” on page 210 for a list and description of network security configuration parameters for RepAgent.

**Table 5-1: Configuration parameters affecting RepAgent**

Configuration parameter	Description
batch ltl	When set to true, sends LTL commands to Replication Server in batches. Otherwise, sends LTL commands to Replication Server one at a time. Default: true
connect database	The name of the database for which RepAgent is configured, or the name of the temporary database RepAgent uses when connecting to Replication Server in recovery mode.
connect dataserver	The name of the RepAgent data server or the name of the temporary data server RepAgent uses when connecting to Replication Server in recovery mode.
data limits filter mode	Specifies how RepAgent handles log records containing column counts greater than 250, column lengths greater than 255 bytes, and parameter lengths greater than 255 bytes before attempting to send them to Replication Server. Values are: <ul style="list-style-type: none"> <li>• off – RepAgent allows all records to pass through. In Replication 12.1 and earlier, this setting can cause undesirable effects.</li> <li>• stop – RepAgent shuts down if it encounters log records containing data that exceeds limits of Replication Server 12.1 and earlier.</li> <li>• skip – RepAgent skips log records containing data that exceeds limits of Replication Server 12.1 and earlier and posts message to error log.</li> <li>• truncate – RepAgent truncates data exceeding 255 bytes per column and 250 columns per table.</li> </ul> Default: off (Replication Server 12.5 and later); stop (Replication Server 12.1 and earlier)
ha failover	Specifies whether, when Sybase Failover has been installed, RepAgent automatically starts after server failover. Default: true
net password encryption	In Adaptive Server 15.0.2, when this parameter is set to true, RepAgent sets both the CS_SEC_ENCRYPTION and the CS_SEC_EXTENDED_ENCRYPTION connection properties. Otherwise, none of these properties are set. Default: true

**Note** If unified login or mutual authentication security properties are set, net password encryption parameter will be ignored, since these security properties are using credentials for authentication.

Configuration parameter	Description
priority	Sets relative priority values for individual RepAgents. Recommended values are 4, 5, and 6, where 6 indicates low priority, 5 indicates medium priority, and 4 indicates high priority. Default: 5
rs name	The name of the Replication Server to which RepAgent connects and transfers transactions from the transaction log. Use rs name when you have changed the name of the Replication Server.
rs password	The password RepAgent uses to log in to Replication Server. Use rs password when you want to change the RepAgent password.
rs username	The user name RepAgent uses to log in to Replication Server. Use rs_username when you want to change the RepAgent username.
retry timeout	The number of seconds RepAgent remains inactive before attempting to reconnect to Replication Server after a recoverable error or when Replication Server is down. Default: 60 seconds
scan batch size	The maximum number of log records to send to Replication Server in each batch. When this number of records have been sent, RepAgent asks Replication Server for a new secondary truncation point. Default: 1000 records
scan timeout	The amount of time RepAgent remains inactive after sending a batch to the Replication Server and before querying the Replication Server for the new secondary truncation point. If there are more records in the log, RepAgent resumes scanning. If there are no more records and the Replication Server has still not acknowledged receipt by sending a secondary truncation point, RepAgent again timeouts for the length of time this parameter is set to. Default: 15 seconds
schema cache growth factor	Controls the duration of time table or stored procedure schema can reside in the RepAgent schema cache before they expire. Larger values require more memory. Range is 1 to 10. Default: 1
send buffer_size	Controls the size, in kilobytes, of the send buffer RepAgent uses to communicate with Replication Server. Values are 2K, 4K, 8K, and 16K. Default: 2K  <b>Note</b> Send-buffer size is not related to database page size.
send maint xacts to replicate	When set to true, RepAgent sends records generated by the maintenance user to the Replication Server for distribution to subscribing sites. Otherwise, RepAgent does not send records from the maintenance user to the Replication Server. Default: false

Configuration parameter	Description
send structured oqids	Specifies whether RepAgent sends origin queue IDs (OQIDs) as structured tokens or as binary strings. When set to true, RepAgent sends OQIDS as structured tokens, which saves space in the LTL and improves throughput. Default: false
send warm standby xacts	Normally schema and system transactions are not sent to a warm standby database. When set to true, RepAgent sends schema, system, and maintenance-user transactions. Otherwise, RepAgent does not send transactions to the standby database. Default: false
short ltl keywords	Specifies whether RepAgent sends an abbreviated form of LTL to Replication Server. When set to true, RepAgent uses the shortened LTL form that requires less space and reduces the amount of data sent to Replication Server. Default: true
skip ltl errors	When set to true, RepAgent ignores LTL errors returned by the Replication Server. This option is normally turned on during recovery. Default: false
skip unsupported features	Instructs RepAgent to skip log records for features unsupported by the Replication Server. This option is normally used if Replication Server is a earlier version than Adaptive Server. Default: false
startup delay	Controls when a specific RepAgent is started during Adaptive Server start-up. This delays the RepAgent startup by a specified duration to allow Replication Server to run before RepAgent attempts to connect to Replication Server. By default, the RepAgent starts without any delay during automatic start-up. Setting a value in seconds results in a delay in RepAgent start-up by the specified number of seconds. Default: 0 (zero) seconds.

To configure RepAgent, log in to Adaptive Server and execute `sp_config_rep_agent` at the `isql` prompt. For complete syntax and usage information, see the *Replication Server Reference Manual*.

Execute `sp_config_rep_agent` once for each parameter you want to configure. For example, to change the maximum number of log records sent to Replication Server in a batch to 2000, perform these steps:

1 Log in to Adaptive Server and enter:

```
use dbname
go
sp_config_rep_agent dbname, 'scan batch size',
'2000'
```

2 Restart RepAgent:

```
sp_start_rep_agent dbname
```

The new parameter takes effect after you restart RepAgent.

Refer to “Starting RepAgent” on page 118 for more information about `sp_start_rep_agent`.

## Starting RepAgent

Normally, you need to start a RepAgent thread only if:

- You have reconfigured the RepAgent parameters.
- You have explicitly shut down the RepAgent.

RepAgent starts automatically when Adaptive Server restarts if the RepAgent has been started at least once with `sp_start_rep_agent` and not stopped with `sp_stop_rep_agent`.

To start RepAgent, log in to Adaptive Server and enter `sp_start_rep_agent` at the isql prompt. For example:

```
sp_start_rep_agent pubs2
```

In this example, *pubs2* is the name of the database for which the RepAgent has been enabled.

---

**Note** RepAgent can be restarted only if its associated database is fully recovered and online and log transfer is on for the connection to the primary database.

---

Refer to Chapter 5, “Adaptive Server Commands and System Procedures,” in the *Replication Server Reference Manual* for detailed information about each option of `sp_start_rep_agent`.

## Stopping RepAgent

To shut down RepAgent, log in to Adaptive Server and execute `sp_stop_rep_agent`. When RepAgent restarts, it scans records starting with the oldest transaction, but it only sends records following the last one processed. As a result, Replication Server does not receive duplicate records.

For example, to stop RepAgent, enter:

```
sp_stop_rep_agent pubs2
```

If you shut down RepAgent in this way, Adaptive Server shuts down RepAgent gracefully at the end of the current batch of transactions.

You can also shut down RepAgent immediately using the `nowait` option. For example:

```
sp_stop_rep_agent pubs2, nowait
```

If you shut down RepAgent with the `nowait` option, Adaptive Server kills the RepAgent without waiting for currently executing operations to finish.

Once RepAgent has been shut down with `sp_stop_rep_agent`, it does not automatically start up when the database comes online during data server startup. You must execute `sp_start_rep_agent`, which starts up RepAgent and resumes automatic start-up.

## Disabling RepAgent

---

**Note** You should disable RepAgent only when you change the replicate database to a primary database, or downgrade Replication Server to an earlier version.

---

Before disabling RepAgent using `sp_config_rep_agent`, you must first shut it down using `sp_stop_rep_agent`.

Normally, when you disable RepAgent, the process also disables the secondary truncation point. For example:

```
sp_config_rep_agent pubs2, 'disable'
```

Once the secondary truncation point is disabled, the log can get truncated past the secondary truncation point.

To disable RepAgent but keep the secondary truncation point, use the `preserve secondary truncpt` option.

```
sp_config_rep_agent pubs2, 'disable', 'preserve
secondary truncpt'
```

Disable RepAgent in this way to disable RepAgent momentarily.

If you are changing the primary to a replicate database, you must also turn log transfer off. After disabling RepAgent, turn log transfer off using `alter connection`.

For example, log in to Replication Server and enter:

```
alter connection to TOKYO_DS.pubs2
set log transfer off
```

## Checking log files for information and error messages

Error and information messages for RepAgent are recorded in the Adaptive Server errorlog file. Refer to the *Adaptive Server Enterprise System Administration Guide* for more information about the Adaptive Server error log.

For example, starting RepAgent generates this message in the Adaptive Server errorlog:

```
00:00000:00022:2003/09/18 12:16:39.15 server Started
RepAgent on database, 'pubs2' (dbid = 4).
```

Stopping RepAgent generates this message:

```
00:00000:00022:2003/09/18 12:17:17.07 server Shutting
down RepAgent for database, 'pubs2' (dbid=4).
```

## Configuring RepAgent for network security

You can secure the pathway between RepAgent and Replication Server using network-based security features. Using `sp_config_rep_agent`, you can change settings for:

- The active security mechanism

- Unified login
- Mutual authentication
- Message confidentiality
- Message integrity
- Message replay detection
- Message origin check
- Message out of sequence check

Refer to “Managing network-based security” on page 210 for a complete description of network security and instructions for setting parameters for RepAgent.

## Handling extended limits

Replication Server version 12.5 and later supports these extended limits for replication definitions:

- More columns, to a maximum of 1024
- Wide columns and parameters, to a maximum of 32768 bytes
- Wide data rows to the width of the data page on the data server
- Wide messages larger than 16K

If the Replication Server site version is 12.5 or later, Replication Server sets the LTL version automatically to 400. If RepAgent is running on Adaptive Server 12.5 or later, RepAgent sends data with extended limits only if Replication Server specifies an LTL version of 400 or higher at connect source time.

If the Replication Server site version is 12.1 or earlier, the LTL version is earlier than 400. If RepAgent is running on Adaptive Server 12.5 or later, Sybase recommends that you do not send extended-limits data to Replication Server 12.1 and earlier. You can specify how RepAgent handles extended-limits data by using the data limits filter mode parameter with `config_rep_agent`. See “Configuring RepAgent” on page 114.

## Support for longer identifiers

Replication Server version 15.0 and later increases the maximum length of these replication object identifiers to 255 bytes:

- Table name and column name
- Stored procedure name and parameter name
- Functions and parameters – for function replication definitions and internal use only
- Function string name
- Replication definitions – including table replication definitions, function replication definitions, and database replication definitions
- Article name
- Publication name

If the Replication Server site version is 15.0, Replication Server sets the LTL version automatically to 700. If RepAgent is running on Adaptive Server 15.0 or later, RepAgent sends data with extended size only if Replication Server specifies an LTL version of 700 or higher at connect source time.

If the Replication Server site version is 12.6 or earlier, the LTL version is earlier than 700. If RepAgent is running on Adaptive Server 15.0 or later, Sybase recommends that you do not send data with longer identifiers to Replication Server 12.6 and earlier.

You can specify how RepAgent handles data with longer identifier by using the data limit filter mode parameter with `config_rep_agent`. See “Configuring RepAgent” on page 114.

---

**Note** The create function, alter function, and drop function commands do not support long identifiers. The name of the function and the parameters of these commands cannot exceed 30 bytes.

---



## Adaptive Server shared-disk cluster support

Replication Server and RepAgent thread both support the Adaptive Server shared-disk cluster environment. In a Sybase shared-disk cluster, a database can be either a replication source or a replication destination. You can perform all of the tasks, such as configuring RepAgent or marking tables for replication, from any instance in the cluster. Replication status is coherent across the entire cluster.

When adding new connections from or to an Adaptive Server cluster environment, the *servername* in the connection syntax must be the *clustername* and not the *instancename*. Use `select @@servername` to retrieve the *clustername*.

By default, the RepAgent starts on the cluster coordinator; However, you can configure it to start on any instance in the cluster. For example, to configure the RepAgent on the primary database *pdb* to always start on the “ase2” instance, enter:

```
sp_config_rep_agent pdb, "cluster instance name" "ase2"
```

For a new configuration to take effect, restart the RepAgent using `sp_start_rep_agent`. To return to the default behavior with the RepAgent starting on the cluster coordinator, enter:

```
sp_config_rep_agent pdb, "cluster instance name",  
"coordinator"
```

When an instance starts, it checks if there are RepAgents configured to start on its node. If there are, and if the database is marked to start automatically, the RepAgent starts.

When the cluster coordinator starts, it also starts all RepAgents that are not configured to start on a specific instance. If the coordinator node fails, or is stopped with a graceful shutdown, a RepAgent starts on the new coordinator node.

If the RepAgent is configured to start on an instance other than the coordinator node, and this instance fails, the RepAgent starts on the coordinator.

---

**Note** The Cluster Edition does not support Adaptive Server Enterprise Replicator, which requires the dbcc log transfer interface.

---

See the *Replication Server Reference Manual* for information about the cluster instance name configuration parameter.

## Reviewing status and configuration information

You can monitor RepAgent in the Adaptive Server plug-in to Sybase Central, or you can use the commands and system procedures described in this section.

### Viewing RepAgent information

You can monitor the RepAgent by using `sp_help_rep_agent` at Adaptive Server. `sp_help_rep_agent` displays information about:

- Recovery – status and other information when you are restoring a database.
- Configuration parameters – the current settings for RepAgent’s configuration parameters.
- Process – information about the RepAgent process, including state, sleep status, number of unsuccessful connection retries (if any), and the number of the last error message.
- Scanned transactions– information about the current batch of log transactions: start, end, and current markers; the number of records in the batch; and the oldest transaction.
- Security – the current settings of the network-based security mechanism.
- All – all of the above information.

Log in to Adaptive Server and execute `sp_help_rep_agent` at the `isql` prompt:

```
sp_help_rep_agent [dbname[, 'recovery' | 'config' |  
'process' | 'scan' | 'security' | 'all']]
```

*dbname* is the name of the database for which the RepAgent is enabled.

Refer to Chapter 5, “Adaptive Server Commands and System Procedures,” in the *Replication Server Reference Manual* for detailed syntax and usage information about `sp_help_rep_agent`.

You can view current status information for one or all options, for example:

- To display information about the RepAgent process, log in to Adaptive Server and enter:

```
sp_help_rep_agent pubs2, 'process'
```

- To display information about the RepAgent log scanning, enter:

```
sp_help_rep_agent pubs2, 'scan'
```

See the *Replication Server Reference Manual* for examples of `sp_help_rep_agent` output.

## Viewing configuration parameter values

To view a list of default, current, and runtime configuration parameter values for a particular RepAgent, log in to Adaptive Server and execute `sp_config_rep_agent` without options. For example:

```
sp_config_rep_agent pubs2
```

If you do not specify a database name, `sp_config_rep_agent` displays configuration values for all RepAgent-enabled databases.

To view values for a specific parameter, include the parameter name. For example:

```
sp_config_rep_agent pubs2, 'scan batch size'
```

See the *Replication Server Reference Manual* for output examples.

Refer to Chapter 5, “Adaptive Server Commands and System Procedures,” in the *Replication Server Reference Manual* for more information about `sp_config_rep_agent`.

## Viewing RepAgent thread information

To view the RepAgent thread status on Adaptive Server, execute `sp_who`. In the display output, Adaptive Server shows the RepAgent information in rows with “REP AGENT” in the “cmd” column.

For example, `sp_who` might display this row for RepAgent:

```
fid spid status loginame origname hostname blk_spid dbname cmd block_xloid
-----
...
0 23 background NULL NULL 0 pubs2 REP AGENT 0
...
```

See the *Adaptive Server Enterprise Reference Manual* for detailed syntax and usage information for `sp_who`.

To view RepAgent thread user status on Replication Server, execute `admin who`. Replication Server displays RepAgent thread user information in rows with “REP AGENT” in the “name” column.

For more information about `admin who` and output examples, refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual*.

## Managing log transfer activity

If you are performing recovery, troubleshooting, or diagnostic tasks, you may need to suspend and resume log transfer. This section describes how to use these log transfer commands:

- `resume log transfer` and `suspend log transfer`
- `alter connection ... set log transfer on/off`

---

**Note** RepAgent cannot connect to Replication Server unless log transfer has first been set on using `alter connection`.

---

See Chapter 7, “Replication System Recovery” of the *Replication Server Administration Guide Volume 2* for information about starting the RepAgent thread in recovery mode so that it can replay database and transaction dumps.

## Using the log transfer commands

This section describes how to suspend and resume log transfer using the `suspend log transfer` and `resume log transfer` commands.

### Suspending log transfer

To disconnect one or all RepAgents and prevent RepAgents from connecting to Replication Server, execute the `suspend log transfer RCL` command. Log transfer to Replication Server remains suspended until you resume it using the `resume log transfer` command.

The suspend log transfer command records information in the RSSD, so if you shut down Replication Server and restart it, log transfer to that Replication Server remains suspended.

---

**Note** Suspending log transfer is the first step in quiescing the replication system. See “Quiescing Replication Server” on page 104.

---

To suspend log transfer, log in to Replication Server and execute suspend log transfer at the isql prompt by entering:

```
suspend log transfer from {data_server.database | all}
```

where:

- *data\_server* – the data server with the database for which log transfer is to be suspended.
- *database* – the database for which log transfer is to be suspended.
- *all* – instructs Replication Server to suspend log transfer from all RepAgent and disallow future connections for all RepAgent.

Examples of using  
*suspend log transfer*

These examples demonstrate the use of the suspend log transfer command.

- 1 The following command suspends log transfer for the database named pubs2, managed by the TOKYO\_DS data server:

```
suspend log transfer from TOKYO_DS.pubs2
```

- 2 The following command suspends log transfer to the current Replication Server from all RepAgent:

```
suspend log transfer from all
```

In both examples, after the command is executed, affected RepAgent are not shut down and may continue to send some messages to Replication Server. To shut down a RepAgent immediately, log in to Adaptive Server and enter `sp_stop_rep_agent`, with the name of the database for which RepAgent is enabled, and the `nowait` option.

## Resuming log transfer

To reconnect RepAgent to a Replication Server, log in to the Replication Server and enter the resume log transfer command at the isql prompt:

```
resume log transfer from {data_server.database | all}
```

- *data\_server* – the data server with the database for which log transfer is to be resumed.

- *database* – the database for which log transfer is to be resumed and for which the RepAgent connection is to be allowed.
- *all* – allows all RepAgents to connect to this Replication Server.

Examples of using  
*resume log transfer*

These examples demonstrate the use of the resume log transfer command.

- 1 The following command resumes log transfer for the database named *pubs2*, managed by the *TOKYO\_DS* data server:

```
resume log transfer from TOKYO_DS.pubs2
```

- 2 The following command resumes log transfer to this Replication Server from all RepAgent:

```
resume log transfer from all
```

## Using *alter connection* and the *set log transfer* option

Shut down log transfer using *alter connection* with the *set log transfer* option. To shut down log transfer, turn the *set log transfer* option off. For example:

```
alter connection to TOKYO_DS.pubs2  
set log transfer off
```

When log transfer is off, Replication Server removes the *DIST* thread, and RepAgent can no longer log in to Replication Server.

When Replication Server no longer recognizes the primary database, you must reestablish this connection using *rs\_init* or *create connection* before you can use *alter connection* to set log transfer on.

To set log transfer on, turn the *set log transfer* option on. For example:

```
alter connection to TOKYO_DS.pubs2  
set log transfer on
```

## Using counters to monitor RepAgent performance

Adaptive Server provides several counters for monitoring RepAgent performance. You can monitor RepAgent performance data using *sp\_sysmon*. Invoking *sp\_sysmon* clears all accumulated data from the set of counters to be used during the sample interval. At the end of the sample interval, the procedure reads the values in the counters, prints a report, and stops executing.

You can direct `sp_sysmon` to print information for RepAgent counters only or for all Adaptive Server counters. `sp_sysmon` displays RepAgent counter information for each database.

See the *Adaptive Server Enterprise Performance and Tuning Guide* for complete usage and syntax information for `sp_sysmon`.

See Chapter 5, “Using Counters to Monitor Performance,” in the *Replication Server Administration Guide Volume 2* for information about using counters to monitor Replication Server activity.

## Invoking `sp_sysmon`

There are two ways to invoke `sp_sysmon`:

- Using a fixed time interval to provide a sample for a specified number of minutes
- Using the `begin_sample` and `end_sample` parameters to start and stop sampling

### Fixed time intervals

To run `sp_sysmon` for 10 minutes and print information for all counters, use this command:

```
sp_sysmon "00:10:00"
```

To print only the RepAgent section of the report, enter:

```
sp_sysmon "00:10:00", repagent
```

### Using `begin_sample` and `end_sample`

When you use `begin_sample` and `end_sample`, you can invoke `sp_sysmon` to start and end the sample, issue queries, and print results at any point in time. For example, to start and end the sample for the RepAgent group of counters, enter:

```
sp_sysmon begin_sample
go
execute procl
go
sp_sysmon end_sample, repagent
```

## RepAgent counter activity

This section provides sample output from `sp_sysmon`, and a description of what that output means.

### Sample output

```
Replication Agent
-----
Replication Agent: pubs2
Replication Server: NY_RS
```



	per sec	per xact	count	% of total
	-----	-----	-----	-----
Log Scan Summary				
Log Records Scanned	n/a	n/a	103	n/a
Log Records Processed	n/a	n/a	44	n/a
Log Scan Activity				
Updates	n/a	n/a	5	n/a
Inserts	n/a	n/a	5	n/a
Deletes	n/a	n/a	5	n/a
Store Procedures	n/a	n/a	0	n/a
DDL Log Records	n/a	n/a	0	n/a
Writetext Log Records	n/a	n/a	0	n/a
Text/Image Log Records	n/a	n/a	10	n/a
CLRs	n/a	n/a	0	n/a
Checkpoints Processed	n/a	n/a	0	n/a
Transaction Activity				
Opened	n/a	n/a	7	n/a
Committed	n/a	n/a	7	n/a
Aborted	n/a	n/a	0	n/a
Delayed Commit	n/a	n/a	0	n/a
Prepared	n/a	n/a	0	n/a
Maintenance User	n/a	n/a	0	n/a
Log Extension Wait				
Count	n/a	n/a	3	n/a
Amount of time (ms)	n/a	n/a	7822	n/a
Longest Wait (ms)	n/a	n/a	5110	n/a
Average Time (ms)	n/a	n/a	2607.3	n/a
Schema Cache Lookups				
Forward Schema				
Count	n/a	n/a	0	n/a
Total Wait (ms)	n/a	n/a	0	n/a
Longest Wait (ms)	n/a	n/a	0	n/a
Average Time (ms)	n/a	n/a	0.0	n/a
Backward Schema				
Count	n/a	n/a	0	n/a
Total Wait (ms)	n/a	n/a	0	n/a
Longest Wait (ms)	n/a	n/a	0	n/a
Average Time (ms)	n/a	n/a	0.0	n/a
Truncation Point Movement				
Moved	n/a	n/a	0	n/a

Gotten from RS	n/a	n/a	0	n/a
Connections to Replication Server				
Success	n/a	n/a	0	n/a
Failed	n/a	n/a	0	n/a
Network Packet Information				
Packets Sent	n/a	n/a	6	n/a
Full Packets Sent	n/a	n/a	2	n/a
Largest Packet	n/a	n/a	2048	n/a
Amount of Bytes Sent	n/a	n/a	7695	n/a
Average Packet	n/a	n/a	1282.5	n/a
I/O Wait from RS				
Count	n/a	n/a	6	n/a
Amount of Time (ms)	n/a	n/a	766	n/a
Longest Wait (ms)	n/a	n/a	206	n/a
Average Wait (ms)	n/a	n/a	127.7	n/a

-----

## Log scan summary

RepAgent scans all records in the transaction log, but not all scanned records need to be processed and sent to Replication Server. For example, RepAgent does not send records generated by data manipulation language (DML) on tables not marked for replication.

This section reports:

- The number of log records RepAgent has scanned
- The number of log records RepAgent has processed and sent to Replication Server

## Log scan activity

This section provides information about the different kinds of log records processed by RepAgent and sent to the Replication Server. It reports the number of:

- Rows affected by update statements
- Rows affected by insert statements
- Rows affected by delete statements
- Stored procedure executions

- Log records containing DDL to be replicated
- Log records processed generated by a WriteText command
- DML log records processed for a table with text, unitext, or image data
- Compensation log records (CLRs), which are generated when a transaction is partially or fully rolled back
- Checkpoint log records indicate that there was an active transaction at the time this log record was written.

### **Transaction activity**

This section summarizes transaction activity. It reports the number of:

- Transactions opened in the primary database
- Transactions committed
- Transactions aborted
- Transactions found in prepare state
- Transactions opened by the maintenance user

### **Log extension wait**

During normal processing, RepAgent reaches the end of the transaction log. It then waits until further activity resumes in the primary database. This section reports:

- The number of times RepAgent waited for extensions to the transaction log
- The total amount of time, in milliseconds (ms), that RepAgent waited for log extensions
- The longest amount of time, in ms, that RepAgent waited for log extensions
- The average amount of time, in ms, that RepAgent waited for log extensions

## Schema cache lookups

When the structure of an object marked for replication is modified—by alter table, for example—Adaptive Server must log special records in the transaction log that later on will help RepAgent identify the correct schema for the object.

This section describes RepAgent activity scanning forward and backward in the transaction log looking for object schema changes.

### Forward schema

This section reports:

- The number of times RepAgent performed forward scans
- The total amount of time, in ms, that RepAgent spent performing forward scans
- The longest amount of time, in ms, that RepAgent spent performing a forward scan
- The average amount of time, in ms, that RepAgent spent performing a forward scan

### Backward schema

RepAgent performs a backward scan when DDL is performed inside a transaction. This section reports:

- The number of times RepAgent spent performing backward scans
- The total amount of time, in ms, that RepAgent performed backward scans
- The longest amount of time, in ms, that RepAgent spent performing a backward scan
- The average amount of time, in ms, that RepAgent spent performing a backward scan

## Truncation point movement

This section reports:

- The number of times RepAgent moved the secondary truncation point
- The number of times RepAgent asked Replication Server for a new truncation point

## Connections to Replication Server

This section reports:

- The number of successful connections to Replication Server
- The number of unsuccessful connections to Replication Server

## Network packet information

This section reports:

- The number of packets sent to Replication Server
- The number of full packets sent to Replication Server
- The largest packet sent to Replication Server
- The number of bytes sent to Replication Server
- The average packet size

## I/O wait from Replication Server

After RepAgent generates LTL, RepAgent sends it to Replication Server. To do this, it uses Open Client capabilities. This section reports:

- The number of times RepAgent has sent a batch to Replication Server
- The total amount of time, in ms, that RepAgent has spent processing results from Replication Server
- The longest elapsed time, in ms, that RepAgent has spent processing results from Replication Server
- The average elapsed time, in ms, that RepAgent has spent processing results from Replication Server



# Managing Routes

This chapter describes creating and managing routes between Replication Servers.

Topic	Page
Overview	137
Before you begin	138
Routing schemes	140
Creating routes	144
Suspending and resuming routes	150
Changing routes	151
Dropping routes	158
Upgrading routes	160
Monitoring routes	161

## Overview

A route is a one-way message stream from a source Replication Server to a destination Replication Server. From each source Replication Server, you create one route for each destination Replication Server, no matter how many databases are managed by the source or destination Replication Servers.

Routes carry:

- Data modification commands and applied functions or applied stored procedures from primary databases managed at the source Replication Server to replicate databases managed at the destination Replication Server
- System table modification commands from a source Replication Server RSSD to a destination Replication Server RSSD

- Request functions or request stored procedures from replicate databases to primary databases (in this case, the source is the replicate Replication Server and the destination is the primary Replication Server).

When you create a route, the source Replication Server:

- Creates an RSI outbound queue to hold messages for the destination site
- Starts an RSI thread that logs in to the destination Replication Server and transfers transactions from the RSI outbound queue to the destination Replication Server

## Before you begin

Before you create or modify routes, be sure you have carefully determined where routes are needed in your system. As part of the design process, you must know where each source Replication Server and its destination Replication Servers reside.

Identify which routes are direct and which are indirect. Indirect routes carry messages to destination Replication Servers through one or more intermediate Replication Servers. Using direct versus indirect routes can have a noticeable effect on system performance.

Refer to the *Replication Server Design Guide* for details on routing and performance issues. Also see “Routing schemes” on page 140 for a general discussion of direct and indirect routes.

Once you have determined your routing scheme, you can set up the required routes based on these rules:

- Replication Servers that manage databases containing primary data require direct or indirect routes to the Replication Servers that manage databases with subscriptions for the data.
- Replication Servers that manage replicate databases where request functions originate require direct or indirect routes to the Replication Server managing the primary database. If no replicated functions originate in the replicate database, a route from a replicate to a primary Replication Server is not required.
- Each route in an indirect route must be a direct route.

See “Indirect routes” on page 141 for examples of indirect routes.



- You customize function strings for system functions with class scope at the primary Replication Server for the function-string class. In this instance, you must create routes from the primary Replication Server to the Replication Server managing the databases that use the function strings.  
See “System functions with function-string-class scope” on page 16 in the *Replication Server Administration Guide Volume 2* for more information.
- You customize error classes at the primary Replication Server. In this instance, you must create routes from the primary Replication Server to the Replication Server managing the databases that use the error mappings.
- A Replication Server that you plan to assign as the new primary site for a function-string class or error class, using the `move primary` command, has the following requirements:
  - It must have routes to and from the Replication Server that is the current primary site for the class, and
  - It must have routes to all the same Replication Servers as the Replication Server that is the current primary site for the class

See “Changing the primary Replication Server for an error class” on page 211 in the *Replication Server Administration Guide Volume 2* for more information. See also “Primary site for a function-string class” on page 29 in the *Replication Server Administration Guide Volume 2*.

## Routing preparations

Before creating and modifying routes, you need to:

- Make sure the source Replication Server is running.
- If you are creating a direct route, define the destination Replication Server in the `interfaces` file at the site of the source Replication Server.

You should also have an `interfaces` file entry for the RSSD of the destination Replication Server.

- Make sure that the `RepAgent` thread for the source Replication Server RSSD is running.
- Make sure that the destination Replication Server and any intermediate Replication Servers in the route are running.

## Routing schemes

Replication Server supports direct and indirect routes. Each type of route is described in the following sections.

Figure 6-1 and Figure 6-2 each show a seven-site enterprise with a single primary site and six replicate sites. Each replicate site has a route originating at the primary site.

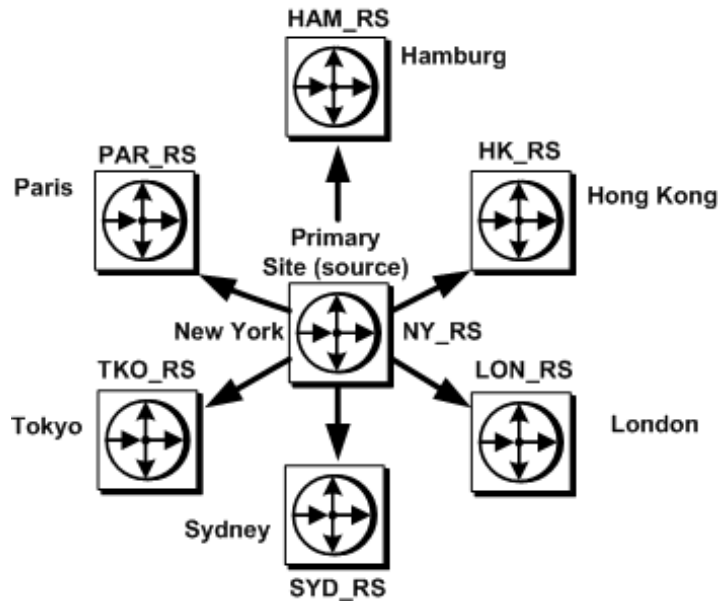
In Figure 6-1, all six routes from the primary site are direct. Thus, the primary Replication Server has six stable queues and six RSI threads connected through the network to the six replicate sites.

In Figure 6-2, only two routes from the primary site are direct; four are indirect. The two intermediate sites each have two direct routes. Table 6-1 lists the routes in Figure 6-2.

### Direct routes

A route with no intermediate sites is called a direct route. A system with direct routes results in network connections between source and destination Replication Servers.

For example, in Figure 6-1, a seven-site enterprise is shown in a star configuration, with one primary site and six replicate sites. If the replicate site TKO\_RS is to submit request functions to the primary site NY\_RS, your system would also require a direct route from TKO\_RS to NY\_RS, in addition to the direct route from NY\_RS to TKO\_RS.

**Figure 6-1: Sites connected with direct route configuration**

## Indirect routes

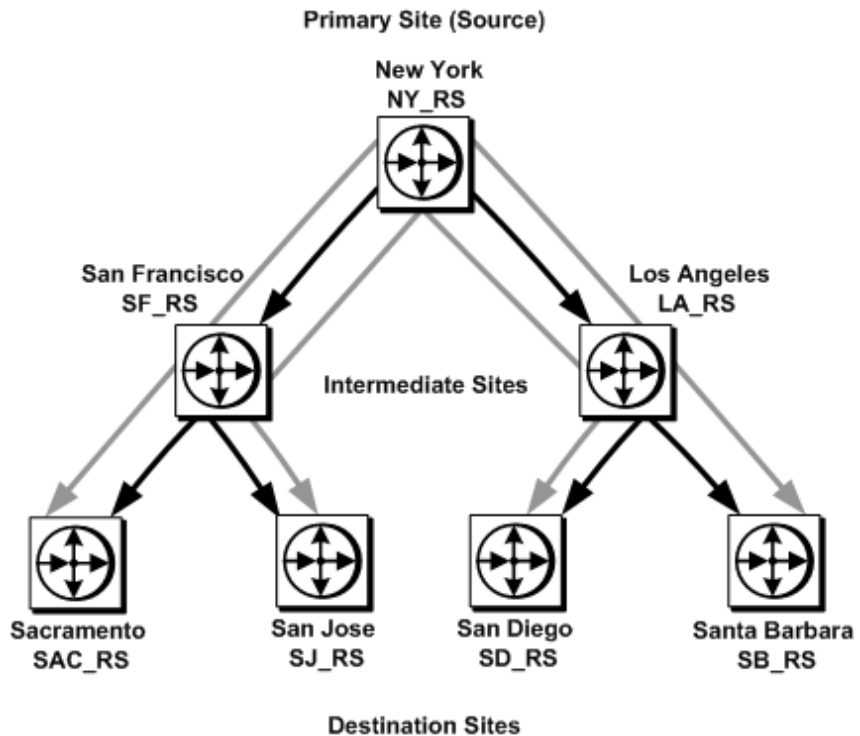
A route with intermediate sites is called an indirect route.

For example, in Figure 6-2, NY\_RS to SAC\_RS is an indirect route, based on the direct routes NY\_RS to SF\_RS and SF\_RS to SAC\_RS. In an indirect route, the source Replication Server sends messages for the destination Replication Server to an intermediate Replication Server, which makes use of a route (direct or indirect) to the destination Replication Server.

To create an indirect route, you create direct routes between each successive Replication Server along the intended indirect route. Once all the direct routes are in place, then you create the indirect route itself. See “Creating routes” on page 144 for details.

For example, to create the indirect route NY\_RS to SAC\_RS, first create the direct routes NY\_RS to SF\_RS and SF\_RS to SAC\_RS. Then create the indirect route based on the existing direct routes.

**Figure 6-2: Sites connected with indirect routes in a hierarchical configuration**



By setting up indirect routes, you reduce the amount of processing at the primary site and distribute the load among intermediate Replication Servers.

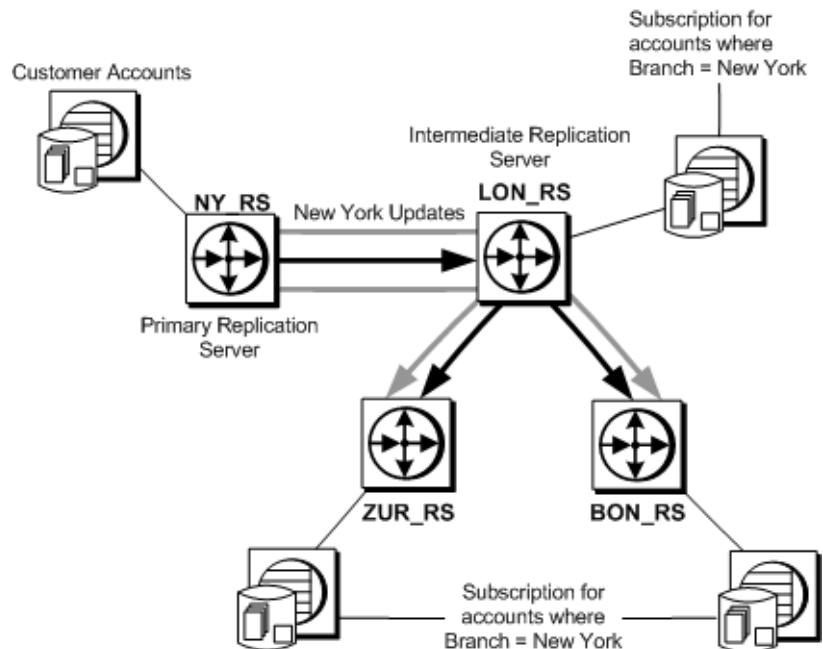
**Table 6-1: Direct and indirect routes between sites in Figure 6-2**

Direct routes	Indirect routes
NY_RS to SF_RS	NY_RS to SAC_RS
NY_RS to LA_RS	NY_RS to SJ_RS
SF_RS to SAC_RS	NY_RS to SD_RS
SF_RS to SJ_RS	NY_RS to SB_RS
LA_RS to SD_RS	
LA_RS to SB_RS	

When you use indirect routes, the primary Replication Server can route portions of subscriptions that are common to destination sites through the same intermediate site. When subscriptions overlap, the primary Replication Server is required to send only one message per row modification to the intermediate Replication Server that is common to the destination sites.

For example, in Figure 6-3, the intermediate Replication Server in LON\_RS receives row modification changes for customer accounts whenever changes occur at the bank headquarters in New York. The New York modifications are also required at branch bank replicate sites in Zurich and Bonn. Because LON\_RS is set up to distribute changes to ZUR\_RS and BON\_RS, the NY\_RS primary Replication Server sends only one copy of each change to LON\_RS. The number of direct routes is also reduced through the use of the two indirect routes, NY\_RS to ZUR\_RS and NY\_RS to BON\_RS.

**Figure 6-3: Sites with overlapping subscriptions**



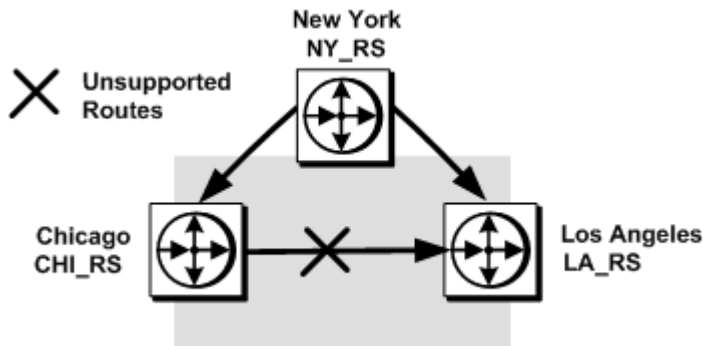
Although indirect routes are helpful for distributing computing resources among sites on the network, overall propagation of data is slowed somewhat because messages are queued by more than one Replication Server. It is better to use direct routes when there are few replicate sites. When using indirect routes, minimize the number of intermediate sites to obtain the best propagation times.

## Unsupported routing schemes

An intermediate Replication Server can accept transactions from one or more Replication Servers. Replication Server, however, does not support routing schemes in which routes diverge from the same source Replication Server, then converge at the same intermediate or destination Replication Server.

For example, in Figure 6-4, only one route from NY\_RS to LA\_RS can be supported. If the route from NY\_RS to LA\_RS is supported, then the route between CHI\_RS to LA\_RS is not supported.

**Figure 6-4: Example of supported and unsupported routes**



## Creating routes

You create routes at the source Replication Server. As soon as you create a direct route between a source and destination Replication Server, the source Replication Server:

- Creates an RSI outbound stable queue to hold messages for the destination site, and

- Starts an RSI thread that logs in to the destination or next Replication Server in the route.

---

**Note** You can create a route from a version 15.0 Replication Server to an older Replication Server (version 11.03 or later).

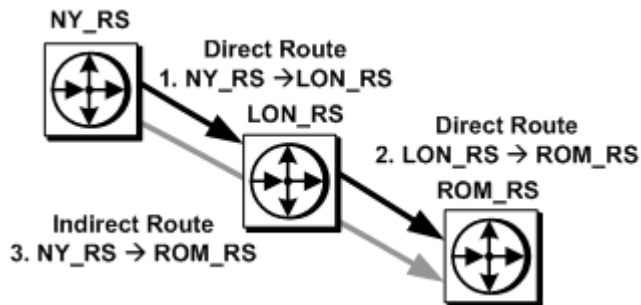
---

When you create either direct or indirect routes, the destination Replication Server creates and materializes subscriptions at the destination site for the replicated RSSD system tables. This process lets the destination Replication Server receive available replication definitions and function classes. Refer to Chapter 2, “Replication Server Technical Overview” for details.

You cannot create an indirect route (1 to 3) unless you have already created two direct routes (1 to 2 and 2 to 3). You also must set up the routes in the correct order, as shown in Figure 6-5. For Replication Server to be able to begin transferring system information to the destination Replication Server, you must create direct routes before you create an indirect route.

When you create an indirect route, Replication Server does not create an RSI queue. The indirect route uses the RSI outbound queues of the direct route segments that compose the indirect route.

**Figure 6-5: Order for creating direct and indirect routes**



## Using the *create route* command

You can create routes in Sybase Central or with the `create route` command.

The syntax for the `create route` command is:

```
create route to dest_replication_server{
  set next site [to] thru_replication_server|
  [set username [to] user]
  [set password [to] passwd]
  [set route_param to 'value']
  [set security_param to 'value']}
```

When creating routes:

- Supply the login name, password, and other parameters for direct routes only.
- Before you create a direct route, create its login name and password in the destination Replication Server. Optionally, you can have the `rs_init` utility create this user.
  - If you are enabling network-based security and unified login, user name and password are optional. The default user name is the principal user name, which is specified by the `-S` flag when you log in to Replication Server or start Replication Server. Refer to “Establishing the principal user” on page 216 for more information about network-based security and the principal user.
  - If you create a route with a *user* and *passwd* that do not exist at the destination Replication Server, add or change the *user* and *password* at that destination. See also “Changing an indirect route to a direct route” on page 153.
  - If you are establishing a direct route from the current Replication Server to the destination Replication Server, do not use the next site clause.
- Enter one create route command at a time, to ensure you have made no mistakes. Wait for a route to become valid before creating the next one.

If you do make a mistake, drop the route and re-create it only as a last resort. Include the `with nowait` option with the drop route command. Since the route has not been created, its current state requires that you use the `with nowait` option to drop it. See “Dropping routes” on page 158.

When you create a route, you can accept the default values for configuration parameters that manage memory size, the size of the amount of data that can be sent over the route at one time, time-outs, and synchronization intervals. You can also set your own values when you create or alter the route.



Table 6-2 displays the route configuration parameters. If network-based security is enabled at your site, you can also configure security parameters for routes. Refer to “Managing network-based security” on page 210. See “Configuration parameters that affect performance” on page 131 in the *Replication Server Administration Guide Volume 2* for a list and discussion of route parameters that affect performance.

**Table 6-2: Configuration parameters affecting routes**

<b>route_param</b>	<b>value</b>
disk_affinity	Specifies an allocation hint for assigning the next partition. Enter the logical name of the partition to which the next segment should be allocated when the current partition is full. Values are “ <i>partition_name</i> ” and “off.” Default: off
rsi_batch_size	The number of bytes sent to another Replication Server before a truncation point is requested. Default: 256K Minimum: 1K Maximum: 128MB
rsi_fadeout_time	The number of seconds of idle time before Replication Server closes a connection with a destination Replication Server. Default: -1 (Replication Server does not close the connection)
rsi_packet_size	Packet size, in bytes, for communications with other Replication Servers. The range is 1024 to 16384. Default: 2048 bytes
rsi_sync_interval	The number of seconds between RSI synchronization inquiry messages. The Replication Server uses these messages to synchronize the RSI outbound queue with destination Replication Servers. The value must be greater than 0. Default: 60 seconds
rsi_xact_with_large_msg	Specifies route behavior if a large message is encountered. This parameter is applicable only to direct routes where the site version at the replicate site is 12.1 or earlier. Values are “skip” and “shutdown.” Default: shutdown
save_interval	The number of minutes that the Replication Server saves messages after they have been successfully passed to the destination Replication Server. Default: 0 minutes

## Examples of creating direct and indirect routes

You need to create the direct routes from the primary Replication Server to the intermediate Replication Server and from the intermediate Replication Server to the destination Replication Server before you can create an indirect route.

The following examples are based upon Figure 6-2.

- 1 To create the direct route NY\_RS to SF\_RS in Figure 6-2, enter this command in the primary Replication Server, NY\_RS:

```
create route to SF_RS
set username SF_rsi_user
set password SF_rsi_ps
```

- 2 To create the direct routes SF\_RS to SAC\_RS and SF\_RS to SJ\_RS in Figure 6-2, enter these commands in the intermediate Replication Server, SF\_RS:

```
create route to SAC_RS
set username SAC_rsi_user
set password SAC_rsi_ps
create route to SJ_RS
set username SJ_rsi_user
set password SJ_rsi_ps
```

- 3 After these direct routes are created, you can create indirect routes through them. The following example creates the indirect routes from the primary site NY\_RS to sites SAC\_RS and SJ\_RS, through the intermediate site, SF\_RS. Enter these commands in the primary Replication Server, NY\_RS:

```
create route to SAC_RS
set next site SF_RS
create route to SJ_RS
set next site SF_RS
```

### An example of creating a route and configuring parameters

This example is based on Figure 6-2. To set the `rsi_packet_size` to 4096 bytes for the route to SF\_RS, enter:

```
create route to SF_RS
set username SF_rsi_user
set password SF_rsi_ps
set rsi_packet_size to '4096'
```

### Configuring a Replication Server to manage primary tables

If you want to add a route from a Replication Server that was previously configured as a replicate-only Replication Server, you must first set up the RepAgent for the Replication Server RSSD. Any database that functions as a primary database also requires a RepAgent.

To set up RepAgent for the RSSD, follow these steps:

At the Replication Server

- 1 Create a RepAgent user so that RepAgent can log in to Replication Server. Use the create user command where *ra\_user\_name* is the name of the RepAgent user and *ra\_password* is the RepAgent's password:

```
create user ra_user_name
set password {ra_password | null}
```

Grant this user connect source permission, using the grant command:

```
grant connect source to ra_user_name
```

If the Replication Server already manages a primary database, you can use the “RepAgent user” that already exists for the new primary database.

- 2 Execute alter connection, using the log transfer on option:

```
alter connection to data_server.database
set log transfer to 'on'
```

At the Adaptive Server

- 1 If the name of the Adaptive Server has not yet been defined, you must define it using the following command where *lname* is the RSSD's name:

```
sp_addserver lname, local
```

- 2 If RepAgent threads have not been enabled for the Adaptive Server, you must enable them:

```
sp_configure 'enable rep agent threads'
```

- 3 Configure RepAgent for the RSSD with the `sp_config_rep_agent` system procedure:

```
sp_config_rep_agent dbname, 'enable', 'rs_name',
'rs_user_name', 'rs_password'
```

Refer to “Configuring RepAgent” on page 114 for detailed instruction on configuring RepAgent.

---

**Note** The “*rs\_user\_name*” and “*rs\_password*” configured at the Adaptive Server must be the same as the “*ra\_user\_name*” and “*ra\_password*” created at the Replication Server in step 1.

---

- 4 Start RepAgent:

```
sp_start_rep_agent dbname
```

## Suspending and resuming routes

When you alter a direct route, change its topology, or perform some other maintenance to a remote site, you must suspend the route so that messages are no longer sent to the destination Replication Server. After maintenance is completed for the route, you can then reactivate the route to resume activity.

You can suspend and resume routes in Sybase Central or with the RCL commands `suspend route` and `resume route`.

The `suspend route` and `resume route` RCL commands are described in this section.

### Using the *suspend route* command

The `suspend route` command suspends a route to another Replication Server. While a route is suspended, no messages are sent to the destination Replication Server, and the messages for the Replication Server are held in a stable queue. The syntax for the `suspend route` command is:

```
suspend route to dest_replication_server
```

For example, to suspend the route to the `CHI_RS` Replication Server, enter:

```
suspend route to CHI_RS
```

### Using the *resume route* command

The `resume route` command resumes a suspended route. Resuming a route allows the source Replication Server to begin sending queued messages to the destination Replication Server. You can also use this command to resume a route that was suspended automatically as the result of an error. The syntax for the `resume route` command is:

```
resume route to dest_replication_server
```

For example, to resume the route to the `CHI_RS` Replication Server, enter:

```
resume route to CHI_RS
```

## Changing routes

You can change a direct route's topology, user name, password, and certain configuration parameters from Sybase Central or with the alter route command. You cannot change an indirect route's parameters with alter route.

The syntax for alter route is:

```
alter route to dest_replication_server{
  set next site [to] thru_replication_server |
  set username [to] 'user' set password [to] 'passwd' |
  set password [to] 'passwd' |
  set route_param [to] 'value' |
  set security_param [to] 'value' |
  set security_services [to] 'default'}
```

Refer to “Managing network-based security” on page 210 for information about configuring security parameters for routes.

This section provides procedures and examples for using alter route to change a route's topology, user name, and route configuration parameters. There is also a routing modification example.

Follow these steps when altering a route:

- 1 Suspend the route.
- 2 Execute alter route.
- 3 Resume the route. You must resume the route for the changes to take effect.

## Changing route topology

You can modify a route's topology by:

- Changing a direct route to an indirect route
- Changing the next intermediate site for an indirect route
- Changing an indirect route to a direct route

## Changing a direct route to an indirect route

To change an existing direct route to an indirect route, perform these steps:

- 1 At the source Replication Server, from which the direct route originates, enter:

```
suspend route to dest_replication_server
```

- 2 At each Replication Server that manages a database with a RepAgent, enter:

```
suspend log transfer from all
```

and follow the instructions in “Quiescing a replication system” on page 104. This procedure quiesces the replication system so that messages will be redirected to your new routing configuration without error.

- 3 Create any additional routes that the new indirect route will use. See “Creating routes” on page 144 for details.
  - If the current Replication Server does not already have a direct route to the Replication Server that you will specify as the intermediate site for the new indirect route, create the route.
  - If the Replication Server that you will specify as the intermediate site for the new indirect route does not already have a direct or indirect route to the destination site, create the route.
- 4 For the direct route you are changing to an indirect route, enter the following command at the source Replication Server where *dest\_replication\_server* is the destination Replication Server for the route you are altering, and *thru\_replication\_server* is the intermediate Replication Server for the route:

```
alter route to dest_replication_server  
set next site [to] thru_replication_server
```

- 5 Resume log transfer connections by entering the following command at each Replication Server where you previously suspended log transfer:

```
resume log transfer from all
```

- 6 At the source Replication Server, resume the suspended route by entering the following command:

```
resume route to dest_replication_server
```

### Changing the next intermediate site for an indirect route

To change the next intermediate site for an existing indirect route, perform the following steps.

- 1 Enter the following command at the source Replication Server, from which the direct route originates:

```
suspend route to dest_replication_server
```

- 2 At each Replication Server that manages a database with a RepAgent, enter:

```
suspend log transfer from all
```

and follow the instructions in “Quiescing a replication system” on page 104. This procedure quiesces the replication system so that messages will be redirected to your new routing configuration without error.
- 3 Create any additional routes that the indirect route will use. See “Creating routes” on page 144 for details.
  - If the current Replication Server does not already have a direct route to the Replication Server that you will specify as the new intermediate site for the indirect route, create the route.
  - If the Replication Server that you will specify as the new intermediate site for the indirect route does not already have a direct or indirect route to the destination site, create the route.
- 4 For the indirect route for which you are specifying a new intermediate Replication Server, enter the following command at the source Replication Server where *dest\_replication\_server* is the destination Replication Server for the route you are altering, and *thru\_replication\_server* is the new intermediate Replication Server for the route:

```
alter route to dest_replication_server  
set next site thru_replication_server
```
- 5 Resume log transfer connections by entering the following command at each Replication Server where you previously suspended log transfer:

```
resume log transfer from all
```
- 6 Resume the suspended route by entering the following command at the source Replication Server:

```
resume route to dest_replication_server
```

## Changing an indirect route to a direct route

To change an existing indirect route to a direct route, perform the following steps.

- 1 Enter the following command at the source Replication Server, from which the indirect route originates:

```
suspend route to dest_replication_server
```

- 2 At each Replication Server that manages a database with a RepAgent, enter:

```
suspend log transfer from all
```

and follow the instructions in “Quiescing a replication system” on page 104. This procedure quiesces the replication system so that messages will be redirected to your new routing configuration without error.
- 3 For the indirect route you are changing to a direct route, enter the following command at the source Replication Server where *dest\_replication\_server* is the destination Replication Server for the route you are altering, and *user* and *passwd* are the RSI user login name and password to use for the direct route:

```
alter route to dest_replication_server
set username user set password passwd
```
- 4 Resume log transfer connections by entering the following command at each Replication Server where you previously suspended log transfer:

```
resume log transfer from all
```
- 5 Resume the suspended route by entering the following command at the source Replication Server:

```
resume route to dest_replication_server
```

## Changing the password for the RSI user for a direct route

To change the password for the RSI user for an existing direct route, perform the following steps.

- 1 Suspend each direct route from the source Replication Server by entering:

```
suspend route to dest_replication_server
```
- 2 At the source Replication Server, enter the following command where *dest\_replication\_server* is the destination Replication Server for the route you are altering, and *passwd* is the password to use for the RSI user login name:

```
alter route to dest_replication_server
set password passwd
```
- 3 Resume each suspended route from the source by entering:

```
resume route to dest_replication_server
```



## Changing parameters affecting direct routes

After a route is created, you can change its configuration parameters with Sybase Central or the `alter route` command. Refer to Table 6-2 on page 147 for a list and descriptions of configuration parameters that affect routes.

To change default configuration parameters for all routes originating at the source Replication Server, use the `configure replication server` command. Refer to “Changing configuration parameters for all routes” on page 155 for more information.

Here is an example of using `alter route` to change the `rsi_sync_interval` parameter to 120 seconds. To execute the command, log in to the source Replication Server and perform these steps:

- 1 Suspend the route. Enter:

```
suspend route to dest_replication_server
```

- 2 Execute the `alter route` command. Enter:

```
alter route to dest_replication_server  
set rsi_sync_interval to '120'
```

- 3 Resume the suspended route by entering:

```
resume route to dest_replication_server
```

Configuration changes take effect after you resume the route.

## Changing configuration parameters for all routes

To set default configuration parameters for all routes originating at the source Replication Server, use the `configure replication server` command. Table 6-2 on page 147 has a list and descriptions of configuration parameters that you can set.

Configuration parameters set for individual routes with `alter route` override default parameters set with `configure replication server`. Thus, you can set default parameters with `configure replication server` and then customize settings for individual routes with `alter route`.

The syntax for changing route parameters with `configure replication server` is:

```
configure replication server  
set route_param to 'value'
```

Here is an example of using configure replication server to change the `rsi_save_interval` parameter to 2 minutes for all routes originating at the Replication Server. To execute the command, log in to the source Replication Server and perform the following steps:

- 1 Suspend all routes from the source Replication Server. For each route, enter:

```
suspend route to dest_replication_server
```

- 2 Execute the configure replication server command:

```
configure replication server  
set rsi_save_interval to '2'
```

- 3 Resume suspended routes from the source Replication Server. For each route, enter:

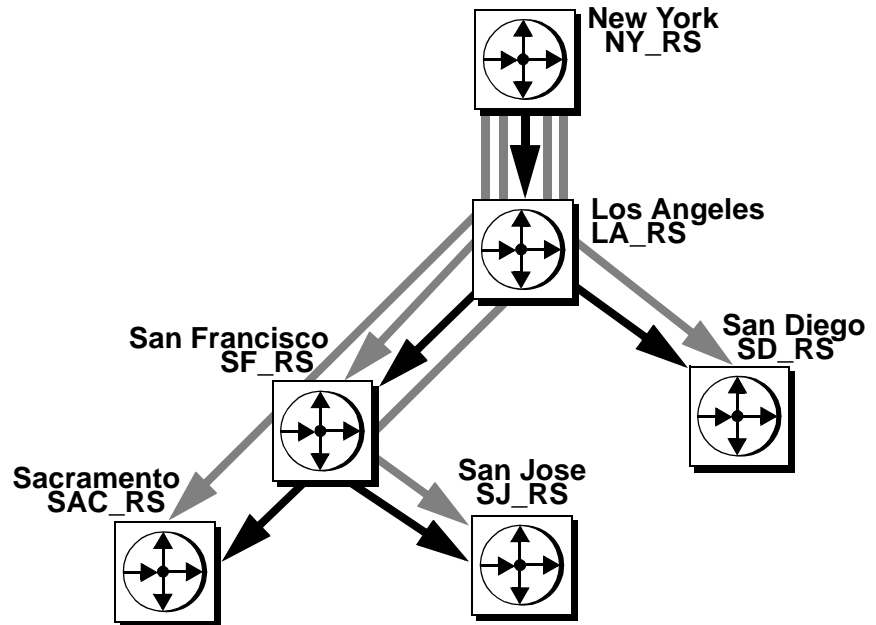
```
resume route to dest_replication_server
```

Configuration changes take effect after you resume the routes.

## Routing modification example

Figure 6-6 revises the routes illustrated in Figure 6-2 on page 142. LA\_RS becomes an intermediate site between NY\_RS and SF\_RS, while direct and indirect routes to SB\_RS are dropped.

Figure 6-6: Indirect routes altered



Here's how you would revise the routing scheme shown in Figure 6-2 to resemble the scheme in Figure 6-6.

- At each Replication Server that manages a database with a RepAgent, enter:
 

```
suspend log transfer from all
```

 and follow the instructions in "Quiescing a replication system" on page 104. This procedure quiesces the replication system so that messages will be redirected to your new routing configuration without error.
- LA\_RS needs a direct route to SF\_RS; create one by entering the following command at Replication Server LA\_RS:
 

```
create route to SF_RS
set username SF_rsi_user
set password SF_rsi_ps
```
- LA\_RS requires indirect routes to SAC\_RS and SJ\_RS, through SF\_RS. Creating these routes instructs LA\_RS to send messages to SF\_RS that are destined for SAC\_RS and SJ\_RS. SF\_RS already has direct routes to SAC\_RS and SJ\_RS. Enter the commands in Replication Server LA\_RS:

```
create route to SAC_RS
set next site SF_RS
create route to SJ_RS
set next site SF_RS
```

- 4 The primary Replication Server, NY\_RS, was previously configured with indirect routes through SF\_RS to SAC\_RS and SJ\_RS. Alter those routes so that Replication Server LA\_RS is the next Replication Server. Enter these commands in Replication Server NY\_RS:

```
alter route to SAC_RS
set next site LA_RS
alter route to SJ_RS
set next site LA_RS
```

- 5 The direct route from the primary Replication Server, NY\_RS, to SF\_RS needs to be changed to an indirect route, with LA\_RS as the intermediate Replication Server. Enter these commands in Replication Server NY\_RS:

```
alter route to SF_RS
set next site LA_RS
```

- 6 At each Replication Server where you previously suspended log transfer, resume log transfer connections to each Replication Server by entering:

```
resume log transfer from all
```

Refer to Chapter 4, “Managing a Replication System” for more information on resuming log transfer.

- 7 Remove the indirect route from NY\_RS to SB\_RS. Enter this command in NY\_RS:

```
drop route to SB_RS
```

- 8 Remove the direct route from LA\_RS to SB\_RS. Enter this command in LA\_RS:

```
drop route to SB_RS
```

The indirect route from NY\_RS to SD\_RS, through LA\_RS, is intact.

## Dropping routes

You can drop routes from Sybase Central or from the command line with the drop route command.

Dropping a route closes the route from the Replication Server where you execute the command to a specified remote Replication Server. It performs the following actions on participating Replication Servers:

- Drops system table subscriptions.
- If the route is direct, the outbound stable queue is dropped and the RSI thread is stopped.
- Deletes information regarding the route.

You cannot drop the route if:

- It is a direct route used by any indirect routes to additional destination Replication Servers.
- The source Replication Server has replication definitions that are subscribed to by the destination Replication Server.
- The source Replication Server is designated as the primary site of a function-string class or error class. The primary site of a derived function-string class is the same as its parent class.

You can monitor the status of the route while it is being dropped:

- In Sybase Central, view status information in the right pane of the Sybase Central main window.
- From the command line, execute the `rs_helproute` stored procedure.

## Using the *drop route* command

The syntax for the `drop route` command is:

```
drop route to dest_replication_server [with nowait]
```

The `with nowait` option instructs Replication Server to close the route even if it is unable to communicate with the destination Replication Server.

---

**Warning!** Use the `with nowait` clause only if you do not intend to ever use the destination Replication Server, or if you must drop the route from the source Replication Server while the destination Replication Server is unavailable, or if you are attempting to add or change login names and passwords for direct routes. Avoid using the `with nowait` clause whenever possible.

---

After you use `drop route` with the `with nowait` clause, use the `sysadmin purge_route_at_replicate` command to remove all references to a primary Replication Server from the Replication Server at the replicate site.

## Using the `sysadmin purge_route_at_replicate` command

The `sysadmin purge_route_at_replicate` command removes all subscriptions and route information originating from a specified primary Replication Server after a route is dropped from that server. Before you execute this command, drop the route from the replicate Replication Server to the primary Replication Server, if it exists. The Replication Server performs a validation check before it processes the command.

Execute `sysadmin purge_route_at_replicate` at the replicate Replication Server, using the following syntax where *replication\_server* is the primary Replication Server:

```
sysadmin purge_route_at_replicate, replication_server
```

## Upgrading routes

The route version is the earliest site version of the source and destination Replication Server. After you upgrade the source and destination Replication Servers on either end of a route to version 11.5 or later and also set their site versions to a higher Replication Server version, you need to upgrade the route. Upgrading the route allows the Replication Servers to exchange information about newer software features.

Upgrading a route rematerializes data in system tables, making information associated with new features available to a newly upgraded Replication Server. After upgrading, new types of information that were not previously allowed can be exchanged.

To display the current version number of routes that originate or terminate at a Replication Server, use the `admin show_route_versions` command.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage information of `admin show_route_versions` command.

There are two possible scenarios for route upgrade:

- If new features have not been used at the source Replication Server, use `sysadmin fast_route_upgrade` to upgrade routes.
- In all other cases, use the commands `route_upgrade`, `route_upgrade_recovery`, and `route_upgrade_status` to upgrade routes.

You cannot downgrade a route after you have upgraded it.

See “Mixed-version replication systems” on page 18 for more information about site versions and system versions.

See the installation and configuration guides for your platform for more information about upgrading routes and setting the site version for a Replication Server.

## Monitoring routes

Routes may display different statuses at different times. When you create a route, the destination Replication Server subscribes to the source Replication Server system tables. Depending on the volume of your data, it may take several minutes for subscriptions to materialize. Dropping a route also may take some time.

- You can use the `admin who` command to display thread status information.
- For comprehensive status information, including the current state of routes you are creating, use `rs_helproute`, described in “Using the `rs_helproute` stored procedure” on page 162.

## Displaying RSI thread status using *admin who*

To view status information about RSI threads, use the `admin who` command:

- `admin who` displays all threads in the system, including RSI threads.
- `admin who, rsi` displays the status of the RSI thread, which Replication Server starts to submit information to other Replication Servers.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for detailed thread status information and `admin who` command.

## Using the *rs\_helproute* stored procedure

Execute the Adaptive Server stored procedure *rs\_helproute* in the RSSD at the source or destination Replication Server for the route. The syntax for the *rs\_helproute* stored procedure is:

```
rs_helproute [replication_server]
```

If you specify the name of a Replication Server, *rs\_helproute* returns information only for routes for which the named Replication Server is a source or destination. Otherwise, it returns information for all routes for which the current Replication Server is a source or destination.

*rs\_helproute* returns two types of information:

- Route status, which reflects the state of the route at the site where *rs\_helproute* is executed. A route is valid when *rs\_helproute* at both source and destination returns “Active.”

Other route status values are:

- Being created
- Being dropped
- Being dropped with nowait
- List of system table subscriptions, which tells you the system table subscriptions that are being created. If a route is being dropped, it tells you which subscriptions are being dropped.

If no system table subscriptions are listed, the route has been created and is in working order.

Refer to the *Replication Server Troubleshooting Guide* for information about correcting route creation problems.



# Managing Database Connections

This chapter describes connecting databases to a replication system and managing those database connections.

Topic	Page
Preparing databases for replication	163
Managing maintenance user login names	165
Creating database connections	167
Altering database connections	170
Dropping database connections	187
Monitoring database connections	188

## Preparing databases for replication

Before you can add databases to a replication system, you need to prepare them so that Replication Server can distribute the primary data and maintain the replicated data stored in them.

- If your databases are managed by Sybase Adaptive Servers:
  - Use Sybase Central or `rs_init` to prepare Adaptive Server databases for use with Replication Server.
  - See the *Replication Server Installation Guide* and *Replication Server Configuration Guide* for more information on `rs_init`.
- If your databases are managed by non-Sybase data servers:
  - Refer to the *Replication Server Design Guide* for required preparations. In addition, to find out how to prepare your database for the heterogeneous datatype support (HDS) feature, see the *Replication Server Configuration Guide* for your platform. HDS enables the translation of primary database column values of one datatype to another datatype acceptable to the replicate database.
  - See “Translating datatypes using HDS” on page 317 for more information about HDS.

When you are connecting a new database to an existing system, always conduct a careful review and analysis of how the database will fit into your system. Determine which other processes are required for the database, and designate required names and login names for these processes.

If you anticipate that an existing “replicate-only” database may in the future be the source of replicated function delivery or contain primary data, you can set up the database so that it can manage primary tables. You can then avoid upgrading the replicate-only database in the future.

## Steps in preparing databases for replication

---

**Note** To prepare non-Sybase databases for replication, use instructions in your Sybase Replication Agent documentation, the *Replication Server Configuration Guide* for your platform, and the *Replication Server Design Guide* to perform these steps.

---

To prepare Adaptive Server databases for replication, use Sybase Central or `rs_init` to perform these steps:

- Create the `rs_lastcommit` system table.
- Load the `rs_update_lastcommit` and `rs_get_lastcommit` stored procedures (for both primary and replicate databases) and the `rs_marker` stored procedure (for primary databases only).
- Create the `rs_threads` system table.
- Load the `rs_initialize_threads` and `rs_update_threads` stored procedures for the database.
- Create the maintenance user login name and verifies that the maintenance user can log in to the database. For details, see “Managing maintenance user login names” on page 165.
- Create a connection from Replication Server to the database, allowing Replication Server to manage the database.
- If the database has primary data, Sybase Central or `rs_init`:
  - Enables RepAgent at the Adaptive Server.
  - Enables and configures RepAgent at the database.

- Sets the secondary truncation point to “valid” in the Adaptive Server database, preventing Adaptive Server from truncating database log records before RepAgent has read them.
- Creates the RepAgent user name and password in the Replication Server, if necessary.
- Starts RepAgent.

Refer to the Replication Server installation and configuration guides for your platform for details on each step.

## Upgrading an existing Adaptive Server database

You may need to upgrade a database to work with the latest version of Replication Server so that you can use newer features. Use `rs_init` to upgrade a database.

Upgrading a database ensures that the database maintenance user has the Replication role and the necessary permissions (update, insert, and delete) in the database. The Replication role gives the maintenance user authorization to execute any necessary replication-related Adaptive Server commands.

You can check the authorizations that have been granted to a database by using the `sp_displaylogin` system procedure in the database.

To grant the Replication role to the maintenance user, execute the following system procedure in the database:

```
sp_role "grant", replication_role, maintenance_user
```

If you need to grant permissions on the tables in the database, execute the following command in the database for each table:

```
grant all on table_name to maintenance_user
```

## Managing maintenance user login names

To update replicated data, Replication Server logs in to the 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.

Initially, Sybase Central or `rs_init` creates the login name for the maintenance user and adds the user to the replicate database. For details, refer to the Replication Server installation and configuration guides for your platform.

The maintenance user login name and password are provided to Replication Server with the `create connection` command for the database. Sybase Central or the `rs_init` program executes this command automatically. If you change the password for the login name in the data server, use Sybase Central or the `alter connection` command to change the password for the Replication Server connection.

## Finding the current maintenance user

To determine the login name that is currently assigned as maintenance user for a database, you can:

Enter the `rs_helpuser` Adaptive Server stored procedure at the RSSD, where *user* is the login name about which you want information:

```
rs_helpuser [user]
```

## Granting permissions in the database

Either Sybase Central or `rs_init` grants the maintenance user permission to access the `rs_lastcommit` system table and the stored procedures that use it.

Neither Sybase Central nor `rs_init` grants permissions to the maintenance user for user tables and stored procedures. You must grant permissions on replicated tables and stored procedures before you can either replicate transactions for replicated tables or replicate executions of the replicated stored procedures.

For each table that is replicated in the database, and for each stored procedure that is executed due to replication, execute the following grant command:

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

---

**Note** Among the permissions granted to the maintenance user is `replication_role`. If you revoke this permission, you will not be able to replicate truncate table unless the maintenance user has been granted `sa_role`, owns the table, or is aliased as the Database Owner.

---

## Granting permissions for a primary database

If a replicate database holds primary data, then it is also a primary database. In a primary database, special permissions are necessary on two replication objects: subscriptions and request functions.

When subscriptions are created, the `rs_marker` stored procedure is executed at the primary database. Any database user who can create subscriptions must have permission to execute `rs_marker`.

A primary database may also receive transactions via request function delivery from clients at replicate sites. These transactions are executed at the primary site as if by the user executing the request function. Any user login name with permission to execute request functions must also have permission to execute `rs_update_lastcommit`, which executes in every DSI transaction.

The permission requirements are the same for request functions and request stored procedures. Refer to Chapter 10, “Managing Replicated Functions” for more information on using request functions.

The following grant commands allow any user in the database to execute `rs_marker` and `rs_update_lastcommit`:

```
grant execute on rs_marker to public
grant execute on rs_update_lastcommit to public
```

These stored procedures should only be executed by Replication Server on behalf of users. Sybase Central or `rs_init` grants these permissions to “public.” You may want to restrict permissions to the database users who are allowed to create subscriptions, execute request functions, or request stored procedures.

## Creating database connections

A connection defines a database to the Replication Server. A Replication Server is designated to manage the database and, if it is a replicate database, to distribute transactions to the database. The database connection provides Replication Server with:

- The name of the data server and database the connection is for
- The error class used to process errors returned from the data server
- The function-string class to use with the database
- The maintenance user login name and password

- Information about whether there is a RepAgent thread for the database connection
- Options for creating active and standby databases for warm standby applications
- Configuration parameters that affect connections

You can create a database connection in these ways:

- To create a standard connection to an Adaptive Server database, use Sybase Central or `rs_init`.
- To create a connection to a non-Sybase database, use the `create connection` command.

## Information for adding a database connection

The Replication Server installation and configuration guides for your platform describe how you use `rs_init` to add databases.

When you add a database, you specify:

- Replication Server name
- Replication Server System Administrator user name and password
- Adaptive Server name
- Adaptive Server System Administrator user name and password
- Database name
- Whether the database requires a RepAgent
- Maintenance user name and password
- Database Owner user name and password
- Whether the physical connection is for an existing logical connection

## Adding databases for logical connections

If you are adding a physical connection for an existing logical connection (which you create with Sybase Central or the `create logical connection` command), you also specify the following information:

- Active or standby connection

- Logical data server name
- Logical database name

In addition, if you are adding a standby connection, you specify the following information in Sybase Central or `rs_init`:

- Active data server name
- Active database name
- Active database System Administrator user name and password
- Whether to initialize standby database using dump and load method
- Whether to use dump marker to start replication

Refer to Chapter 3, “Managing Warm Standby Applications” in the *Replication Server Administration Guide Volume 2* for more information about warm standby operations.

## Adding a database that requires a RepAgent thread

If you are adding an Adaptive Server primary database that requires a RepAgent, you specify the Replication Server user name and password.

## Using the *create connection* command

To add a database for a non-Sybase data server, use the `create connection` command.

To add a database for a Sybase data server, you normally use Sybase Central or `rs_init`, both of which prepare the database for replication. If you use `create connection`, you must prepare the database for replication yourself. Refer to “Steps in preparing databases for replication” on page 164.

Enter `create connection` at the Replication Server that is to manage the database. The syntax is:

```
create connection to data_server.database
  set error class [to] error_class
  set function string class [to] function_class
  set username [to] user
  [set password [to] passwd]
  [set database_param [to] 'value']
  [set security_param [to] {'required' | 'not_required'}]
  [with {log transfer on, dsi_suspended}]
  [as active for logical_ds.logical_db |
```

as standby for *logical\_ds.logical\_db*  
[use dump marker]

You must use the `with dsi_suspended` clause, which starts the connection with the DSI suspended, when you create a connection to a database that will not be a replicate database.

The `as active`, `as standby`, and `use dump marker` clauses are used only when you create physical connections for a logical connection for a warm standby database. Only Adaptive Server databases may be used in warm standby applications.

If your system supports network-based security, use the `set security_param` command according to instructions in “Managing network-based security” on page 210.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about the `create connection` command.

## Altering database connections

To change the attributes of a database connection, use Sybase Central or perform the following steps at the Replication Server where the connection was created:

- 1 Use `suspend connection` to suspend activity on the connection. See “Suspending database connections” on page 171 for details.



- 2 Execute the alter connection command. See “Setting and changing parameters affecting physical connections” on page 172 for detailed instructions.

---

**Note** Using the set log transfer off clause for the alter connection command drops the RepAgent connection from a primary site. Before using this clause, be sure there are no replication definitions defined for data in the database.

---

- 3 Use resume connection to resume activity on the connection. See “Resuming database connections” on page 183 for more information.

---

**Note** When you alter a connection in Sybase Central, you need to suspend and resume the connection before the new value will take effect.

---

## Suspending database connections

If you have sa permission, you can temporarily suspend access to a data server. You must suspend a database connection before you alter it or when you remove a data server from service for maintenance.

While data server access is suspended, the Replication Server queues transactions for the data server so they can be applied when the connection is resumed.

You can temporarily suspend access to a data server using Sybase Central or you can use the following command:

```
suspend connection to data_server.database
[with nowait]
```

By default, suspend connection completes the current transaction before suspending. Use the with nowait clause to suspend the connection in mid-transaction. This may be appropriate if a large transaction is responsible for a failure in a replicate database.

## Setting and changing parameters affecting physical connections

You set configuration parameters for a connection when you create it. Later, you can update those parameters with Sybase Central or the alter connection command.

You can change the configuration of either a single database connection or of all database connections that originate from a single Replication Server. If you are adding many database connections to a Replication Server, you may want to change configuration parameters affecting all connections in order to fine-tune server performance.

To change configuration parameters for *all* connections originating at the current Replication Server, use the configure replication server command. Refer to “Changing parameters affecting all connections” on page 181 for more information.

Configuration parameters that are set for individual connections with alter connection override parameters that are set with configure replication server. Thus, you can set default parameters with configure replication server and then customize settings for specific connections with alter connection.

### Changing parameters affecting a single connection

After a connection is created, you can change its configuration parameters with the alter connection command. Refer to Table 7-1 on page 174 for a list and description of configuration parameters that affect connections.

#### Using *alter connection*

alter connection lets you change the attributes of a database connection. Use this command, for example, if you have added an Adaptive Server database connection using Sybase Central or rs\_init, and then decide that you want the database connection to use a derived function-string class instead of a system-provided class. The syntax for alter connection is:

```
alter connection to data_server.database {
    set function string class [to] function_class |
    set error class [to] error_class |
    set password [to] passwd |
    set log transfer [to] {on | off} |
    set database_param [to] 'value' |
    set security_param to {'required' | 'not_required'} |
    set security_services [to] "default"
}
```

You indicate the data server and database that is connected to the Replication Server and specify one or more of the attributes to change. These include:

*function\_class* – the function-string class to use with the database connection.

*error\_class* – the error class to use for handling database errors.

*passwd* – the new password to use with the login name for the database connection.

log transfer on – allows transactions to be sent, using this connection, to the Replication Server.

log transfer off – stops transactions from being sent, using this connection, from a primary database to the Replication Server.

*database\_param* – updates a configuration parameter that affects connections. See Table 7-1 for a list of parameters you can change.

*security\_param* – updates a network security configuration parameter that affects connections. See “Managing network-based security” on page 210 for a list and description of parameters you can change.

set security\_services [to] 'default' – resets all network-based security features for the connection to “not required.” See “Managing network-based security” on page 210 for a description of network security for Replication Server.

### **An example of using *alter connection***

To change the function-string class for the pubs2 database in the SYDNEY\_DS data server to *sqlserver\_derived\_class*, enter the following commands in the SYDNEY\_RS Replication Server:

```
suspend connection to SYDNEY_DS.pubs2
alter connection to SYDNEY_DS.pubs2
    set function string to class
        sqlserver_derived_class
resume connection to SYDNEY_DS.pubs2
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about the keywords and options of the *alter connection* command.

### **Configuration parameters affecting individual connections**

Table 7-1 displays the configuration parameters that affect database connections. (These configuration parameters affect physical database connections only. For parameters that affect logical database connections, see “Changing parameters affecting logical connections” on page 99 in the *Replication Server Administration Guide Volume 2*.)

If your system supports network-based security, see “Managing network-based security” on page 210 for information about security parameters that affect connections.

See “Using parallel DSI threads” on page 151 in the *Replication Server Administration Guide Volume 2* for information about parameters for setting up and tuning parallel DSI connections.

See “Connection parameters that affect performance” on page 137 in the *Replication Server Administration Guide Volume 2* for a list of configuration parameters that affect performance.

**Table 7-1: Configuration parameters affecting database connections**

Parameter (database_param)	Value (value)
batch	The default, “on,” allows command batches to a replicate database. Default: on
batch_begin	Indicates whether a begin transaction can be sent in the same batch as other commands (such as insert and delete). Default: on
command_retry	The number of times to retry a failed transaction. The value must be greater than or equal to 0. Default: 3
db_packet_size	The maximum size of a network packet. During database communication, the network packet value must be within the range accepted by the database. You may change this value if you have Adaptive Server that has been reconfigured. Maximum: 16384 bytes Default: 512-byte network packet for all Adaptive Server databases
disk_affinity	Specifies an allocation hint for assigning the next partition. Enter the logical name of the partition to which the next segment should be allocated when the current partition is full. Values are “ <i>partition_name</i> ” and “off.” Default: off
dsi_alt_writetext	Controls how large object updates are sent to the replicate database. Values are: <ul style="list-style-type: none"> <li>• dcanyc – generates a writetext command that includes primary key columns. This setting prevents full table scans when populating non-ASE replicate databases using DirectConnect Anywhere™ as an interface.</li> <li>• off (default) – generates an Adaptive Server writetext command that includes a text pointer.</li> </ul>

Parameter (database_param)	Value (value)
dsi_charset_convert	<p>The specification for handling character set conversion. This parameter applies to all data and identifiers to be applied at the DSI in question. The values are:</p> <ul style="list-style-type: none"> <li>• “on” (default) – convert from the primary Replication Server character set to the replicate Replication Server character set; if character sets are incompatible, shut down the DSI with an error.</li> <li>• “allow” – convert where character sets are compatible; apply any unconverted updates to the database, as well.</li> <li>• “off” – do not attempt conversion. This option is useful if you have different but compatible character sets and do not want any conversion to take place. During subscription materialization, a setting of “off” behaves as if it were “allow.”</li> </ul>
dsi_check_lock_wait	<p>The number of milliseconds before the DSI executor thread executes the <code>rs_thread_check_lock</code> function string, which queries the replicate database about lock status.</p> <p>Default: 3000 milliseconds (3 seconds)</p>
dsi_cmd_batch_size	<p>The maximum number of bytes that Replication Server places into a command batch.</p> <p>Default: 8192 bytes</p>
dsi_cmd_separator	<p>The character that separates commands in a command batch. For example, if you have specified a different separator character and want to change it back to the default character, enter:</p> <pre>alter connection to data_server.database set dsi_cmd_separator to '&lt;Return&gt;'</pre> <p>Press the Return key, and no other characters, between the two single-quote characters.</p> <p>Default: newline (\n)</p> <p><b>Note</b> Pressing the Return key is effective only in an interactive update; it is not applicable to executing a script, such as a DDL generated script. You must update this parameter in an interactive mode. You cannot reset it from within a script.</p>
dsi_commit_check_locks_intrvl	<p>The number of milliseconds (ms) the DSI executor thread waits between executions of the <code>rs_dsi_check_thread_lock</code> function string. Used with parallel DSI. See “Using parallel DSI threads” on page 151 in the <i>Replication Server Administration Guide Volume 2</i>.</p> <p>Default: 1000 ms (1 second)</p> <p>Minimum: 0</p> <p>Maximum: 86,400,000 ms (24 hours)</p>

Parameter (database_param)	Value (value)
dsi_commit_check_locks_max	<p>The maximum number of times a DSI executor thread checks whether it is blocking other transactions in the replication database before rolling back its transaction and retrying it. Used with parallel DSI. See “Using parallel DSI threads” on page 151 in the <i>Replication Server Administration Guide Volume 2</i>.</p> <p>Default: 400</p> <p>Minimum: 1</p> <p>Maximum: 1,000,000</p>
dsi_commit_control	<p>Specifies whether commit control processing is handled internally by Replication Server using internal tables (on) or externally using the rs_threads system table (off). Used with parallel DSI. See “Using parallel DSI threads” on page 151 in the <i>Replication Server Administration Guide Volume 2</i>.</p> <p>Default: on</p>
dsi_exec_request_sproc	<p>Turns on or off request stored procedures at the DSI of the primary Replication Server.</p> <p>Default: on</p>
dsi_fadeout_time	<p>The number of seconds of idle time before a DSI connection is closed. A value of -1 specifies that the connection should not fade out.</p> <p>Default: 600 seconds</p>
dsi_ignore_underscore_name	<p>When the transaction partitioning rule is set to name, specifies whether or not Replication Server ignores transaction names that begin with an underscore. Values are “on” and “off.”</p> <p>Default: on</p>
dsi_isolation_level	<p>Specifies the isolation level for transactions. ANSI standard and Adaptive Server supported values are:</p> <ul style="list-style-type: none"> <li>• 0 – ensures that data written by one transaction represents the actual data.</li> <li>• 1 – prevents dirty reads and ensures that data written by one transaction represents the actual data.</li> <li>• 2 – prevents nonrepeatable reads, prevents dirty reads, and ensures that data written by one transaction represents the actual data.</li> <li>• 3 – prevents phantom rows, prevents nonrepeatable reads, prevents dirty reads, and ensures that data written by one transaction represents the actual data.</li> </ul> <p>Through the use of custom function strings, Replication Server can support any isolation level the replicate data servers may use. Support is not limited to ANSI standard only.</p> <p>The default value is the current transaction isolation level for the target data server.</p>

<b>Parameter (database_param)</b>	<b>Value (value)</b>
dsi_keep_triggers	Specifies whether triggers should fire for replicated transactions in the database. “off” – causes Replication Server to set triggers off in the Adaptive Server database, so that triggers do not fire when transactions are executed on the connection. Use this setting for standby databases. “on” – specifies all databases except standby databases. Default: on (except standby databases)
dsi_large_xact_size	The number of commands allowed in a transaction before the transaction is considered to be large. Minimum: 4 Default: 100
dsi_max_cmds_to_log	The number of commands to write into the exceptions log for a transaction. Default: -1 (all commands)
dsi_max_xacts_in_group	Specifies the maximum number of transactions in a group. Larger numbers may improve data latency at the replicate database. Range of values: 1 – 100. Default: 20
dsi_max_text_to_log	The number of bytes to write into the exceptions log for each rs_writetext function in a failed transaction. Change this parameter to prevent transactions with large text, unitext, or image columns from filling the RSSD or its log. Default: -1 (all text, unitext, or image columns)
dsi_num_large_xact_threads	The number of parallel DSI threads to be reserved for use with large transactions. The maximum value is one less than the value of dsi_num_threads. Default: 0
dsi_num_threads	The number of parallel DSI threads to be used. The maximum value is 255. Default: 1
dsi_partitioning_rule	Specifies the partitioning rules (one or more) the DSI uses to partition transactions among available parallel DSI threads. Values are origin, origin_sessid, time, user, name, and none. See also “Partitioning rules: reducing contention and increasing parallelism” on page 162 in the <i>Replication Server Administration Guide Volume 2</i> for detailed information. Default: none

Parameter (database_param)	Value (value)
dsi_replication	<p>Specifies whether or not transactions applied by the DSI are marked in the transaction log as being replicated.</p> <p>When dsi_replication is set to “off,” the DSI executes set replication off in the Adaptive Server database, preventing Adaptive Server from adding replication information to log records for transactions that the DSI executes. Since these transactions are executed by the maintenance user and, therefore, usually not replicated further (except if there is a standby database), setting this parameter to “off” avoids writing unnecessary information into the transaction log.</p> <p>dsi_replication must be set to “on” for the active database in a warm standby application for a replicate database, and for applications that use the replicated consolidated replicate application model.</p> <p>Default: on (“off” for standby database in a warm standby application)</p>
dsi_serialization_method	<p>Specifies the method used to maintain serial consistency between parallel DSI threads when applying transactions to a replicate data server.</p> <ul style="list-style-type: none"> <li>• no_wait – specifies that a transaction can start as soon as it is ready—without regard to the state of other transactions.</li> <li>• wait_for_commit – specifies that a transaction cannot start until the transaction scheduled to commit immediately preceding it is ready to commit.</li> <li>• wait_for_commit – maintains transaction serialization by instructing the DSI to wait until a transaction is ready to commit before initiating the next transaction (off) or wait until a transaction has committed before initiating the next transaction (on).</li> <li>• none – same as wait_for_start. Retained for backward compatibility.</li> <li>• single_transaction_per_origin – same as wait_for_start with dsi_partitioning_rule set to origin. Retained for backward compatibility.</li> </ul> <p>Default: wait_for_commit</p>
dsi_sqt_max_cache_size	<p>Maximum SQT (Stable Queue Transaction) interface cache memory for the database connection, in bytes.</p> <p>The default, 0, means the current setting of the sqt_max_cache_size parameter is used as the maximum cache size for the connection.</p> <p>Default: 0</p>



Parameter (database_param)	Value (value)
dsi_text_convert_multiplier	<p>Changes the length of text or untext datatype columns at the replicate site. Use dsi_text_convert_multiplier when text or untext datatype columns must expand or contract due to character set conversion. Replication Server multiplies the length of primary text or untext data by the value of dsi_text_convert_multiplier to determine the length of text or untext data at the replicate site. The value type is float.</p> <ul style="list-style-type: none"> <li>• If the character set conversion involves expanding text or untext datatype columns, set dsi_text_convert_multiplier equal to or greater than 1.0.</li> <li>• If the character set conversion involves contracting text or untext datatype columns, set dsi_text_convert_multiplier equal to or less than 1.0.</li> </ul> <p>Default: 1</p>
dsi_xact_group_size	<p>The maximum number of bytes, including stable queue overhead, to place into one grouped transaction. A grouped transaction is a set of transactions that the DSI applies as a single transaction. A value of -1 means no grouping.</p> <p>Sybase recommends that you set dsi_xact_group_size to the maximum value and use dsi_max_xacts_in_group to control the number of transactions in a group.</p> <p>Maximum: 2,147,483,647 Default: 65,536 bytes</p>
dump_load	<p>Set to “on” at replicate sites only to enable coordinated dump. See “Loading from coordinated dumps” on page 239 in the <i>Replication Server Administration Guide Volume 2</i> for details.</p> <p>Default: off</p>
exec_cmds_per_timeslice	<p>Specifies the number of LTL commands an LTI or RepAgent Executor thread can process before it must yield the CPU to other threads. The range is 1 to 2, 147, 483, 648.</p> <p>Default: 5</p>
dynamic_sql	<p>Turns dynamic SQL feature on or off. Other dynamic SQL related configuration parameters will only take effect if this parameter is set to “on”.</p> <p>Default: off</p>
dynamic_sql_cache_size	<p>Gives the Replication Server a hint on how many database objects may use the dynamic SQL statement for a connection.</p> <p>Minimum: 1 Maximum: 65536 Default: 100</p>

Parameter (database_param)	Value (value)
dynamic_sql_cache_management	<p>Manages the dynamic SQL cache for a DSI/E thread.</p> <p>Values:</p> <ul style="list-style-type: none"> <li>mru - keep the most recently used statements and deallocate the to allocate new dynamic statements when dynamic_sql_cache_size is reached.</li> <li>fixed (default)- Replication Server stops allocating the new dynamic statements once dynamic_sql_cache_size is reached.</li> </ul>
exec_sqm_write_request_limit	<p>Specifies the amount of memory available to the LTI or RepAgent Executor thread for messages waiting to be written to the inbound queue.</p> <p>Default: 1MB</p> <p>Minimum: 16K</p> <p>Maximum: 2GB</p>
md_sqm_write_request_limit	<p>Specifies the amount of memory available to the Distributor for messages waiting to be written to the outbound queue.</p> <hr/> <p><b>Note</b> In Replication Server 12.1, md_sqm_write_request_limit replaces md_source_memory_pool. md_source_memory_pool is retained for compatibility with older Replication Servers.</p> <hr/> <p>Default: 1MB</p> <p>Minimum: 16K</p> <p>Maximum: 2GB</p>
rep_as_standby	<p>When rep_as_standby is on, table subscriptions replicate tables marked by sp_reptostandby.</p> <p>For rep_as_standby on to succeed, the RepAgent parameters send maint xacts to replicate must be false and send warm standby xacts must be true.</p> <p>Default: off</p>
save_interval	<p>The number of minutes that the Replication Server saves messages after they have been successfully passed to the destination data server.</p> <p>Default: 0 minutes</p>
sub_sqm_write_request_limit	<p>Specifies the memory available to the subscription materialization or dematerialization thread for messages waiting to be written to the outbound queue.</p> <p>Default: 1MB</p> <p>Minimum: 16K</p> <p>Maximum: 2GB</p>

Parameter (database_param)	Value (value)
use_batch_markers	<p>If use_batch_markers is set to on, the function strings rs_batch_start and rs_batch_end will be executed.</p> <hr/> <p><b>Note</b> This parameter must be set to on only for replicate data servers that require additional SQL translation to be sent at the beginning and end of a batch of commands that are not contained in the rs_begin and rs_commit function strings.</p> <hr/> <p>Default: off</p>

## Changing parameters affecting all connections

To set default configuration parameters for all connections originating at the source Replication Server, use the configure replication server command. Refer to Table 7-1 for a list of configuration parameters that affect connections that you can set with configure replication server.

The syntax for configure replication server is:

```
configure replication server
set database_param to 'value'
```

### Example 1

Here is an example of using configure replication server to change the dsi\_fadeout\_time parameter so that the DSI connection does not close. Log in to the source Replication Server and enter:

- 1 Suspend all connections from the source Replication Server. For each connection, enter:

```
suspend connection to data_server.database
```

- 2 Execute configure replication server. Enter:

```
configure replication server
set dsi_fadeout_time to '-1'
```

- 3 Resume suspended connections from the source Replication Server. For each connection, enter:

```
resume connection to data_server.database
```

Configuration changes take effect after you resume the connections.

### Example 2

Here is an example of using configure Replication Server to change the ha\_failover parameter to enable Failover support for all non-RSSD connections from a Replication Server to Adaptive Servers.

- 1 Execute configure replication server. Log in to the Replication Server for which you want to enable Failover support and enter:

```
configure replication server
set ha_failover to 'on'
```

See “Configuring the replication system to support Sybase Failover” in Chapter 7, “Replication System Recovery,” of the *Replication Server Administration Guide Volume 2*

Configuration changes take effect after you resume the connections.

## Changing Replication Server connection parameters to improve performance

Sybase sets default values for configuration parameters for average installations and usage. Depending on your system configuration and how Replication Server is used at your site, you may find that careful altering of certain default values may improve performance. See “Configuration parameters that affect performance” on page 131 in the *Replication Server Administration Guide Volume 2* for a general discussion of performance and configuration parameters. See also “Using parallel DSI threads” on page 151 in the *Replication Server Administration Guide Volume 2*.

If you are adding many new connections, you may want to change the `memory_limit` or `num_threads` Replication Server parameters to improve performance.

Increasing the  
*memory\_limit*

To increase the amount of memory specified for Replication Server, increase the value specified for the `memory_limit` parameter by using `configure replication server` at Replication Server.

For example, execute `configure replication server` in the following manner to increase `memory_limit` to 25MB:

```
configure replication server
set memory_limit to '25'
```

Increasing the  
*num\_threads*

You may need to increase the number of Open Server threads that the Replication Server can use. To do this, increase the value specified for the `num_threads` parameter, using `configure replication server` at the Replication Server.

For example, execute `configure replication server` in the following manner to increase `num_threads` to 70:

```
configure replication server
set num_threads to '70'
```

See configure replication server in Chapter 3 “Replication Server Commands” in the *Replication Server Reference Manual* for more information about the `memory_limit` and `num_threads` parameters.

## Resuming database connections

Once you have changed the attributes of a database connection, you can resume activity on the connection either in Sybase Central or by using the resume connection command.

To resume a database connection from the command line, enter:

```
resume connection to data_server.database
[skip [n] transaction | execute transaction]
```

When the connection is resumed, Replication Server retrieves rows from the `rs_lastcommit` system table so that it can find the correct place in the transaction stream to begin submitting transactions.

The optional `skip [n] transaction` clause instructs Replication Server skip a specified number of transactions in the connection queue before resuming the connection. The first *n* transactions are written to the exceptions log.

The `skip [n] transaction` clause is necessary if the first *n* transactions cause Replication Server to suspend the connection and the cause of the failure cannot be corrected. For example, the transaction may have produced a data server error that is assigned the `retry_stop` or `stop_replication` error action. Or, perhaps it was necessary to use `suspend connection` and the `with nowait` clause to manually interrupt the transaction.

---

**Warning!** If you execute `resume connection` with the `skip transaction` clause, you must correct any inconsistency that results from the lost transaction. Only use the `skip transaction` clause when the condition causing the transaction to fail cannot be corrected.

---

The optional `execute transaction` clause instructs Replication Server to execute the first transaction in the connection’s queue. Use this clause *only* when a system transaction has failed to execute. See “Duplicate detection for system transactions” on page 220 in the *Replication Server Administration Guide Volume 2* for information about error handling for system transactions.

## Changing replicate databases to primary databases

Each primary database must have a Replication Agent that scans the database log and transfers data to the Replication Server for distribution to replicate databases. If you want to change an Adaptive Server database that is designated as replicate-only to be a source of replicated functions or to contain primary data, you must enable the RepAgent thread for the database by following these steps:

At the Replication Server

- 1 Create a RepAgent user so that RepAgent can log in to Replication Server.

Use the following create user command, where *ra\_user\_name* is the name of the RepAgent user and *ra\_password* is the RepAgent's password:

```
create user ra_user_name
set password {ra_password | null}
```

Grant this user connect source permission, using the grant command:

```
grant connect source to ra_user_name
```

If the Replication Server already manages a primary database, you can use the "RepAgent user" that already exists for the new primary database.

- 2 Execute the alter connection command using the log transfer on option:

```
alter connection to data_server.database
set log transfer to 'on'
```

At the Adaptive Server

- 1 If the name of the local Adaptive Server has not yet been defined, you must define it with the following command, where *lname* is Adaptive Server's name:

```
sp_addserver lname, local
```

- 2 If RepAgent threads have not been enabled for the Adaptive Server, you must enable them:

```
sp_configure 'enable rep agent threads'
```

- 3 Configure RepAgent for the database with the `sp_config_rep_agent` system procedure:

```
sp_config_rep_agent dbname, 'enable', 'rs_name',
'rs_user_name', 'rs_password'
```

Refer to Chapter 5, “Setting Up and Managing RepAgent,” for detailed instruction on configuring RepAgent.

---

**Note** The “*rs\_user\_name*” and “*rs\_password*” configured at the Adaptive Server must be the same as the “*ra\_user\_name*” and “*ra\_password*” created at the Replication Server in step 1.

---

- 4 Create the `rs_marker` stored procedure and set its replicate status to “true”, using the `sp_setreplicate` system procedure.

You can find the `rs_marker` stored procedure in the file `rs_install_primary.sql` or `rsinsys.sql` in the `scripts` directory of the Sybase release directory.

See “Creating the `rs_marker` stored procedure” on page 185 for details.

- 5 Start RepAgent:

```
sp_start_rep_agent dbname
```

## Creating the `rs_marker` stored procedure

Replication Server executes the `rs_marker` system function in a primary database during subscription materialization. The function works by executing a replicated stored procedure that is also named `rs_marker`. The procedure checks to make sure it is marked replicated and issues a warning if it is not. Because the `rs_marker` stored procedure is replicated, Adaptive Server records its executions in the transaction log for the database, where they can be read by RepAgent.

Sybase Central and `rs_init` create `rs_marker` when you designate a database as having primary data. It is not required in databases that have no primary data. The exact text of the stored procedure can always be found in `rs_install_primary.sql` or `rsinsys.sql` in the `scripts` directory of the Sybase release directory.

Here is a sample text:

```
create procedure rs_marker
    @rs_api varchar(255)
as
    declare @rep_constant smallint
    select @rep_constant = -32768
    if not exists (select sysstat from sysobjects
        where name = 'rs_marker'
        and type = 'P')
```

```
        and sysstat & @rep_constant != 0)
    begin
        print "Have your DBO execute
        'sp_setreplicate' on the procedure
        'rs_marker'"
    return(1)
end
```

The `rs_marker` stored procedure does not modify data in the database. Its purpose is to execute so that it can be recorded in the transaction log.

If `rs_marker` is not marked as replicated, you can mark it with `sp_setreplicate`:

```
sp_setreplicate rs_marker, 'true'
```

## Changing primary databases to replicate databases

If you want to change a primary database to a replicate database, use the following procedure:

At the current replicate Replication Server

- Drop all subscriptions and publication subscriptions to the replication definitions in this database.

At the current primary Replication Server

- Drop all replication definitions defined for this database.

At the Adaptive Server

- 1 Shut down RepAgent:

```
sp_stop_rep_agent dbname
```

- 2 Disable RepAgent:

```
sp_config_rep_agent dbname, disable
```

At the Replication Server

- Log in to the Replication Server that manages the database and execute alter connection using the log transfer off option:

```
alter connection to data_server.database
set log transfer off
```

At the Adaptive Server

- 1 Set the status of `rs_marker` to “false:”

```
sp_setreplicate rs_marker, 'false'
```

- 2 Set the replicate status of all replicated objects to “false”:

- a Execute `sp_setreptable` without arguments to generate a list of all replicated tables and stored procedures in the database.



- b One by one, set the replicate status of each table and stored procedure to “false,” using `sp_setreptable` and `sp_setrepproc`.

## Dropping database connections

To remove a database from the replication system, use Sybase Central or execute `drop connection`. Before you execute the command, drop any subscriptions for replication definitions for data in the database. If you are dropping a connection to a primary database, first drop all replication definitions for tables in the database.

---

**Note** `drop connection` removes database connection information from the Replication Server system tables. It does not remove replicate data from any database in the system. To remove replicate data, use `drop subscription` using the `with purge` option.

---

To drop a connection, specify the data server with the database whose connection is to be dropped. The syntax is:

```
drop connection to data_server.database
```

For example, to drop the connection to the `pubs2` database in the `SYDNEY_DS` data server, enter:

```
drop connection to SYDNEY_DS.pubs2
```

---

**Note** If you are using RepAgent for log transfer, you should also stop (if necessary) and then disable RepAgent at the primary database. See “Disabling RepAgent” on page 119 for information about disabling RepAgent.

---

For information about dropping logical connections, see “Dropping logical database connections” on page 103 in the *Replication Server Administration Guide Volume 2*.

## Dropping a database from the ID Server

Replication system databases, data servers, and Replication Servers are listed in the `rs_idnames` system table in the RSSD for the ID Server. Occasionally, you may need to remove the entry for a database from this system table.

For example, the drop connection command fails and you want to reuse the connection name. You must force the ID Server to delete from the `rs_idnames` system table the row that corresponds to the database. (Physical database connections have a “P” in the `ltype` column in this system table.)

Log in to the ID Server and execute the `sysadmin dropdb` command to delete the entry for the specified database. The syntax for `sysadmin dropdb` is:

```
sysadmin dropdb, data_server, database
```

You must have `sa` permission to execute any `sysadmin` command.

## Monitoring database connections

This section describes how you can monitor your database connections. Refer to the *Replication Server Troubleshooting Guide* if you need to monitor connections for troubleshooting purposes.

### Viewing current database connections

To check the status of current database connections:

- Use Sybase Central, or
- Use the `admin show_connections` command to display information about all database connections from the Replication Server. This command also displays information about all routes from the Replication Server. Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for information about `admin show_connections` command.

### Listing databases managed by a Replication Server

The `rs_databases` system table contains entries for all of the databases managed by the Replication Server, including databases managed by other Replication Servers that have a route to the Replication Server.

To list the databases that a Replication Server manages:

- Use Sybase Central, or

- Use the `rs_helpdb` stored procedure at the Replication Server RSSD.

The syntax for `rs_helpdb` is:

```
rs_helpdb [data_server, database]
```

Refer to Chapter 6, “Adaptive Server Stored Procedures,” in the *Replication Server Reference Manual* for detailed usage and syntax information of `rs_helpdb` command.

## Displaying DSI thread status

To view DSI thread status, use Sybase Central or the `admin who` commands to display thread status information.

- `admin who` displays all threads in the system, including DSI threads.
- `admin who, dsi` displays the status of the DSI thread, which Replication Server starts to submit transactions to the data server.

Refer to Chapter 4, “Managing a Replication System” for more information about `admin who` commands. Also refer to the *Replication Server Reference Manual*, which provides complete thread status listings.



# Managing Replication Server Security

This chapter describes the RCL commands for managing Replication Server security, including creating and modifying login names, passwords, and permissions; it also describes the dependencies involved in making modifications.

This chapter also describes how to set up and manage third-party, network-based security systems to authenticate users and ensure secure data transmissions.

<b>Topic</b>	<b>Page</b>
Overview	191
Managing Replication Server system security	192
Managing Replication Server user security	199
Managing network-based security	210
Managing SSL security	237

## Overview

Careful management of login names, passwords, and permissions is essential to the security of the replication system. Replication Server login names and specific permissions are required for:

- Each component of the replication system, such as the data server and the Replication Server
- Each user who is setting up replicated data or monitoring and managing the Replication Server

You can set up encrypted passwords throughout the replication system and change passwords that are encrypted. Refer to the Replication Server installation and configuration guides for your platform for details on password encryption. Also see “Enabling and disabling password encryption in sysattributes” on page 201 for a brief overview of encryption capabilities.

In addition, Replication Server supports third-party security services that ensure secure message transmission over the network and enable user authentication for seamless login to Replication Servers in the replication system. See “Managing network-based security” on page 210.

## Managing Replication Server system security

This section provides details on replication system login names and passwords.

Often, one process must log in to another, remote process. In such cases, the login name and password assigned to the process logging in must also exist at the remote process. If a password used to log in to the remote process is changed at the current process only, login attempts fail. This section also describes these dependencies.

You must establish login names and passwords for the various components of the Replication Server system, including RSSDs, RepAgent, the ID Server, and Replication Servers themselves.

As a general rule, if you are specifying or modifying system login names, keep the names unique. If you use the same login name for different roles, then any time you change the password, many of the dependencies described in this section are affected.

Table 8-1 lists all login names required in a replication system.

**Table 8-1: Overview of replication system login names**

Source server	Destination server/database	Login name description
Primary Replication Server	Primary Adaptive Server/RSSD	RSSD primary user
Replicate Replication Server	Replicate Adaptive Server/RSSD	RSSD maintenance user
Replicate Replication Server	Replicate Adaptive Server/replicate database	database maintenance user
RepAgent for RSSD	Replication Server	RepAgent user for RSSD
RepAgent for primary database	Replication Server	RepAgent user for primary database
Replication Servers	ID Server (Replication Server)	ID Server user
Replication Servers	Other Replication Servers	Replication Server user (RSI user)

## RSSD login names and passwords

When you install Replication Server, the `rs_init` program creates the primary and maintenance Adaptive Server login names to maintain the RSSD.

Replication Server uses the **primary user** login name to modify the system tables in the RSSD for the primary Replication Server. Modifications may include route, replication definition, and function-string information changes to be replicated to RSSDs for other Replication Servers. You set up the primary user when you create the primary RSSD using `rs_init`.

Replication Server uses the **maintenance user** login name to apply modifications to replicate RSSDs. RepAgent filters out RSSD modifications made by the maintenance user to avoid replicating them to other RSSDs. You set up the maintenance user when you create the replicate RSSD using `rs_init`.

If the login name or password is changed for either the primary or maintenance user, edit the Replication Server configuration file to match these changes, and restart the Replication Server.

### Changing RSSD primary user login name and password

Observe these guidelines when you change the RSSD primary user login name and password. Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for command syntax details.

- Never change the RSSD primary user login name and/or password while routes are being created.

While a route is being created, the destination Replication Server uses the primary user login name and password to create and materialize subscriptions at the destination site for replicated RSSD system tables.

- Be sure to also apply the same RSSD primary user login name and/or password changes to the Replication Server.
  - To change an encrypted or clear text password, use `alter user` with the `set password` clause.
  - To change both a login name and password (encrypted or clear text), use `drop user` to drop the old login name and `create user` to create the new login name and password. Then grant the user primary subscribe permission.

See “Managing Replication Server permissions” on page 202 for more information.

- Update the Replication Server configuration file with the new login name and/or password. Use `rs_init` if the password is encrypted.
- For the updates to take effect, restart the Replication Server.

## Replication Server login name and password for the RepAgent

RepAgent retrieves information about changes to the replicated system tables in the RSSD or to the primary database from the database transaction logs and submits them to the Replication Server for distribution.

Replication Server needs a login name for RepAgent. The `rs_init` program uses the `create user` command to add this Replication Server user.

Observe these guidelines when you change the Replication Server login name and/or password for the RepAgent. The login name and password you create at the Replication Server must be the same as that used to configure the RepAgent at Adaptive Server.

Refer to Chapter 3, “Replication Server Commands,” and Chapter 5, “Adaptive Server Commands and System Procedures,” in the *Replication Server Reference Manual* for syntax details.

- |                       |  |
|-----------------------|--|
| At Replication Server | <ul style="list-style-type: none"><li>• To change the password, use the <code>alter user</code> command with the <code>set password</code> clause.</li><li>• To change both the login name and password, use the <code>drop user</code> command to drop the old user login name and the <code>create user</code> command to create the new login and password. Then grant the user connect source permission.</li></ul>  |
| At Adaptive Server    | <ul style="list-style-type: none"><li>• To change the login name and password, use the <code>sp_config_rep_agent</code> system procedure with the <code>dbname</code>, <code>rs_servername</code>, <code>rs_username</code>, and <code>rs_password</code> options.<br/><br/>This updates the login name and password in the database <code>sysattributes</code> table. The password is always encrypted.</li><li>• For the updates to take effect, restart RepAgent.</li></ul> |

## ID Server login name and password

The ID Server registers Replication Servers and databases in a replication domain. Replication Servers use the `ID_user` configuration parameter in the Replication Server configuration file to connect to the ID Server. For each Replication Server, the ID Server login name and password must match the ID Server entry.

The ID Server must be the first Replication Server installed. The ID Server login name and password are established using `rs_init`.



If you change the login name and/or password for the ID Server, be sure to modify *ID\_user* in the Replication Server configuration file of each Replication Server that is defined to the ID Server, as well as the Replication Server configuration file for the ID Server itself. You can make password changes using *rs\_init*.

You also must change the ID Server login name and/or password in the Replication Server. See “Managing Replication Server login names and passwords” on page 200 for more information on changing login names and/or passwords.

## Replication Server login name and password for Replication Servers

To send operations, Replication Servers log in to other Replication Servers. The login name is created using *rs\_init*. The login name is used when a direct route is created, from one Replication Server to another.

To change the password for a login name used for a direct route, execute the *alter route* command. See Chapter 6, “Managing Routes” for details.

## Maintenance user Adaptive Server login name and password

Replication Servers log in to Adaptive Server for the RSSD database or a user database using the maintenance user login name. When applying primary changes (insert, delete, or update operations) to replicate databases, Replication Server uses the maintenance user login name and password.

---

**Note** Among the permissions granted to the maintenance user is *replication\_role*. If you revoke maintenance user’s *replication\_role*, Replication Server will not replicate truncate table unless the maintenance user has been granted *sa\_role*, owns the table, or is aliased as the Database Owner.

---

To change the password for the maintenance user, use the *alter connection* command.

## Sending encrypted passwords for Replication Server client connections

Replication Server supports the `-X` option in `isql` that sends encrypted passwords through the network when making a client connection.

To ensure that all Replication Server client connections—except the first connection to the RSSD—send encrypted passwords, set the Replication Server configuration parameter `send_enc_password` to “on.” For example, enter:

```
configure replication server
  set send_enc_password to 'on'
```

To ensure that all Replication Server client connections, including the first connection to the RSSD, send encrypted passwords, set the configuration parameter `RS_send_enc_pw` to “on” in the `rs_name.cfg` file using a text editor.

If `RS_send_enc_pw` is “on,” all Replication Server connections to the RSSD send encrypted passwords—even if `send_enc_password` is “off.”

## Existing Encrypted Password Migration

Newly created passwords uses the Federal Information Processing Standards (FIPS)-certified encryption algorithm.

Use the information in the following table to migrate existing encrypted passwords in the Replication Server configuration file, `rs_users` and `rs_maintusers` tables.

**Table 8-2: Commands to encrypt passwords in new algorithm**

Task	Command/Step
Encrypt existing user passwords to the new algorithm	<pre>alter user user set password password</pre> <p>where:</p> <ul style="list-style-type: none"> <li>• <i>user</i> is the login name of the existing user</li> <li>• <i>password</i> is the existing password you want to encrypt using the new algorithm.</li> </ul>
Encrypt existing database maintenance user passwords to the new algorithm	<pre>alter connection to data_server.database set password to password</pre> <p>where, <i>password</i> is the existing password you want to encrypt using the new algorithm.</p>
Encrypt existing route user passwords to the new algorithm	<pre>alter route to dest_replication_server set password to passwd</pre> <p>where:</p> <ul style="list-style-type: none"> <li>• <i>dest_replication_server</i> is the name of the destination Replication Server</li> <li>• <i>passwd</i> is the existing password you want to encrypt using the new algorithm.</li> </ul>
Encrypt existing user passwords in the configuration file to the new algorithm	<ul style="list-style-type: none"> <li>• Use <code>rs_init</code> to encrypt the passwords using the new algorithm.</li> </ul>

## Extended password encryption support

Replication Server uses Sybase Common Security Infrastructure (CSI) to provide server or client authentication, cryptography for encryption and decryption of passwords that are stored in the RSSD tables, and key-pair generation to support extended password encryption.

Extended password encryption uses asymmetric key encryption, which allows Open Client applications with connection property `CS_SEC_EXTENDED_ENCRYPTION` enabled to connect to the Replication Server. It also allows Replication Server to enable `CS_SEC_EXTENDED_ENCRYPTION` when connecting to other servers.

Asymmetric key encryption uses a public key to encrypt the password and a private key to decrypt the password. The private key is not shared across the network, and is therefore secure.

---

**Note** To use the extended password encryption feature, you must have a server that supports extended password encryption, such as ASE 15.0.2 ESD #2 or later.

---

## Sybase Central dependencies

The Replication Manager logs into the Replication Server and the RSSD using the login names and passwords that you specified when you added the server to the RM.

If you are using the RM, be sure to update the login information. This information can be found in the Replication Server properties dialog.

## Replication Server object creation dependencies

Login name and password dependencies also apply when you create Replication Server objects, specifically subscriptions and replicated functions (applied and request functions) that are executed at primary or replicate Replication Servers. This section addresses these dependency issues.

### Subscriptions

When you create a subscription, the login name that you used to log in to the replicate Replication Server must exist on both the primary Replication Server and the primary Adaptive Server. The login name must have the same password on all three servers.

When you drop a subscription, the replicate Replication Server logs in to the primary Replication Server using the login name and password you used to log in to the replicate Replication Server. Do not change the password of this login name on the primary Replication Server before the drop subscription process is complete.

The RSSD's "primary" user login name that is automatically created on the Replication Server is used as the "subscribing user" when routes are created. The rules for a user creating a subscription apply to the RSSD primary user.

**Suggestions**

- Do not create subscriptions as the sa user.
- The select command, issued at the primary Replication Server when creating the subscription, does not include a table owner name unless an owner name is specified in the replication definition. If no owner name is specified, make sure that either the user owns the table or the table is owned by the “dbo” user.
- Do not change passwords while subscriptions are materializing or dematerializing.

**Replicated functions and stored procedures**

When a primary Replication Server receives a request function or a request stored procedure from a replicate Replication Server, it logs in to the primary data server with the login name and password of the user who initiated the request function or request stored procedure at the replicate site.

Therefore, to execute a request function or request stored procedure at a replicate data server, the user must have the same login name and password at the primary data server, and must have execute permission for the stored procedure at the primary data server.

When a replicate Replication Server receives an applied function or applied stored procedure from a primary site, the replicate Replication Server uses the maintenance user login name and password to execute the stored procedure in the replicate database.

## **Managing Replication Server user security**

Replication Server has its own set of login names, which are separate from data server login names. Users do not need Replication Server login accounts to access data replicated by Replication Server. Replicated data is available to users if they have permissions to access specific databases. The Database Administrator is responsible for creating databases and authorizing access to them.

Replication Server login names are required so that administrative users of the system can execute Replication Server commands. See “Examining users, passwords, and permissions” on page 208 for information on viewing current Replication Server login names. Password encryption for users can be enabled or disabled.

## Managing Replication Server login names and passwords

The replication system administrator, or any other user who has sa permission, manages login names. Table 8-3 summarizes the RCL commands for managing login names.

**Table 8-3: Commands for managing login names**

Command	Task
create user	Create a new login name.
alter user	Change the password for a login name.
drop user	Drop a user login name.

### Creating a Replication Server login name

The create user command adds new user login names to Replication Server. All users have permission to execute all admin commands. See “Permission summary” on page 205 for individual commands.

You must specify a password for the user when the login name is created. If the user has no password, you must set the password to “null,” which specifies an empty string.

The create user command requires sa permission. The syntax for create user is:

```
create user user
set password {passwd | null}
```

A user’s password can be up to 30 characters long and include letters, digits, and symbols. Case is significant. If the password contains spaces, enclose the password in single quotation marks.

Users can change their own passwords using the alter user command, which is described in the section “Changing a Replication Server password” on page 200.

The following example creates a login name for the user “thomk” with the password “vacUUm”:

```
create user thomk
set password vacUUm
```

### Changing a Replication Server password

The replication system administrator can change any user’s password. Also, a user can change his or her own password. The alter user command is used in each case.

The syntax for alter user is:

```
alter user user
set password {new_passwd | null}
[verify password old_passwd]
```

The same rules that you use for specifying the password using create user also apply to alter user.

The verify password clause, which prevents users from altering each other's passwords, is required for users without sa permission.

The following command changes the password for user "louise" from "null" to "polyESter":

```
alter user louise
set password polyESter
verify password null
```

## Dropping a Replication Server login name

The drop user command removes an existing login name from the Replication Server. This command requires sa permission. The syntax for drop user is:

```
drop user user
```

For example, the following command removes the "thomk" login name:

```
drop user thomk
```

## Enabling and disabling password encryption in sysattributes

The password the RepAgent thread uses to log in to Replication Server is always encrypted before it is stored in the *sysattributes* file of the database. However, you can choose whether or not to encrypt other replication system passwords.

The rs\_init program allows you to enable password encryption when you install or upgrade the replication system. This allows you to encrypt passwords throughout sensitive areas of the replication system. Once the system is installed or upgraded, you can use rs\_init at any time to enable encryption.

If you enable password encryption for a Replication Server, new passwords, passwords contained in the Replication Server configuration file, and passwords stored in the RSSD are all encrypted.

For details on enabling password encryption using the rs\_init program, refer to the Replication Server installation and configuration guides for your platform.

## Disabling encryption on new and existing passwords

Use this procedure to disable password encryption:

- 1 Disable encryption on new passwords that are entered for Replication Server, using `configure replication server`. At Replication Server, enter:

```
configure replication server
set password_encryption to '0'
```

- 2 Change existing passwords in the RSSD to clear text.

To do this, change each user's password using the `alter user` command, the `alter connection` command for maintenance users, and the `alter route` command for routes.

- 3 In the Replication Server configuration file, manually reenter, in clear text, passwords that are currently encrypted.
- 4 Restart the Replication Server to pick up the new `password_encryption` configuration parameters.

## Changing encrypted passwords in the configuration files

To change an encrypted password (to another encrypted password) in a Replication Server configuration file, use the `rs_init` program. You cannot change encrypted passwords directly in the Replication Server configuration files.

For details on dependencies involved in changing passwords for specific login names, see “Managing Replication Server system security” on page 192.

## Managing Replication Server permissions

Replication system administrators manage Replication Server permissions with the `grant` and `revoke` commands. Permissions determine which RCL commands users are permitted to execute.

Any user with a Replication Server login name can execute all admin commands and the `check subscription` command. Other commands can be executed only by users who have been granted the required permissions.

Replication Server users can be granted any of four permissions.



**Table 8-4: Replication Server permissions**

Permission	Description
sa	Users with sa permission are Replication System Administrators. They can execute any Replication Server command and may grant and revoke other permissions, including sa, to and from other users.
create object	Users with create object permission can create objects such as replication definitions, subscriptions, and function strings. Users with create object permission automatically have primary subscribe permission.
primary subscribe	Users with primary subscribe permission can execute the commands needed to create subscriptions for primary data stored in databases managed by the Replication Server. Users with primary subscribe permission at the primary site and create object permission at the replicate site can create a subscription for data at the primary site, but cannot create replication definitions or function strings at the primary site.
connect source	The connect source permission is required for: <ul style="list-style-type: none"> <li>• Login names that RepAgents use to log in to Replication Server, allowing RepAgent to execute the subset of RCL commands known as Log Transfer Language (LTL). Refer to the <i>Replication Server Design Guide</i>.</li> <li>• Login names that a source Replication Server uses to connect to a destination Replication Server for the purpose of sending replicated data or replicated functions. You provide this login name using the create route command.</li> </ul>

## Requirements for creating subscriptions

A subscription creator must have accounts on both the primary and replicate Replication Servers, and the accounts must have the same login name and password. The subscription creator enters a command or a series of commands at the replicate Replication Server, which passes the request to the primary Replication Server.

When the optional clauses use dump marker and subscribe to truncate table are used, the login name and password for the replicate Replication Server should be the same for the primary Replication Server, as well as for both the primary and replicate databases.

At the replicate Replication Server (the destination of the subscription data), the subscription creator must have, at minimum, create object permission in order to materialize the subscription.

At the primary Replication Server (the source of the subscription data), the subscription creator must have, at minimum, primary subscribe permission in order to enter at the replicate site all commands involved in creating subscriptions:

- create subscription (for atomic and nonatomic materialization)

- define subscription (for bulk materialization)
- activate subscription (for bulk materialization)
- validate subscription (for bulk materialization)
- drop subscription

The primary subscribe permission, a subset of create object permission, is provided at the primary Replication Server. It lets users at replicate sites create subscriptions to data stored at primary sites. From replicate sites, these users cannot create any other objects at primary sites, only subscriptions.

---

**Note** Users with create object and sa permissions can also create subscriptions from replicate Replication Servers. The minimal permission required at the primary Replication Server for a user at a replicate site to create subscriptions is primary subscribe.

---

A user creating a subscription must have the following Adaptive Server permissions:

- select permission on the table in the primary database
- insert, update, and delete permission on the replicate table
- execute permission on the rs\_marker stored procedure in the primary database

If you are a replication system administrator, restrict primary subscribe and create object permissions at primary sites to users who require them in order to create subscriptions.

It is possible for a user who has primary subscribe or create object permission to begin creating a subscription without having select permission on the table. If this occurs, Replication Server responds in the following manner:

- If the subscription is created with atomic materialization, the select with holdlock operation fails at the primary database during materialization. The subscription retry daemon (dSUB) retries the select with holdlock until the subscription is dropped or until the select permission is granted to the user for the table at the primary database.
- If the subscription is created with nonatomic materialization, the select operation fails at the primary database during materialization. The subscription retry daemon (dSUB) retries the select until the subscription is dropped or the select permission is granted.

- If the subscription is created with bulk materialization, there is no select transaction, so no error messages are logged, and the subscription succeeds.

## Permission summary

Table 8-5 lists the minimum permission required to execute each RCL command. Users with create object permission automatically have primary subscribe permission. Users with sa permission can execute any command.

**Table 8-5: Minimum permissions to execute RCL commands**

To execute	Minimum permission required
abort switch	sa
activate subscription	create object at replicate, primary subscribe at primary
create partition	sa
admin commands	Can be executed by any user
allow connections	sa
alter connection	sa
alter database replication definition	create object
alter function	create object
alter function replication definition	create object
alter applied function replication definition	create object
alter request function replication definition	create object
alter function string	create object
alter function string class	sa
alter logical connection	sa
alter partition	sa
alter queue	sa
alter replication definition	create object
alter route	sa
alter user	sa – Users can change their own passwords by including the verify clause
assign action	sa
check publication	Can be executed by any user
check subscription	Can be executed by any user
configure connection	sa
configure logical connection	sa
configure replication server	sa

<b>To execute</b>	<b>Minimum permission required</b>
configure route	sa
create article	create object
create connection	sa
create database replication definition	create object
create error class	sa
create function	create object
create function replication definition	create object
create applied function replication definition	create object
create request function replication definition	create object
create function string	create object
create function string class	sa
create logical connection	sa
create partition	sa
create publication	create object
create replication definition	create object
create route	sa
create subscription	create object at replicate, primary subscribe at primary
create user	sa
define subscription	create object at replicate, primary subscribe at primary
drop article	create object
drop connection	sa
drop database replication definition	create object
drop error class	sa
drop function	create object
drop function replication definition	create object
drop function string	create object
drop function string class	sa
drop logical connection	sa
drop partition	sa
drop publication	create object
drop replication definition	create object
drop route	sa
drop subscription	create object at replicate, primary subscribe at primary
drop user	sa
grant	sa

To execute	Minimum permission required
ignore loss	sa
move primary	sa
rebuild queues	sa
resume connection	sa
resume distributor	sa
resume log transfer	sa
resume queue	sa
resume route	sa
revoke	sa
set proxy	sa
set autocorrection	create object
set log recovery	sa
shutdown	sa
suspend connection	sa
suspend distributor	sa
suspend log transfer	sa
suspend route	sa
switch active	sa
sysadmin commands	sa
validate subscription	create object at replicate, primary subscribe at primary
wait for create standby	sa
wait for switch	sa

## Granting permissions

The ability to grant and revoke permissions is reserved for the replication system administrator. Any user who has been granted sa permission can play the role of replication system administrator, and can transfer the grant and revoke ability to other users by granting them sa permission.

The syntax for the grant command is:

```
grant
  {sa | create object | primary subscribe |
  connect source}
to user
```

The *user* is the login name of the user to receive the permission. You can grant only one permission at a time.

Permissions are assigned to Replication Server users—not to database users. A Replication Server user who has create object permission can create Replication Server objects that are associated with any database managed by the Replication Server.

In the following example, the replication system administrator grants create object permission to the “thomk” login name:

```
grant create object to thomk
```

## Revoking permissions

To remove permissions previously granted to a user, use Sybase Central or the revoke command.

The syntax for the revoke command is:

```
revoke {sa | create object | primary subscribe |  
connect source}  
from user
```

---

**Note** You cannot revoke the sa permission from, or drop, the sa login name. This ensures that Replication Server is never without a replication system administrator.

---

The four permissions are managed independently. They can be granted and revoked in any order and the result is the same.

The following revoke command prevents user “louise,” who does not have sa permission, from creating replication definitions:

```
revoke create object from louise
```

## Examining users, passwords, and permissions

You can display the login names, passwords, and permissions for Replication Server users and threads by using the rs\_helpuser stored procedure or by querying the rs\_maintusers and rs\_users system tables in the RSSD.

You can also use Sybase Central to view information on Replication Server login names.

## Using the *rs\_helpuser* stored procedure

Use the *rs\_helpuser* stored procedure to display information about user login names known to a Replication Server. The syntax for *rs\_helpuser* is:

```
rs_helpuser [user]
```

With no parameters, *rs\_helpuser* displays information about all user login names known to the current Replication Server. Permissions are displayed for each primary or maintenance user login name.

If you supply a login name parameter, *rs\_helpuser* displays information about that login name only.

## Querying the *rs\_maintusers* system table

The *rs\_maintusers* system table in the RSSD contains the login name and password information for maintenance users.

*rs\_maintusers* includes a column to identify if the password is encrypted or clear text, and a column to hold the encrypted password.

For example, the following query, executed in the RSSD, lists all available information, including login names, for maintenance users:

```
select * from rs_maintusers
```

## Querying the *rs\_users* system table

The *rs\_users* system table in the RSSD contains the login name and password information for Replication Server users.

*rs\_users* also includes a column to identify if the password is encrypted or clear text, and a column to hold the encrypted password.

The *rs\_users* system table also includes a permissions column, which stores the permissions for each login name. The permissions column is a bit-mask of the permissions granted to users.

Table 8-6 lists the mask values for each of the four permissions.

**Table 8-6: Permission bitmask values in the *rs\_users* system table**

Permission	Mask value
sa	0x0001
connect source	0x0002
create object	0x0004
primary subscribe	0x0008

For example, the following query, executed in the RSSD, lists users who have sa permission:

```
select username, uid from rs_users
where permissions & 0x0001 != 0
```

## Managing network-based security

In a client/server environment, it is important to provide secure data pathways so data transmission remains confidential. Replication Server supports third-party, network-based security mechanisms that focus on:

- Authentication and unified login
- Secure message transmission

With network-based security, users are authenticated—the process of verifying that users are who they say they are—by the security system at login. They receive a credential that can be presented to remote servers in lieu of a password. As a result, users have seamless access to the components of the replication system through a single login.

Replication Server version 12 and later supports MIT Kerberos version 5 or later, CyberSafe Kerberos version 5 Security Server, and Transarc DCE version 1.1 Security Server. Depending on which of these security mechanisms you choose, you can select one or more of these features to secure data transmission:

- Unified login – enables the user to log in to components of the replication system with a single credential issued by the security mechanism.
- Confidentiality – enables the sending and receiving of encrypted data.
- Integrity – ensures that data has not been tampered with.
- Replay detection – verifies that data has not been intercepted.
- Origin check – verifies the source of each data packet.
- Out-of-sequence detection – checks that data packets are received in the order sent.



The security mechanism allows Replication Server to establish secure connections with other Replication Servers, with Adaptive Server, and with other data servers that support the Kerberos or DCE security mechanism and certain Replication Server requirements. You choose the method or methods to secure data transmission between them.

## How security services work

Clients use the security mechanism to ensure a secure pathway to a remote server. Replication Server logs in to remote servers (acting as a client) and also accepts incoming logins (acting as a server). How security services work depends on whether Replication Server (or Adaptive Server or other data server) is acting as client or server.

Replication Server, when acting as the client, uses the security mechanism to ensure a secure pathway to a remote Replication Server or Adaptive Server. Once the secure pathway is established, the security mechanism can provide message protection. When Replication Server acts as a server, it accepts or rejects logins based on its default security settings.

## Login authentication

If a client requests authentication services:

- 1 The client validates the login with the security mechanism and receives a credential, which contains relevant security information.
- 2 The client sends the credential to the server and informs the server it wants to establish a secure connection.
- 3 The server authenticates the client's credential with the security mechanism. If the credential is not valid, the secure connection is rejected.
- 4 The server checks message protection properties, if the properties are compatible, the connection is established.

## Message protection

If the current Replication Server (the client) requests data protection services:

- 1 The client uses the security mechanism to prepare the data packet it will send to the server.

For example, if the client requests message confidentiality, the security mechanism encrypts the commands that will be sent to the remote server. If the client requests out-of-sequence checking, the security mechanism time-stamps each data packet.

- 2 The client sends the data to the destination server.
- 3 When the server receives the data, it uses the security mechanism to perform the appropriate decryption or validation.
- 4 The server returns the results to the client, using the security mechanism to perform the security action requested. For example, the server returns results in encrypted form or time-stamps each data packet returned to the client.

## Requirements and restrictions

To enable network-based security you need:

- A network-based security mechanism installed on all machines for which network security is to be enabled. The security mechanism must be supported by Replication Server.

---

**Note** Make sure that you use either the MIT Kerberos, CyberSafe Kerberos or Transarc DCE security mechanism. Sybase network-based security will not run on other Kerberos or DCE security mechanisms.

---

- Replication Server version 11.5 or later for all client and destination Replication Servers.
- Adaptive Server version 12 or later and/or compatible heterogeneous data servers for all client and destination data servers.

Compatible heterogeneous data servers must support the security mechanism installed on Replication Server and the set proxy concept. See the *Replication Server Reference Manual* for a description of this Adaptive Server command.

These restrictions apply:

- Both ends of a secured pathway (client and server) must support the same security mechanism, and the security parameters must have the same feature settings. See “Maintaining network security” on page 232 for more information about security settings.

- User names must be unique throughout the replication system.

If your replication system supports multiple security systems, and you cannot guarantee unique user names, you may need to turn off request stored procedures to avoid a potential security breach. See “Potential security issue” on page 236 for details.

## Setting up network-based security

To set up network security, perform these steps:

- Modify configuration parameters and environment variables, as necessary.
- Identify the Replication Server principal user.
- Activate the security mechanism.
- Configure security services for connections, routes, and other Replication Server pathways.

Each of these tasks is described in the following sections.

## Modifying configuration parameters and environment variables

Configuration files are created during installation at default locations in the Sybase directory structure. The configuration files you may need to configure for network security are:

- *libtcl.cfg*
- *objectid.dat*
- The *interfaces* file

If you are using Kerberos security services, you may also need to modify the CSFC5KTNAME environment variable.

## Configuring *libtcl.cfg*

Drivers are libraries that provide an interface to an external service provider. The *libtcl.cfg* file provides a template into which you enter all the configuration information about the security drivers installed on a machine. It is located in the *SYBASE/SYBASE\_REP/config* directory (UNIX), or the *%SYBASE%\ini* directory (Windows 2000 or 2003).

This section provides the information you need to configure the security driver. Refer to the *Open Client and Open Server Configuration Guide* for more information about Sybase drivers.

The syntax for a security driver entry is:

```
provider=driver init-string
```

where

- *provider* is the local name for the security mechanism, for example, “dce.” The mapping of the local name to a global object identifier is defined in *objectid.dat*.
  - The default local name for the DCE security mechanism is “dce.”
  - The default local name for the Kerberos security mechanism is “csfkrb5.”

If you use a local mechanism name other than the default, you must change the local name in the *objectid.dat* file.

- *driver* is the name of the security driver, for example, *libsdce.so*.
- *init-string* is the initialization string for the driver.
  - For the DCE driver, use the following syntax for *init-string*, where *cell\_name* is the name of your DCE cell:

```
secbase=././cell_name
```
  - For the Kerberos driver, use the following syntax for *init-string*, where *domaine\_name* is the name of your Kerberos domaine:

```
secbase=@domaine_name
```

Use a text editor to customize *libtcl.cfg* for your site. Make sure that lines you do not want are preceded with the “;” character. Change one parameter at a time and reboot Replication Server to effect the changes you make.

- An example of an entry for a DCE driver is:

```
[SECURITY]  
dce=libsdce.so secbase=././cell_name
```

- An example of an entry for a Kerberos driver is:

```
[SECURITY]
csfkr5=libsybkrb.so
secbase=@ASElibgss=/krb5/lib/libgss.so
```

## Configuring *objectid.dat*

The *objectid.dat* file maps global object identifiers (OIDs) to local names. It is located in the *\$SYBASE/config* directory (UNIX), or the *%SYBASE%/ini* directory (Windows 2000, 2003). You need to edit this file only if you have changed the local name of a security service in the *libtcl.cfg* file.

- A sample entry in the *objectid.dat* file for DCE is:

```
[secmech]
1.3.6.1.4.1.897.4.6.1 = dce, dcesecmech
```

- A sample entry in the *objectid.dat* file for Kerberos is:

```
[secmech]
1.3.6.1.4.1.897.4.6.6 = csfkrb5, kerberos
```

## Configuring the *interfaces* file

The *interfaces* file contains network and security information for servers. It is located in *\$SYBASE/SYBASE\_REP/interfaces* (UNIX), or *%SYBASE%/ini/sql.ini* (Windows 2000 or 2003). If you use network security, you must include a *secmech* line that gives the global identifiers of supported security services. Supported security mechanisms are listed by their OIDs. Multiple security mechanisms are separated by commas.

The following is a sample entry for the *interfaces* file for either DCE or Kerberos where *server\_principal\_user\_name* is the name of the Replication Server principal user:

```
#
server_principal_user_name
    query tcp ether plum 1050
    master tcp ether plum 1050
    secmech 1.3.6.1.4.1.897.4.6.1
```

See “Identifying the principal user” on page 217 for more information.

## Setting environment variables (Kerberos)

When Replication Server serves as a client, set the `KRB5CCANME` to indicate the location of the credential cache. The credential is a combination of a ticket-granting ticket and the ticket's session key. Replication Server uses this credential to identify itself when login to remote server.

If you are using the Kerberos network security, you may need to reset the `shared-library` path and `CSFC5KTNAME` environment variables.

- Make sure that the shared-library file is in a directory specified in the shared library path so that the client can find the shared-library file at runtime. Shared-library files are:
  - `libgss.so` on Sun Solaris
  - `libgss.sl` on HP-UX
- If the server key table file is in a location other than the Kerberos system default, set the `CSFC5KTNAME` environment variable to the fully qualified pathname of the key table file.
- Make sure that the `LD-LIBRARY_PATH` environment variable includes the path to the CyberSafe `lib` directory as well as the `lib` directories for Adaptive Server, Open Client/Server, and Replication Server.
- Similarly, make sure that the `PATH` environment variable includes the path to the CyberSafe `bin` directory as well as the `bin` directories for Adaptive Server, Open Client/Server, and Replication Server.

## Establishing the principal user

When network security is not enabled, Replication Server logs in to remote servers as one of several possible users, depending on the task to be performed. When network-based security is enabled with unified login, Replication Server must log in to remote servers as the principal user. The principal user credential is the only credential Replication Server has to log in to other processes when network security is active.

When Replication Server logs in to another Replication Server or a data server, the principal user name contained in the credential is mapped to the server name space and a secure connection is established.

---

**Note** Make sure that principal user names are unique. Replication Server cannot log in to another server of the same name.

---

Replication Server executes the `set proxy` command in the remote server (as the principal user) and switches to the appropriate user for the current task.

## Identifying the principal user

It is the responsibility of the replication system administrator to establish a principal user for each Replication Server. Sybase recommends that you use the name of the Replication Server as the principal user name. When you log in to or start Replication Server, you can specify the principal user name with the `-S` flag.

If you do not specify a principal user name using the `-S` flag, Replication Server uses the Replication Server name.

## Identifying the principal user to the security mechanism

The security administrator for the security mechanism must define the Replication Server principal name to the security mechanism.

For DCE:

- Use the DCE `dcecp` tool's `user create` command to create the principal user.

When you are defining a server to DCE, use options that specify that the new principal user can act as a server.

- Use the `keytab create` command of the `dcecp` utility to create a DCE key table file, which contains a principal user's password in encrypted form.

---

**Note** DCE is not supported on UNIX

---

For CyberSafe Kerberos:

- Use the Kerberos `csfadml` tool to create the principal user.
- Use `csfadml` to extract the key table file.

For MIT Kerberos:

- Use administrative command `addprinc` to create principal user.
- Use administrative command `ktadd` to extract key table file.

Refer to documentation from the security mechanism provider for detailed information about identifying servers and users to the security mechanism.

## Identifying principal users to Replication Server

The principal user for other processes—including RepAgents, data servers, and other Replication Servers—using system security and unified login to connect to Replication Server must be identified in the `rs_users` table for the current Replication Server. You can use the `create user` command to add principal user names to `rs_users`.

## Identifying the Replication Server principal user to the replication system

You must add the Replication Server principal user name to destination processes—Replication Servers and data servers—including the ID Server and the RSSD to which Replication Server is connecting using unified login.

Refer to the *Adaptive Server Enterprise System Administration Guide* for information about adding login names to Adaptive Server.

## Activating network-based security

Before configuring security services, you must turn on network-based security for the Replication Server using the `configure replication server` command.

To activate network security, follow these steps:

- 1 Log in to Replication Server and enter:

```
configure replication server
    set use_security_services to 'on'
```

- 2 Shut down Replication Server.
- 3 Restart Replication Server by executing the `repserver` command or the Replication Server run file.
  - If you are using the DCE security mechanism, make sure you include the `-K` flag to specify the key table file location.



- If you are using the Kerberos security mechanism, the key table location must be specified by the CSFC5KTNAME environment variable (UNIX) or the key table registry key entry (Windows 2000 or 2003).

Refer to the *Replication Server Reference Manual* for syntax and other information about the `repserver` command.

To turn off security services, see “Disabling network-based security” on page 233.

## Starting server and clients

For the network security environment to work properly, *both servers and clients should be started only after they have a valid credential.*

For Kerberos systems:

- On UNIX systems, servers and clients should be started after a kinit
- On Windows NT systems, server and clients can be started automatically using the single sign-on feature or manually using the Kerberos credentials manager.

Refer to your Kerberos documentation for more information.

Transarc DCE systems behave in similar manner, refer to your Transarc documentation for information about setting up the proper environment.

## Configuring security services for Replication Server

Replication Server provides parameters for configuring network-based security. Configuration parameters enable:

- Unified login
- Mutual authentication
- Choice of supported security mechanism
- Message confidentiality through encryption

- Other secure message transmission features: message integrity, origin check, replay detection, and out-of-sequence detection

---

**Note** Depending on the security mechanism you choose, one or more of these security features may not be available at your site.

---

You set default parameters in the `rs_init` program during system configuration. Refer to the *Replication Server Configuration Guide* for your platform for information about `rs_init`. This section describes how you set these parameters at the command line.

## Identifying Replication Server pathways

Replication Server coordinates data replication activities for local data servers and exchanges data with Replication Servers and data servers at other sites. Each of these pathways can be configured for network-based security.

- When Replication Server is acting as a client, you can configure security for:
  - All pathways established when Replication Server logs in to another server. These are default global settings.
  - The connection to the RSSD.
  - Individual connections.
  - Individual routes.
  - Replication Server to ID Server pathway.
  - Pathways used to create a route, create a subscription, or drop a subscription.
- When Replication Server is acting as a server, you can configure security for:
  - All incoming logins. These are default global settings.
  - User connection to Replication Server (set when logging on).

**Table 8-7: Network pathways**

Pathway	How to secure it	Special parameters and exceptions
All pathways initiated by the current Replication Server (acting as a client)	Set global security parameters using configure replication server. This is the default setting for all outgoing logins unless overridden for individual pathways.	Use use_security_services to turn off all network security with a single command. See “Disabling network-based security” on page 233.
Connection to the RSSD	Use a text editor to configure the <i>rs_config</i> file.	Security parameters have an “RSSD_” prefix. For example: RSSD_unified_login.
Individual connections	Set security parameters for a connection to a remote database with: <ul style="list-style-type: none"> <li>• create connection, or</li> <li>• alter connection</li> </ul> See the <i>Replication Server Reference Manual</i> for more information about these commands.	Use dsi_exec_request_sproc to suspend request stored procedures. See “Configuring security for database connections” on page 226.
Individual routes defined using the create route command	Set security parameters using: <ul style="list-style-type: none"> <li>• create route, or</li> <li>• alter route</li> </ul> See the <i>Replication Server Reference Manual</i> for more information about these commands.	
Replication Server to ID Server	Set security parameters with configure replication server. See the <i>Replication Server Reference Manual</i> for more information about this command.	Security parameters have an “id-” prefix. For example: id_msg_confidentiality.
Replication Server to primary Replication Server and primary database to: <ul style="list-style-type: none"> <li>• create a route</li> <li>• create or drop a subscription</li> </ul>	Replication Server duplicates the security settings used when the user creating the route or creating or dropping the subscription logs in to Replication Server. See “Borrowing security settings to secure other pathways” on page 232 for more information.	

Pathway	How to secure it	Special parameters and exceptions
All incoming logins (Replication Server acting as server)	Set parameters for incoming logins with configure replication server. Default parameters for outgoing and incoming parameters are set at the same time and are identical.	
Pathway established when user logs in to Replication Server.	Set security parameters with the isql utilities.	Security parameters set for this pathway must be compatible with those set at the Replication Server for all incoming logins. Security for this pathway cannot be configured using the rs_init utility.

## Configuration parameters

Table 8-8 describes the configuration parameters generally available for all pathways. Exceptions and special cases are listed in Table 8-7 and described in detail in each pathway section.

**Table 8-8: Security parameters affecting Replication Server**

configuration_parameter	Description
msg_confidentiality	Indicates whether Replication Server sends and receives encrypted data. If set to “required,” outgoing and incoming data must be encrypted. If set to “not_required,” Replication Server accepts incoming data that is encrypted or not encrypted. Values are “required” or “not_required.” Default: not_required
msg_integrity	Indicates whether data is checked for tampering. Values are “required” or “not_required.” Default: not_required
msg_origin_check	Indicates whether the source of data must be verified. Values are “required” or “not_required.” Default: not_required
msg_replay_detection	Indicates whether data should be checked to make sure it has not been intercepted and re-sent. Values are “required” or “not_required.” Default: not_required
msg_sequence_check	Indicates whether data packages should be checked to ensure that they have been received in the order sent. Values are “required” or “not_required.” Default: not_required
mutual_auth	Requires remote server to provide proof of identify before a connection can be established. Values are “required” or “not_required.” Default: not_required

configuration_parameter	Description
security_mechanism	Specifies the name of the network-based security mechanism. Default: First security mechanism listed in <i>libtcl.cfg</i> .
unified_login	Indicates how Replication Server seeks outgoing connections and accepts incoming connections. The values are: <ul style="list-style-type: none"> <li>“required” – always seeks to log in to remote server with a credential; only accepts incoming logins with a credential.</li> <li>“not_required” – always seeks to log in to remote server with a password; accepts incoming logins with a credential or a password.</li> </ul> <p><b>Note</b> unified_login must be “required” before other security parameters can take effect.</p> <p>Default: not_required</p>

## Planning for compatible settings

Replication Server accepts incoming logins and initiates logins to other servers. Security parameters for all incoming logins (when Replication Server is acting as a server) are set with the `configure replication server` command. Security parameters for outgoing logins (when Replication Server is acting as a client) are set as described in Table 8-7.

When you set up network-based security, you must plan for the interaction between security settings at each end of the secured pathway. Security settings at both ends of each pathway must be compatible.

**Note** It is the replication system administrator’s responsibility to choose and set security features for each server. Replication Server does not query the security features of remote servers before attempting to establish a pathway. An attempted login fails if security features at both ends of the pathway are not compatible.

Use `configure replication server` with the `use_security_services` parameter to activate or deactivate all security services. Table 8-9 describes compatible security settings for client/server interaction. If the security services parameters are not compatible, for example, if a parameter is set to “not\_required” at the client and “required” at the server, the server does not allow the client to log in.

**Table 8-9: Compatible client/server settings**

Client	Server
use_security_services “off”: no security services	Compatible settings: <ul style="list-style-type: none"> <li>• use_security_services “off”, or</li> <li>• use_security_services “on” and security feature “not required”</li> </ul>
use_security_services “on” and security feature “not required”	Compatible settings: <ul style="list-style-type: none"> <li>• use_security_services “on” and security feature “not required,” or</li> <li>• use_security_services “off”</li> </ul>
use_security_services “on” and security feature “required”	Compatible settings: <ul style="list-style-type: none"> <li>• use_security_services “on” and security feature “required”</li> </ul>

## Configuring default values

Use configure replication server to establish default security settings for all outgoing logins (when Replication Server acts as a client) and incoming logins (when Replication Server acts as a server).

You can override default security settings for these outgoing pathways:

- Individual connections – see “Configuring security for database connections” on page 226.
- Individual routes – see “Configuring security for routes” on page 227.
- The pathway from Replication Server to ID Server – see “Configuring security to the ID Server” on page 229.

---

**Note** You cannot override any default security settings that control security for incoming logins.

---

When Replication Server seeks to open a pathway to another server, it checks to see if security parameters have been set specifically for that pathway. If not, Replication Server uses the default security settings determined using configure replication server.

To set global security parameters, log in to Replication Server and execute configure replication server at the isql prompt. Here is the syntax:

```
configure replication server {
    set security_mechanism to 'mechanism_name' |
    set security_parameter to { 'required' |
    'not_required' }}
```

You can set all of the configuration parameters listed in Table 8-8 on page 222. They are stored in the `rs_config` table in the RSSD. You must have `sa` permission to execute them.

## Examples

This section provides examples of using configure replication server.

### Requiring unified login

To require all servers and users that connect to Replication Server to be authenticated by the security mechanism, set `unified_login` to “required.” Log in to Replication Server and execute this command at the `isql` prompt:

```
configure replication server
  set unified_login to 'required'
```

If `unified_login` is “not\_required”, Replication Server allows servers and users to connect with either a credential or a password.

---

**Note** `unified_login` must be “required” for other security services to take effect.

---

### Requiring data encryption

To require all data sent or received by Replication Server to be encrypted, log in to the Replication Server and execute this command at the `isql` prompt:

```
configure replication server
  set msg_encryption to 'required'
```

## Configuring security for the connection to the RSSD

At startup, Replication Server contacts the RSSD for configuration information. You can secure this pathway using network-based security.

When you set up Replication Server, `rs_init` creates the RSSD connection and places default security information in the Replication Server configuration file, `Rep_Server_name.cfg`. By default, `rs_init` sets all network security parameters to “not required.” If you want to secure the pathway, you must use a text editor to change desired default values to “required.”

Configuration values for the RSSD are preceded by an “RSSD\_” prefix. For example:

- `RSSD_mutual_auth`
- `RSSD_msg_origin_check`

See Table 8-8 for a list and description of the available parameters.

## Configuring security for database connections

To configure security for individual connections, use `create connection` or `alter connection`. Security parameters configured with these commands affect security for the outgoing connection to the data server. They override parameters set with `configure replication server`.

### Creating a secure connection

You can set security parameters when you create a connection with `create connection`. Normally, you use this command to add connections to non-Sybase databases.

Here is the syntax for including security features with the `create connection` command. See `create connection` in the *Replication Server Reference Manual* for detailed information about using `create connection`.

```
create connection to data_server.database...
  set username [to] user
  [set password [to] passwd]
  [set security_mechanism [to] 'mechanism_name' |
  set dsi_exec_request_sproc [to] { 'on' | 'off' } |
  set security_mechanism [to] 'mechanism_name' |
  set security_parameter [to] { 'required' |
  'not_required' } ]
```

Table 8-8 on page 222 describes the security parameters you can set with `create connection`. In addition, you can set the `dsi_exec_request_sproc` parameter described in Table 8-10 on page 226.

Connections parameters are stored in the `rs_config` table in the RSSD, and you must have `sa` permission to execute them.

**Table 8-10: Special security parameters for connections**

security_parameter	Description
<code>dsi_exec_request_sproc</code>	Indicates whether request stored procedures at the primary Replication Server are “off” or “on.” Use in multiple security-system environments. Refer to “Using more than one security mechanism” on page 236 for more information. Default: off

Security parameters set at both ends of a connection must be compatible. See “Planning for compatible settings” on page 223 for details.

### Modifying security for a connection

To change the security settings for a database connection, use `alter connection`.



Here is the syntax for altering security:

```
alter connection to data_server.database {  
  ...  
  set password to passwd |  
  set security_mechanism to 'mechanism_name' |  
  set dsi_exec_request_sproc to { 'on' | 'off' } |  
  set security_parameter to { 'required' |  
    'not_required' }}
```

Refer to Table 8-8 on page 222 and Table 8-10 on page 226 for a list and description of parameters you can alter.

To change the security parameters of a database connection, perform these steps at the Replication Server:

- Execute `suspend connection` to suspend activity on the connection.
- Execute `alter connection` to change security parameters. Set one parameter at a time.
- Execute `resume connection` to resume activity on the connection.

## Examples

This section provides some examples of using `alter connection`.

To require Replication Server to connect to the target database (TOKYO\_DS.pubs2) with a credential, execute:

```
alter connection to TOKYO_DS.pubs2  
  set unified_login to 'required'
```

---

**Note** `unified_login` must be “required” for other security services to take effect.

---

To turn “off” request stored procedures at the TOKYO data server in a multiple security-system environment, execute:

```
alter connection to TOKYO_DS.pubs2  
  set dsi_exec_request_sproc to 'off'
```

## Configuring security for routes

You can configure security for individual routes using `create route` or `alter route`. Security parameters configured with these commands affect security for the outgoing login to the destination Replication Server. They override default parameters set with `configure replication server`.

## Creating a secure route

You can set security parameters when you create a route. Here is the syntax for including security features using the create route command.

```
create route to dest_replication_server {
...
[set username to 'user' ]
[set password to 'passwd' ]
[set security_mechanism to 'mechanism_name' |
set security_parameter to { 'required' |
'not_required' } ]
```

Table 8-8 on page 222 describes the security parameters you can set with create route. They are stored in the *rs\_config* table in the RSSD. You must have sa permission to execute them.

Security parameters set at both ends of a route must be compatible. See “Planning for compatible settings” on page 223 for details.

## Modifying security for a route

To change the security settings for a route, use the alter route command.

Log in to Replication Server and execute alter route at the isql prompt. Here is the syntax for altering security:

```
alter route to dest_replication_server {
...
set password to 'passwd' |
set security_mechanism to 'mechanism_name' |
set security_parameter to { 'required' |
'not_required' }}
```

Table 8-8 on page 222 describes the security parameters you can change with alter route.

To change the security parameters of a route, you must first suspend the route. Perform these steps at the Replication Server:

- 1 Execute suspend route to suspend activity on the route.
- 2 Execute alter route to change a security parameter. Change one parameter at a time.
- 3 Execute resume route to resume activity on the route.

## Examples

This section provides some examples of using alter route.

- To require Replication Server to connect to the target Replication Server (TOKYO\_RS) with a password, execute these commands:

```
alter route to TOKYO_RS
    set username 'TOKYO_rsi_user'
alter route to TOKYO_RS
    set password 'TOKYO_rsi_pw'
alter route to TOKYO_RS
    set unified_login to 'not_required'
```

---

**Note** If unified\_login is “not\_required,” you must specify an RSI user and password.

---

- To specify that all messages exchanged with the target Replication Server (TOKYO\_RS) are checked for tampering, execute:

```
alter route to TOKYO_RS
    set msg_integrity to 'required'
```

## Configuring security to the ID Server

To configure network-based security for the network connection from Replication Server to ID Server, use `configure replication server`. The syntax is:

```
configure replication server
    set id_security_param to { 'required' |
    'not_required' }
```

Refer to the *Replication Server Reference Manual* for complete syntax and usage information about `configure replication server`. Table 8-8 on page 222 describes the security parameters you can set for the pathway to the ID Server. They are stored in the `rs_config` table in the RSSD. You must have `sa` permission to configure them. To distinguish settings for this pathway, all ID Server parameters begin with the “id\_” prefix. For example:

- `id_msg_confidentiality`
- `id_security_mechanism`

ID Server security parameters configured with `configure replication server` are dynamic. They take effect immediately and do not require that you restart Replication Server.

## Examples

- To require that the source of all messages be verified, log in to the source Replication Server and enter:

```
configure replication server
  set id_msg_origin_check 'required'
```

- To require that Replication Server logs in to ID Server with a credential, enter:

```
configure replication server
  set id_unified_login to 'required'
```

## Logging in to Replication Server

Connect to Replication Server using a client application such as `isql` or a custom application program you create with Open Client Client-Library. The `isql` utility includes command line options that enable network-based security services for the connection to Replication Server.

Table 8-11 describes the command line options that you can use with `isql` to enable network-based security on the connection.

**Table 8-11: `isql` command line options for security**

Option name	Meaning
<code>-K keytab_file</code>	Use only with DCE security. It specifies a DCE keytab file that contains the security key for the user logging into the server. Keytab files can be created with the DCE <code>dcecp</code> utility—see your DCE documentation for more information. Replication Server must have read permission on this file.  <b>Note</b> For Kerberos users: Specify the location of the key table file using the key table registry key entry (Windows 2000 or 2003).
<code>-S server_name</code>	Specifies the server's network name. If unified login is enabled, this option also specifies the principal user.
<code>-V security_options</code>	Specifies unified login. With this option, the user must log in to the network's security system before running the <code>isql</code> utility. If a user specifies the <code>-U</code> option, the user must supply the network user name known to the security mechanism; any password supplied with the <code>-P</code> option is ignored.  <code>-V</code> can be followed by a string of options that enable additional security services. Here is a list of options and the services they enable. <ul style="list-style-type: none"> <li>• <code>c</code> – data confidentiality</li> <li>• <code>i</code> – data integrity</li> <li>• <code>m</code> – mutual authentication</li> <li>• <code>o</code> – data origin stamping service</li> <li>• <code>r</code> – data replay detection</li> <li>• <code>q</code> – out-of-sequence detection</li> </ul>
<code>-X</code>	Specifies that connections are made with encrypted passwords.

Option name	Meaning
<code>-Z security_mechanism</code>	Specifies the name of a security mechanism to use on the connection to Replication Server.  Supported security mechanism names are listed in the <i>libtcl.cfg</i> file. If no security mechanism is supplied, the default is used, which is the first security mechanism listed under SECURITY in <i>libtcl.cfg</i> .

### Examples of connecting to Replication Server

You can connect to Replication Server by logging in to the security mechanism and then logging in to Replication Server, or you can log directly in to Replication Server.

You must include the `-S` flag to identify the principal user. Some sample logins follow.

Connecting to Replication Server from the security mechanism

To log in first to the DCE security mechanism and then to Replication Server, you can follow these steps:

1 Log in to the DCE security mechanism and receive a credential:

- For DCE, enter

```
dce_login user_name password
```

- For Kerberos, enter

```
kinit user_name password
```

2 Log in to Replication Server with `isql`:

- For DCE, enter

```
isql -Srs_server_name -Vsecurity_option
```

- For Kerberos, enter

```
isql -Srs_server_name -Vsecurity_option
```

---

**Note** When using DCE, if you want to log in as another user, you must include the `-U` and `-K` options.

---

Connecting to Replication Server from outside security

To connect to Replication Server from outside the security mechanism, you can enter:

- For DCE, enter

```
isql -Srs_server_name -User_name  
-Kkeytab_file
```

- For Kerberos, enter

```
isql -Srs-server_name -Yuser_name
```

## Borrowing security settings to secure other pathways

The security services you use when logging in to Replication Server from the command line not only secure the pathway between the client and the server, they may also be duplicated later on in the session when Replication Server opens other pathways.

Replication Server logs in to the primary Replication Server and the primary database over informal pathways when executing these commands:

- create subscription
- drop subscription
- create route

To secure these pathways, Replication Server borrows the security settings entered by the user executing create subscription, drop subscription, or create route when that user logged in to Replication Server.

## Maintaining network security

This section describes the tasks you perform to manage and maintain network security.

### Using *set proxy* to switch logins

When `unified_login` is enabled, Replication Server always logs in to remote processes as the principal user. Nonetheless, Replication Server commands must be executed on the target data server by the correct user for a particular operation. For example, Replication Server must use the maintenance user login name when applying changes to replicate databases. Replication Server uses the Adaptive Server `set proxy` command to switch automatically from the login user to the required user.

You can customize the `set proxy` command with the `rs_setproxy` function string. `rs_setproxy` changes the login name in a data server. `rs_setproxy` has function-string-class scope.

Refer to the *Replication Server Reference Manual* for more information about `rs_setproxy`.

## Disabling network-based security

You can disable all security services with the `configure replication server` command and the `use_security_services` to 'off' parameter.

When you disable network security, Replication Server does not accept incoming logins with security credentials and does not attempt to log in to other processes with a security credential. No security services are active.

Here is the procedure for disabling security:

- 1 Log in to the Replication Server, and at the `isql` prompt enter this command:

```
configure replication server
set use_security_services to 'off'
```

- 2 Restart Replication Server by executing the `repserver` command or the Replication Server run file. Refer to “Replication Server executable program” on page 86 for information about the `repserver` command.

To enable network-based security at the Replication Server, refer to “Setting up network-based security” on page 213.

## Changing the security mechanism

To change to another security mechanism, log in to Replication Server and execute this command at the `isql` prompt:

```
configure replication server
set security_mechanism to 'mechanism_name'
```

The security mechanism you change to must be installed and listed in the SECURITY section of the `libtcl.cfg` file.

## Resetting per-target values to default values

You can change all of your per-target security values for connections or routes using the `set security_services` to 'default' option.

### For connections

Use `alter connection` to change per-target security settings to global values set with `configure replication server`.

Follow this procedure:

- 1 Log in to the Replication Server and execute the suspend connection command at the isql command. Here is the syntax:

```
suspend connection to dataserver.database
```

- 2 Change security settings with the alter connection command. Here is the syntax:

```
alter connection to dataserver.database  
set security_security services to 'default'
```

- 3 Resume the route or connection with the resume connection command for the changes to take effect. Here is the syntax:

```
resume connection to dataserver.database
```

Changes take effect after you resume the connection. This procedure does not affect the configuration of use\_security\_services.

## For routes

Use the set security\_services to 'default' parameter with alter route to change per-target security settings to global values set with configure replication server.

Follow this procedure:

- Suspend the route. Enter:

```
suspend route to dest_replication_server
```

- Alter the route. Enter:

```
alter route to dest_replication_server  
set security_services to 'default'
```

- Resume the route. Enter:

```
resume route to dest_replication_server
```

Changes take effect after you resume the route. This procedure does not affect the configuration of use\_security\_services.

## Viewing information about security services

You can display information about the Replication Server network-based security using the admin security\_property and admin security\_setting commands.



**What security mechanisms and services are available?**

To find out which security mechanisms and services are supported at the Replication Server, execute the `admin security_property` command at the isql prompt:

```
admin security_property[, security_mechanism]
```

Replication Server displays the name of supported security mechanisms, the security services available for that mechanism, and whether or not those services are supported at your site.

**What are the current security settings?**

To determine the status of supported security services, use the `admin security_setting` command. You can view status information for security parameters that have been set for routes and/or the ID Server. You can use the following syntax, where `rs_idserver` is the name of the ID Server and `rep_server` is the name of the destination Replication Server:

```
admin security_setting[, rs_idserver |, rep_server ]
```

**Mapping a security system login to a Replication Server login**

Your network-based security mechanism may use login names that are not valid on Replication Server. For example, login names on Replication Server must not exceed 30 characters or include certain special characters such as \*, (, and %. Login names on Replication Server must be valid identifiers, which are described in “Identifiers” in Chapter 2 of the *Replication Server Reference Manual*.

If the security login name is not a valid identifier, Replication Server automatically maps invalid characters to valid characters and truncates login names at 30 characters. Table 8-12 describes how Replication Server translates invalid characters.

**Table 8-12: Replication Server converts invalid characters**

Invalid characters	Convert to
\ % & , : = > ' ' ~	an underscore: _
! ^ ( ) . < ? { }	a dollar sign: \$
“ - ; * + / [ ]	a pound sign: #

## Using more than one security mechanism

If your replication system supports multiple security mechanisms, you may need to install more than one security mechanism on your Replication Server to ensure that both ends of each pathway can support the same mechanism. In this scenario, you can:

- 1 Configure the Replication Server, for all routes, connections, and other pathways, using `configure replication server`. Make sure that the default security mechanism name is the first one listed under SECURITY in the `libtcl.cfg` file.
- 2 Configure security for the individual pathways that use a different security mechanism. Make sure that the security mechanism is listed in `libtcl.cfg`. Table 8-7 lists pathways and the methods for securing them.

To find out the security mechanisms and supported security parameters of the Replication Server, use the `admin security_property` command. To find out the security mechanisms and current settings of a particular pathway, use the `admin security_settings` command. Refer to “Viewing information about security services” on page 234 for more information.

## Potential security issue

If different security mechanisms are used at the primary and replicate databases and Adaptive Server user names cannot be guaranteed unique at these sites, a potential security breach exists for request stored procedures.

If this scenario exists on your system, you can make sure that security is maintained by turning “off” the `dsi_exec_proc` parameter for the connection with the primary database. Executing `alter connection` and turning `dsi_exec_proc` “off” disables the Replication Server request-stored-procedures feature.

Here is the syntax:

```
alter connection to data_server.database
set dsi_exec_request_sproc 'off'
```

## Managing SSL security

The Replication Server secure sockets layer (SSL) Advanced Security option provides session-based security. SSL is the standard for securing the transmission of sensitive information, such as credit card numbers and stock trades, over the Internet.

### SSL overview

SSL, also called Transport Layer Security (TLS), provides a lightweight, easy-to-administer security mechanism with several encryption algorithms. It is intended for use over those database connections and routes where increased security is required.

SSL uses certificates issued by certificate authorities (CAs) to establish and verify identities. A certificate is like an electronic passport; it contains all the information necessary to identify an entity, including the public key of the certified entity and the signature of the issuing CA.

This document provides instructions for setting up SSL on Replication Server. See documentation from your third-party SSL security mechanism for instructions for using that software. See also the Internet Engineering Task Force (IETF) Web site for additional information.

An SSL installation requires these items:

- *Certificate authority* – a valid entity that verifies and signs certificates. Each CA has its own verification policies for issuing digital signatures.
- *Certificate* – an electronic document that identifies a server, a user, an organization, or other entity. A certificate contains the public key of the certified entity and a signature of the issuing CA.
- *Filter* – a special network driver that filters information delivered to and from a port.
- *Identity file* – concatenates a certificate and the certificate's private key.
- *Trusted roots file* – contains a list of certificates. Open Client/Server accepts only those CAs listed in the trusted roots file.
- *CipherSuites* – a set of cryptographic algorithms for authenticating a client and server, transmitting certificates, encrypting data, and establishing security session keys.

The SSL protocol runs above TCP/IP and below application protocols such as HTTP or TDS. Before the SSL connection is established, the server and client exchange a series of I/O round trips to negotiate and agree upon a secure encrypted session. This process is called the SSL handshake.

## The SSL handshake

The standard SSL handshake consists of these steps:

- 1 The client sends a connection request, which includes the SSL options the client supports, to the server.
- 2 The server returns its certificate and a list of supported encryption algorithms called CipherSuites, key-exchange algorithms, and digital signatures.
- 3 Both client and server agree on a CipherSuite, and a secure, encrypted session is established.

## SSL on Replication Server

Replication Server does not directly invoke SSL APIs. Replication Server SSL support is based on functionality provided by Sybase Open Client/Server. Sybase uses the SSS Plus™ library API from Certicom to support SSL in Open Client/Server applications. See the *Open Server Server-Library/C Reference Manual* for a complete description of Open Client/Server support for SSL.

Replication Server Advanced Security option supports server authentication and data encryption; it does not support client authentication. For incoming connections, Replication Server supports both SSL and non-SSL ports. For outgoing connections, Replication Server supports both SSL and non-SSL ports *on the target server*. Clients must log in to the server using a user name and password. Replication Server verifies the user name and password. Once this connection is made, a secure encrypted session can be established.

Use of SSL-secured links can impact Replication Server performance. Sybase recommends SSL only for those connections or routes that transmit sensitive data.

## Requirements

- RS 15.0 or later supports TLS version 1.0 or 2.0; it does not support SSL version 3.0.

- SSL requires Replication Server version 12.5 and later. Earlier versions of Replication Server do not support SSL.
- The Replication Server Administrator must generate and secure the server certificates and trusted root CA certificates as files outside Replication Server.

## Setting up SSL security

Before setting up SSL services on Replication Server review the SSL Plus user documentation and documentation for any third-party SSL security software you are using.

To set up SSL services on Replication Server, follow these steps:

- 1 Edit `$$SYBASE/$SYBASE_OCS/config/libtcl.cfg` to include SSL driver location.
- 2 Edit `$$SYBASE/config/trusted.txt` to include trusted CA certificates.
- 3 Obtain a certificate from a trusted CA for each Replication Server accepting SSL connections.
- 4 Create the identity file that concatenates a certificate and its private key.
- 5 Use `rs_init` to enable SSL on Replication Server and to add an encrypted SSL password to the Replication Server configuration file.

---

**Note** You can also enable and disable SSL on Replication Server using `configure replication server` and the `use_ssl` option.

---

- 6 Create an SSL entry in the Replication Server interfaces file or directory service.
- 7 Restart Replication Server.

See the *Replication Server Configuration Guide* for detailed instructions for each of these steps.

## Enabling SSL security

You can enable or disable the SSL security feature using `rs_init`; you can also enable or disable SSL using `configure replication server` with the `use_ssl` option.

To enable configure replication server, enter:

```
configure replication server
set use_ssl to 'on'
```

Set `use_ssl` to “off” to disable SSL. By default, SSL is not enabled on Replication Server. When `use_ssl` is “off,” Replication Server does not accept SSL connections.

`use_ssl` is a static option. You must restart Replication Server after you change its value.

# Managing Replicated Tables

This chapter describes setting up and managing replicated tables.

Topic	Page
Introduction	242
Planning a replication system	243
Summarizing the process	245
Creating replication definitions	250
Marking tables for replication	270
Replicating Java columns	273
Replicating text, unitext, image, and rawobject columns	277
Replicating new large-object (LOB) datatypes	287
Replicating computed columns	288
Replicating encrypted columns	289
Working with special datatypes	291
Modifying replication definitions	293
Modifying replicated data	305
Using publications	308
Translating datatypes using HDS	317

You can copy data from one database to another in different ways depending on which method best suits the needs at your site:

- Using a single database replication definition that lets you choose whether or not to replicate individual tables, transactions, functions, system stored procedures, and data definition language (DDL).

See Chapter 12, “Managing Replicated Objects Using Multisite Availability,” for more information about database replication definitions and multisite availability (MSA).

- Using function replication definitions, where each one identifies a specific system stored procedure for replication.

See Chapter 10, “Managing Replicated Functions,” for information about setting up and managing function replication definitions.

- Using table replication definitions, where each one identifies a specific table for replication and, optionally, specifies a subset of columns to be replicated.

This chapter discusses the preparations, the procedures, and the specific commands used to manage replicate tables, table replication definitions, and publications.

## Introduction

Replication Server allows you to copy and update data from a table in one database—the *primary*—to a table in another database—the *replicate*.

---

**Note** The primary database is also referred to as the “source.” The replicate database is also referred to as the “destination.”

---

To establish a table as the source, you create a *replication definition* that specifies the location of the data you want to copy and describes the structure of the table in which the data resides.

Before you copy data from the source table, you must also create a duplicate of the table in the destination data server. Then, in the Replication Server that manages the destination table, you create a *subscription* to the replication definition. A subscription resembles a SQL `select` statement.

If you do not want to duplicate all of a table’s data, Replication Server lets you specify a subset of columns to copy in the replication definition or use a `where` clause in the subscription to specify a subset of rows to receive.

You can include replication definitions for related tables and stored procedures in a **publication** and then create subscriptions against all of them as a group. When you use publications you can organize your subscriptions and monitor status information for all subscriptions in the group with a single command.

You can change the datatype of replicated values using the heterogeneous datatype support (HDS) feature. HDS allows you to translate the datatype of a replicated column value to a datatype acceptable to the replicate data server. You can use HDS in Sybase environments, in non-Sybase environments, and in mixed Sybase and non-Sybase data server environments.



See Chapter 11, “Managing Subscriptions” for information on creating subscriptions for individual replication definitions and for publications. See “Subscription example” on page 382 for an example of the entire transaction replication process.

## Planning a replication system

This section summarizes the information you need to consider when planning your replication system. See the *Replication Server Design Guide* for more information.

### Design considerations

When you set up a replication system, consider the following:

- Security, including user login names and passwords, permissions required for executing commands, and third-party security systems. See Chapter 8, “Managing Replication Server Security”.
- Concurrency control; specifically, protecting your replication system from conflicts that may result from data being modified by one client when it is also being used by another. See “Transaction Management” in Chapter 1 of the *Replication Server Design Guide*.
- CPU, memory, disk, and network resources. See Appendix A, “Capacity Planning,” in the *Replication Server Design Guide*.
- Consider your replicated data model and routing scheme. See Chapter 1, “Introduction” and Chapter 6, “Managing Routes”.
- Requirements for using heterogeneous data servers as data sources or data destinations. Refer to “Heterogeneous Data Server Support” in Chapter 1, “Introduction,” in the *Replication Server Design Guide*.
- Compatibility between Adaptive Servers and Replication Servers of different versions. See “Restrictions on data replication” on page 244.

For information about Sybase compatibility issues, see the release bulletin for your platform.

## Restrictions on data replication

When you design your replication system, you should also consider the following restrictions.

- Adaptive Server and Replication Server system tables cannot be copied during normal replication. However, the execution of supported commands and system procedures on certain system tables can be copied in warm standby applications. Refer to “What information is replicated?” on page 61 in the *Replication Server Administration Guide Volume 2* for more information. In addition, some data is automatically copied between RSSDs in the replication system.
- Tables that you want to copy must have unique primary keys.
- Client applications should not update unique index or primary key columns in a way that a key could duplicate the key of another row. Because of the way Replication Server copies transactions, this type of update could result in duplicate rows or errors at replicate databases.

For example, if `pk_col` is the primary key column for `table1`, the following command could cause errors or incorrect data at the replicate database:

```
update table1
set pk_col = pk_col + 1
```

If there is a primary key or unique index constraint on the replicate table, the updates fail and the DSI thread for the replicate database is suspended.

- If a trigger is associated with a replicate table, do not put a commit statement inside the trigger. Triggers that contain commit statements at replicate sites may cause a duplicate key and make Replication Server recovery fail.
- Replication Servers of different versions can work together in the same replication system, but certain features may be restricted. See “Mixed-version replication systems” on page 18 for more information.
- Virtual computed columns cannot be replicated since Replication Agent cannot forward virtual columns to Replication Server, and Replication Server cannot insert or update virtual columns.

## Preparing a replication system

Before you replicate data, complete the following preparatory tasks:

- Set up the replication system:

- Install Replication Servers. See the Replication Server installation and configuration guides for your platform.
- Create the databases that will be the primary and replicate. See the *Adaptive Server Enterprise Reference Manual* or the documentation for your non-Sybase database software.
- Establish connections from Replication Servers to the primary and replicate databases.  
  
See the *Replication Server Configuration Guide* for your platform and Creating database connections in Chapter 7, “Managing Database Connections.”
- Establish all necessary routes between Replication Servers. See Chapter 6, “Managing Routes.”
- Configure and start up database RepAgent for source databases. See Chapter 4, “Managing a Replication System.”
- Verify that all replication system components are working. See Chapter 1, “Verifying and Monitoring Replication Server” in the *Replication Server Administration Guide Volume 2*.

Refer to the Replication Server installation and configuration guides for your platform for more information. Also see Chapter 4, “Managing a Replication System” for more information about starting and stopping Replication Servers.

## Summarizing the process

This section describes how to replicate data between tables. For more information, see “Specifying data for replication” on page 32, and the following tables:

- Table 9-1 on page 249
- Table 10-1 on page 333
- Table 11-3 on page 370

For information about how to group replication definitions in publications and create publication subscriptions against them, see “Using publications” on page 308 and “Using publication subscriptions” on page 395.

## Replication procedure

The following procedure summarizes the steps required to replicate data using table replication definitions and subscriptions, and where to turn for detailed instructions. For an example of the entire process, see “Subscription example” on page 382.

- 1 Be sure you understand the issues described in “Planning a replication system” on page 243. Verify that you prepared the replication system as described under “Preparing a replication system” on page 244.
- 2 Create the table as the Database Owner in the primary database, if it does not already exist, or, if there is a different table owner, specify the table owner name when you create the replication definition.
  - In Adaptive Server, use `create table` to create the table, or use `sp_help` to verify that the table exists.
  - If you are replicating data from a source other than Adaptive Server, create the table according to the instructions for your database software. Other data server steps in this procedure may vary for heterogeneous replication.
- 3 In the primary Replication Server, create one or more replication definitions for the table from which you want to copy data. Each replication definition can be subscribed to by a different site that uses a different table view.

When you create replication definitions, anticipate the requirements for the subscribing table, as described in step 8. The replication definition may contain all or a subset of the columns in the source table. It may specify the same or different table names, owner names, column names, or datatypes for the source and destination tables. It may change the datatype of the replicated value.

See “Using the create replication definition command” on page 251 for details. See also “Creating multiple replication definitions per table” on page 265.

- 4 If you are using publications, execute the following steps at the primary Replication Server.
  - Create one or more publications for the tables you want to replicate using `create publication`.

- Create one or more **articles**, replication definition extensions, for each replication definition you want to include in the publication using `create article`. You can include a `where` clause to specify a subset of rows to send to the destination database.
- Validate the publications, using `validate publication`, so that you can create subscriptions against them.

See “Using publications” on page 308 for more information about creating publications.

5 Mark the source table for replication.

In the primary Adaptive Server, use `sp_setreptable` to enable table replication. This step allows the RepAgent thread to forward transactions for the table to the primary Replication Server.

---

**Note** For non-Adaptive Server primaries, see your Replication Agent documentation for instructions on marking tables and columns.

---

See “Marking tables for replication” on page 270 for details.

6 If the source table contains text, `unitext`, image, or `rawobject` columns, you may need to use `sp_setrepcol` in the primary Adaptive Server to adjust the replication status for these columns.

---

**Note** For non-Adaptive Server primaries, see your Replication Agent documentation for instructions.

---

See “Replicating text, `unitext`, image, and `rawobject` columns” on page 277 for details.

7 Prepare a login name for the user creating the subscription. Login names that create subscriptions at destination Replication Servers must also exist at the source Replication Server.

See Chapter 8, “Managing Replication Server Security”

8 In the replicate database, create a table that matches the schema published by the replication definition. Create the destination table as the Database Owner or as the same table owner specified in the replication definition.

In Adaptive Server, use `create table` to create the table, or use `sp_help` to verify that the table exists.

The destination table may have the same or different name and/or the same or different owner name as the source table. It may contain all or a subset of the columns in the source table, with the same or different column names or datatypes. The replication definition must specify any such differences between the source and destination tables.

---

**Note** The destination table may include a column that is not in the replication definition if the column accepts null values, has a defined default value, or you use a custom function string to apply a value to that column.

---

- 9 Grant the replicate database maintenance user login name select, insert, delete, and update permissions on the destination table. The maintenance user executes commands for replicated transactions.

See Chapter 8, “Managing Replication Server Security”.

- 10 If necessary, customize your database operations using functions, function strings, and function-string classes. Replication Server function strings execute data server operations.

See Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2* for details.

- 11 Create a subscription in the replicate Replication Server. If you are using publications, proceed to step 12.

Log in to a replicate Replication Server and create one or more subscriptions to the table replication definition for the data you want to copy. You can subscribe to all the rows in the replication definition’s columns, or use a *where* clause to copy only certain rows.

A replicate database can subscribe to multiple replication definitions of a primary (source) table, but a replicate table can subscribe to only one replication definition of a source table.

When you create a subscription, the destination table is filled in with the initial table data in a process called *materialization*. In most cases, Replication Server copies data into the destination table automatically. You can also manually materialize the data.

See Chapter 11, “Managing Subscriptions” for more information about creating and materializing subscriptions.

- 12 If you are using publications, create a publication subscription against the publications created in step 4. Execute create subscription at the replicate Replication Server.

When you create a publication subscription, Replication Server creates subscriptions against each article in the publication. Article subscriptions do not contain `where` clauses.

See “Using publication subscriptions” on page 395 for more information about publication subscriptions.

13 Check the subscription status.

Verify that the subscription data has fully materialized in the replicate database and that transactions are replicating successfully.

See Chapter 11, “Managing Subscriptions” for details.

## Commands for managing table replication definitions

Table 9-1 lists the Replication Server commands for working with table replication definitions.

**Table 9-1: Commands for managing table replication definitions**

Command	Task
create replication definition	Creates a replication definition for a primary table, which describes the columns you want to copy, the location of the table, and other information. See “Creating replication definitions” on page 250.
alter replication definition	Modifies an existing replication definition in a variety of ways, including: <ul style="list-style-type: none"> <li>• Adding columns</li> <li>• Add or drop primary keys</li> <li>• Add or drop searchable columns from the replication definition</li> <li>• Specifying different replicate table, table owner, column name, or datatype</li> <li>• Changing minimal column replication</li> <li>• Add or alter column-level datatype translations</li> <li>• Change <code>text</code>, <code>unitext</code>, <code>image</code>, or <code>rawobject</code> column replication status</li> <li>• Change how the replication definition is used in replicating to a standby database</li> </ul> See “Altering replication definitions” on page 298.
drop replication definition	Removes a replication definition from the replication system. You must drop all subscriptions for a replication definition before you can drop the replication definition. See “Dropping replication definitions” on page 304.

## Creating replication definitions

A replication definition describes the source table to Replication Server, specifying the columns you want to copy. It may also describe attributes of the destination table. Destination tables that match the specified characteristics can subscribe to the replication definition. You can create multiple replication definitions for the same primary table, each customized for a particular use. See “Creating multiple replication definitions per table” on page 265.

To create a replication definition, use `create replication definition` at the Replication Server managing the source table. See “Using the `create replication definition` command” on page 251.

Information about each replication definition is sent to each qualifying Replication Server with a route from the primary Replication Server. Replication Server version 11.5 (and later) receives information about all replication definitions. Replication Server version 11.0.x (and earlier) receives information about no more than one replication definition per primary table. See “Replication definition restrictions in mixed-version systems” on page 268 for details.

Replication definitions are stored in the `rs_objects` and `rs_columns` system tables in the RSSD. The primary version of a replication definition resides at the primary Replication Server.

## Replication definition settings

Each replication definition must include the following information:

- The name of the replication definition.
- The names of the source and destination tables.

Replication Server assumes that the replication definition name is the name of *both* the source and destination tables, unless you specify differently.

- The name of the data server and database where the source table is located.
- The column names and datatypes that you want to copy. You can copy all or a subset of the source table’s columns. The replicate column names and datatypes are the same as the primary column names and datatypes, unless you specify differently.
- The primary key—one or more columns that uniquely identify each row in the source table.



Optionally, a replication definition may also include:

- The names of the owners of the source and destination tables. The default table owner is the Database Owner (dbo).
- The names of searchable columns—columns that can be specified in the where clause of a subscription to indicate the rows from the primary table to copy into the destination table.
- For a warm standby application, whether to use the replication definition to copy data into a standby database and whether to copy all of the table's columns or just the columns in the definition's column list.
- Whether to copy only the minimal number of columns required for update and delete operations. This option may enhance overall system performance.
- Replication options for text, unitext, image, and rawobject columns.
- Column-level datatype translations

## Using the *create replication definition* command

Use *create replication definition* to describe characteristics to Replication Server of a table you want to replicate.

Execute *create replication definition* at the Replication Server that manages the source table's database. A replication definition must include the name of the source data server and database.

The following example creates a basic replication definition named *publishers* for source and destination tables with the same name. The primary database is *pubs2* managed by the *TOKYO\_DS* data server. All of the table's columns are included and the *pub\_id* column is specified as the primary key.

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
```

Each part of the command is discussed in the following subsections. See *create replication definition* in Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for complete command syntax and usage guidelines.

## Specifying the replication definition name and table names

A replication definition has a global **name space**—that is, at every Replication Server with routes from the primary Replication Server, the name refers to the same replication definition.

Replication Server cannot always enforce the unique-name requirement when you enter create replication definition. You must ensure that there is no existing replication definition (table or function) with the same name when you create a new replication definition.

By default, the replication definition name is the name of *both* the source and destination tables.

In some instances, you may need to use different names for your source and destination tables, or different names for your tables and replication definitions. Include one of the optional clauses with all tables named, with primary table named, or with replicate table named to specify table names where they differ from the replication definition name.

### When source and destination tables share the same name

When the source table and all destination tables share the same name but you want to give the replication definition a different name, use with all tables named to specify the table names.

For example, to create a replication definition named `publishers_rep` for source and destination tables named `publishers`, enter this command:

```
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with all tables named publishers
...
```

### When source and destination tables have different names

When the source table and any destination tables have different names, use with primary table named to specify the name of the source table, or use with replicate table named to specify the destination table name. You can use one of these clauses or both of them together.

If you don't specify different table names, the replication definition name is assumed by Replication Server to be the name of *both* the source and destination tables.

For example, to create a replication definition named `publishers_rep` for a source table named `publishers1` and destination tables named `publishers2`, enter:

```
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with primary table named publishers1
with replicate table named publishers2
...
```

For a replication definition and a source table named *publishers*, and destination tables named *publishers2*, enter:

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
with replicate table named publishers2
...
```

In this example, the *publishers* replication definition also becomes the source table's name.

### Specifying the name of the source or destination table owner

You can specify the table owner's name as an optional qualifier along with the name of the source or destination table. Data server operations may fail if the table owner does not correspond to what is specified in the replication definition.

For example, to create a replication definition for the *publishers* source table and the *publishers2* destination table owned by the user "ravi," enter:

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
with replicate table named ravi.publishers2
...
```

### Specifying column names and datatypes

When you create a replication definition, you list the names and datatypes of the columns from the table that you want to copy.

A column's name and datatype will be the same in the replicate table as in the primary table unless you specify a different replicate (published) column name or datatype.

Enclose the names of all of the columns and their datatypes in parentheses. For multiple columns, separate each column and its datatype from the next column with a comma.

For example, the following command creates a replication definition named *publishers\_rep1* for source and destination tables named *publishers*. It includes all the columns and their datatypes.

```
create replication definition publishers_rep1
with primary at TOKYO_DS.pubs2
with all tables named publishers
(pub_id char(4),
pub_name varchar(40),
city varchar(20),
state char(2))
primary key (pub_id)
```

The following command creates a replication definition named `publishers_rep2` that omits the `city` column. Destination sites that do not require this column can subscribe to this replication definition.

```
create replication definition publishers_rep2
with primary at TOKYO_DS.pubs2
with all tables named publishers
(pub_id char(4),
pub_name varchar(40),
state char(2))
primary key (pub_id)
```

Performance is best if columns are listed in the same order in the replication definition as in the tables themselves.

You can use only datatypes supported by Replication Server. If a primary table has columns with user-defined datatypes, you must use a compatible supported datatype in the replication definition. You can also employ user-defined datatypes supplied with Replication Server after you install them.

Refer to “Datatypes” in Chapter 2, “Topics,” in the *Replication Server Reference Manual* for complete details on the datatypes supported by Replication Server.

### When source and destination columns have different names

When you want only one replication definition for a source table, and the source column names differ from their destination counterparts, use the `column_name` as `replicate_column_name` clause in the replication definition.

For example, for a source table named `publishers1` and a destination table named `publishers2`, where the source column `pub1_name` corresponds to the destination column `pub2_name`, enter this:

```
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with primary table named publishers1
with replicate table named publishers2
(pub_id char(4),
```

```
pub1_name as pub2_name varchar(40),  
city varchar(20),  
state char(2)  
primary key (pub_id)
```

### Datatypes in multiple primary table replication definitions

When you create multiple replication definitions for the same source table, the declared column datatype (the column datatype in the primary table) must be the same, except when the column's datatype is `rawobject` or `rawobject in row`, which correspond respectively to the `image` and `varbinary` datatypes.

Specifically you can:

- Declare a column's datatype as `rawobject` in one replication definition, but declare the same column's datatype as `image` in another replication definition for the same table
- Declare a column's datatype as `rawobject in row` in one replication definition, but declare the same column's datatype as `varbinary` in another replication definition for the same table

The replicate (published) column datatype can be different between replication definitions for the same table, with no restrictions.

When a column is listed in an existing replication definition for a primary table, specifying the column datatype is optional in subsequent replication definitions for the same primary table—the datatype is inherited from the previous replication definition and retained for the subsequent definition, even if the first definition (where you specified the datatype) is dropped.

To change a column datatype, use the `alter replication definition` command. Refer to “Altering column datatypes” on page 302.

### Additional columns in the replicate table

The replicate table may include a column that is not in the replication definition if the column has a defined default value or you use a custom function string to apply a value to that column.

Columns can be specified to accept null values in `create table`. When source rows are copied to the destination table, extra columns are filled with null values or may be updated separately by the local data server.

### Including *text*, *unitext*, *image*, and Java columns

- To copy text, unitext, image, or the Java datatypes rawobject and robject in row column data to any destination site, include those columns in the replication definition. Replicating text, unitext, image, or Java columns involves additional special procedures and considerations.

See “Replicating text, unitext, image, and rawobject columns” on page 277 and “Java datatypes in Replication Server” on page 274 for more information.

### Using Special Datatypes

To distribute updates to particular sites, use the `rs_address` special datatype. See “Using the `rs_address` datatype” on page 291 and “Bitmap subscriptions” on page 388 for more information.

You can use the identity special datatype if the table you are copying contains an identity column. See “Replicating identity columns” on page 291 for more information.

You can also use the timestamp special datatype if the table you are copying contains a timestamp column. See “Replicating timestamp columns” on page 292 for more information.

### Using user-defined datatypes

To change the datatype of the replicated value at the primary database to a datatype acceptable to the replicate database, use user-defined datatypes. See “Translating datatypes using HDS” on page 317 for more information.

### Specifying the primary key

The **primary key** is the column or combination of columns that uniquely identifies each row. Although many data servers, including Adaptive Server, allow tables that contain duplicate rows, Replication Server requires that the source and destination tables have unique values for the primary key columns in each row.

You must include the primary key clause in create replication definition to identify the primary key columns in the source table. Primary key columns must also be included in the column list.

When Replication Server applies the default `rs_update` or `rs_delete` function string at a destination site, it specifies values for the primary key in the where clause of the update or delete statement.

Enclose the names of the primary key columns in parentheses. For example:

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
```

For multiple primary key columns, separate each column from the next with a comma.

---

**Note** You cannot include columns of datatypes , text, unitext, image, rawobject, rawobject in row or rs\_address as part of the primary key.

---

## Specifying searchable columns

Use searchable columns in create replication definition to specify which columns to use in the where clause of create subscription or define subscription (or create article for publications) to restrict the rows copied to a subscribing site. If you do not include a searchable columns clause in a replication definition, you cannot use a where clause in a subscription or article that references that replication definition.

Enclose the names of the searchable columns in parentheses. For multiple searchable columns, separate each column from the next with a comma.

In the following example, three columns, pub\_id, pub\_name, and state, are specified as searchable columns. You can include any of these columns in a subscription's where clause.

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
searchable columns (pub_id, pub_name, state)
```

See “Using the where clause” on page 371 for additional information on using where in subscriptions.

## Restrictions on searchable columns

Searchable columns have these restrictions:

- You cannot specify text, unitext, image, or Java rawobject or rawobject in row columns as searchable columns.

- Columns included in the searchable columns clause cannot have null values.
- To perform bitmap comparison using the where clause in the subscription, you must include any columns that use the `rs_address` datatype in the replication definition's searchable columns clause. See “Using the `rs_address` datatype” on page 291 for more information.
- The more searchable columns in the searchable columns list of a replication definition, the slower Replication Server processes subscriptions; that is, the fewer searchable columns, the more efficiently Replication Server evaluates rows against subscriptions for the table.

## Replicating the minimal set of columns

Normally, Replication Server sends all the columns in each row when applying updates and deletes, as well as inserts, in each replicate database. Replication Server normally sends *maximum columns* to the standby database—if replication definitions are not used for the table or the replication definitions are not used for the standby connection.

---

**Note** You must send all columns when replicating to SQL Remote databases. Do not send minimal columns or replication will fail.

---

To enhance replication system performance, specify `replicate minimal columns` in `create replication definition`. This clause lets you send only those columns that are required for delete and update operations to replicate databases.

When you set `replicate minimal columns`:

- For a delete operation, the source Replication Server sends only the primary key columns to destination Replication Servers or the standby database.
- For an update operation, the source Replication Server sends only the columns modified by the update operation and the primary key columns, to destination Replication Servers or the standby database.

---

**Note** `replicate minimal columns` does not apply to insert operations, for which all columns are copied.

---



A destination Replication Server uses the primary key columns in constructing the data server commands that it applies to the replicate or the standby database.

The following replication definition includes replicate minimal columns:

```
create replication definition publishers
with primary at TOKYO_DS.pubs2
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
replicate minimal columns
```

### Changing minimal columns setting

Use alter replication definition to change an existing replication definition to replicate only the minimal set of columns or to replicate all columns.

### Minimal columns and *rs\_update* and *rs\_delete* function strings

If you specify replicate minimal columns and need to create non-default *rs\_update* and *rs\_delete* function strings, use the *rs\_default\_fs* function string variable to represent the default function string behavior. See “Using the default system variable” on page 49 in the *Replication Server Administration Guide Volume 2* for details.

### Minimal columns and *autocorrection*

If you specify replicate minimal columns, you cannot also specify autocorrection, which corrects discrepancies that may occur during materialization by converting each update or insert operation into a delete followed by an insert.

If you set autocorrection on before you specify minimal columns (for example, using alter replication definition), autocorrection is not performed. Replication Server logs informational messages for any update operations.

You must set autocorrection on when you create a subscription using nonatomic materialization. If minimal column replication is set for the replication definition and you create a new subscription that uses nonatomic materialization or the bulk materialization method that simulates nonatomic materialization, autocorrection cannot resolve inconsistencies.

See Chapter 11, “Managing Subscriptions” for details on materialization methods. See “Using autocorrection” on page 356 for more information on this command.

## Using replication definitions with warm standby applications

You do not need to use replication definitions with warm standby applications. However, you can use them to control the flow of information to the standby database—even though no subscriptions are needed. You can create replication definitions just for replication to the standby database or use existing replication definitions for this purpose.

Use `send standby` in create replication definition as follows:

- Use `send standby` in any form to replicate transaction data into the standby database using this replication definition. Replication Server uses the replication definition's primary key and minimal columns setting.

See “Specifying the primary key” on page 256 and “Replicating the minimal set of columns” on page 258 for more information.

- Use `send standby` or `send standby all columns` to send all columns in the table to a standby database.
- Use `send standby replication definition columns` to send only the columns specified in the replication definition to a standby database.

If you omit `send standby`, another replication definition may be used in replicating data for this table to the standby database, or no replication definition may be used.

The replication definition in the following example replicates transactions to a standby database. The primary key and minimal set of columns settings will be used in standby replication. Only the columns specified in the replication definition will be replicated into the standby database—the `city` column is omitted from this replication definition.

```
create replication definition publishers_ws
with primary at LDS.pubs2
with all tables named 'publishers'
(pub_id char(4),
pub_name varchar(40),
state char(2))
primary key (pub_id)
send standby replication definition columns
replicate minimal columns
```

If a replication definition already exists for the same primary table and is marked for use by the standby, creating a new replication definition using `send standby` (or altering another replication definition) unmarks the previous replication definition as being used by the standby.

See “Using replication definitions and subscriptions” on page 111 in the *Replication Server Administration Guide Volume 2* for more information about using replication definitions with warm standby applications.

### **Specifying *text*, *unitext*, and *image* column replication**

To create a replication definition for a table that contains text, unitext, or image columns datatypes:

- Include each text, unitext, or image column that you want to replicate in the column list, and
- Include each column in the optional clauses `replicate_if_changed` or `always_replicate`.

In each clause, enclose the names of the text and image columns in parentheses. For multiple columns, separate each column from the next with a comma.

- Ensure that each text, unitext, or image column has a corresponding status in Adaptive Server.

See “Replicating text, unitext, image, and rawobject columns” on page 277 for more information on replicating text, unitext, and image columns.

See “Replicating text, unitext, image, and rawobject data” on page 70 in the *Replication Server Administration Guide Volume 2* for information about replicating text, unitext, and image columns in warm standby applications.

## Specifying computed column replication

To create a replication definition for a computed columns use the base column datatype in the replication definition for materialized columns. Do not include virtual columns in the replication definition.

## Specifying *rawobject* and *rawobject in row* column replication

You can include Java columns in a replication definition. Replication Server replicates Java columns as either *rawobject* or *rawobject in row* datatypes. To create a replication definition for a table that contains Java datatypes:

- Include each *rawobject* or *rawobject in row* column that you want to replicate in the column list, and
- Include each *rawobject* column in the optional clauses *replicate\_if\_changed* or *always\_replicate*.

In each clause, enclose the names of the *rawobject* columns in parentheses. For multiple columns, separate each column from the next with a comma.

---

**Note** *rawobject in row* columns do not have replication status.

---

- Ensure that each *rawobject* column has a corresponding status in Adaptive Server.

See “Replicating text, unitext, image, and *rawobject* columns” on page 277 for more information about replicating Java columns.

## Specifying column-level datatype translations

You can specify column-level datatype translations in the replication definition. Sybase provides a set of datatype definitions that you install using instructions from the *Replication Server Configuration Guide* for your platform.

- The *declared\_datatype* defines the datatype of the value delivered to the Replication Server from the Replication Agent. It must be the Replication Server base datatype or a datatype definition for the datatype in the primary database.
- The *published\_datatype* defines the datatype of the value after a column-level translation. It must be the Replication Server base datatype or a datatype definition for the datatype in the replicate database.

See “Translating datatypes using HDS” on page 317 for detailed information about datatype translations.

## Creating replication definitions using extended limits

Replication Server version 12.5 and later can replicate wider columns, wider parameters, and larger numbers of columns than earlier versions. It can also handle wider data rows and wider messages.

Replication Server supports the extended limits capabilities of Adaptive Server version 12.5 and later. See the Adaptive Server documentation for more information. For information about using Replication Server extended limits with non-Sybase data servers, see the documentation for your Sybase Replication Agent and the *Replication Server Heterogeneous Guide*.

### Before you use extended limits

To use extended limits, make sure that both the primary and replicate Replication Server are upgraded to site version 12.5 or later, which automatically sets the LTL version to 400. In addition, make sure that all routes using extended limits are set to 12.5 or later. If you are using Adaptive Server, make sure that both the primary and replicate databases are set to version 12.5 or later. Both the primary and replicate databases must be configured for the same page size.

See “Replication definition restrictions in mixed-version systems” on page 268 for information about using extended limits with Replication Server version 12.1 and earlier. See also the Replication Server white paper “Using Adaptive Server Enterprise version 12.5 with Replication Server version 12.1 and earlier: Schema-length and compatibility issues.”

### Using extended limits

You can create replication definitions using extended limits for both replicate and standby databases. Extended limits are defined as:

- Wide columns – data rows containing more than 255 to a maximum of 32768 bytes.
- More columns – replication definitions containing more than 250 up to a maximum of 1024 columns in a replication definition.

- Wide data – data rows up to the size of the data page on the data server. Adaptive Server version 12.5 and later supports page sizes of 2K, 4K, 8K, and 16K.
- Wide messages – messages larger than 16K.

### Wide columns

Replication Server can replicate wide columns containing char, varchar, binary, univarchar, unichar, unitext or Java inrow data to a maximum of 32768 bytes. Maximum column width on each system may vary; it is a function of the total number of columns and the page size of the data server.

You can use wide columns as primary keys and searchable columns and in replication definition where clauses.

---

**Note** The maximum number of bytes in the where clause of a subscription or article is 255 bytes. You cannot use wide columns in the where clause of subscriptions or articles.

---

### More columns

You can include as many as 1024 columns in a replication definition. As long as the total number of columns does not exceed 1024, Replication Server does not limit the number of primary key or searchable columns.

Replication Server uses primary key columns to build where clauses of SQL statements for the data server. Consider data server limitations when determining the actual number of columns available for primary keys in replication definitions.

Similarly, although Replication Server imposes no limits on the number of searchable columns in a replication definition, the number of columns in the where clause of a subscription or article may also be constrained by data server limitations.

### Wide data

Data rows can equal the size of the data page on the data server. Adaptive Server version 12.5 and later supports page sizes of 2K, 4K, 8K, and 16K.

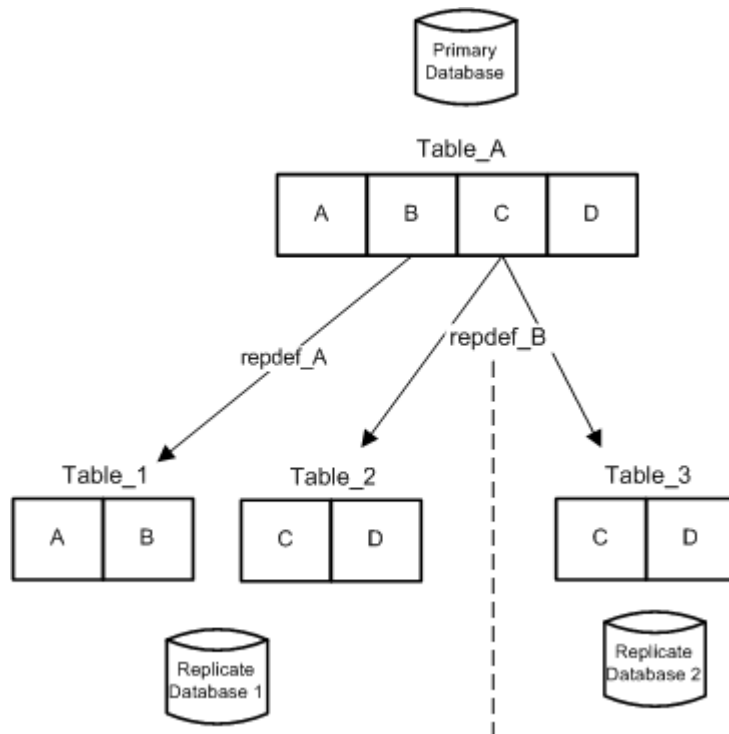
**Wide messages**

Replication Server copies data rows as messages in stable queues managed by the SQM. These messages contain before and after images of replicated data rows as well as other information. They require significantly more space than the data rows on which they are based. With extended limits, messages can span blocks and are no longer limited to 16K.

**Creating multiple replication definitions per table**

You can create multiple replication definitions for the same primary table and customize each one so that it can be subscribed to by a replicate table whose characteristics are different from those of the primary table or from other replicate tables.

For example, you can create two separate replication definitions for the same primary table, one that replicates columns A and B, and another that replicates columns C and D. Each subscribing site receives only the columns that it needs (see Figure 9-1).

**Figure 9-1: Using multiple replication definitions from one primary table**

In addition to describing the primary table, each replication definition can specify a smaller number of columns, different column names, different published datatypes, or a different table name for a replicate table. Replicate tables that match the specified characteristics can subscribe to the replication definition.

Different replication definitions created for the same primary table must use the same declared column datatype (unless the datatype is rawobject or rawobject in row) and the same null and not null status for text, untext, and image columns. To change a column's datatype or null status, use alter replication definition. Refer to "Altering column datatypes" on page 302 for instructions.

You can change replication status using alter replication definition. For example, you can change the replication status of text, untext, and image columns from replicate\_if\_changed to always\_replicate. The replication status for the column will also change for other replication definitions for the same primary table.

See "Creating multiple replication definitions per table" on page 265 for more information.



## Restrictions

When you have multiple replication definitions for the same primary table, the following restrictions apply:

- A replicate database can subscribe to multiple replication definitions. However, a replicate table can subscribe to only one replication definition of a particular primary table.
- A pre-version 12.0 Replication Server may not subscribe to replication definitions that either declare columns with User-Defined Datatypes or employ column-level translations.
- Different replication definitions created for the same primary table must use the same column datatype (unless the datatype is rawobject or rawobject in row) and, for text, unitext, image, and rawobject columns, the same null or not null status and the same replication status.

See “Specifying column names and datatypes” on page 253.

- You cannot create multiple replication definitions for a single primary stored procedure.
- Multiple replication definitions for one primary table are only supported in Replication Server version 11.5 and later; however, one replication definition can be marked and propagated to a Replication Server of a previous version, if compatible; that is, has the same primary and replicate table names, same primary and replicate column names, and does not include table owner name.

See “Replication definition restrictions in mixed-version systems” on page 268 for additional information.

## Replication definitions and function strings

Function strings map Replication Server functions to data server commands for execution in a database.

For each replication definition, the primary Replication Server creates default function strings for the **system functions** with replication definition scope (rs\_insert, rs\_update, rs\_delete, rs\_select, and so on). These function strings are distributed with the replication definition to other qualifying Replication Servers with routes from the primary Replication Server.

Some circumstances may require you to create the function strings for system functions (that is, Replication Server does not create them for you).

See Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2* for details on function strings and function-string classes. See Chapter 2, “Replication Server Technical Overview” for more information about how Replication Servers share information.

## Replication definition restrictions in mixed-version systems

Replication Server version 12.5 and later can handle messages larger than 16K only if the site version is also 12.5 or later. If the site version is 12.1 or earlier, messages larger than 16K may cause the stable queue to shut down. Similarly, if a replicate or intermediate site of a route is not set to a site version of 12.5 and later, messages larger than 16K may cause the route to shut down.

- If a large messages shuts down a stable queue, you can restart the queue using `resume queue`. To restart the queue and, optionally, to instruct Replication Server to skip the first large message encountered, enter the following commands, where *q\_number* is the number of the queue, and *q\_type* is either “0” for outbound queues or “1” for inbound queues:

```
resume queue, q_number, q_type[, skip transaction
with large message]
```

- To set default behavior for a stable queue encountering a large message, use `alter queue`. Enter:

```
alter queue, q_number, q_type,
set sqm_xact_with_large_msg to {skip | shutdown }
```

- If a large message has shut down a route, you can restart the route using `resume route`. You can enter the following commands, where *dest\_rep\_server* is the Replication Server to which the message is sent:

```
resume route to dest_rep_server
[skip transaction with large message]
```

This command applies only to direct routes.

- To set default behavior for a route encountering a large message, use `alter route`. Enter:

```
alter route to dest_rep_server
set sqm_xact_with_large_msg to
{ skip | shutdown }
```

- If you create or alter a replication definition that includes an identifier longer than 30 characters, only Replication Server 15.0 or later can subscribe to that replication definition.

- If you create or alter a replication definition that includes a rawobject or rawobject in row column, only Replication Server version 12.0 or later can subscribe to that replication definition.
- You can introduce column-level and class-level datatype translations only between Replication Servers of version 12.0 or later.
- If your replication system uses different versions of Replication Server (for example, version 11.0.x and version 11.5 or later), Replication Server version 11.0.x is subject to the following limitations:

- You cannot subscribe to a replication definition that specifies any of the following information:
  - Different source and destination table names
  - Different source and destination column names
  - Source or destination table owner names

Such a replication definition is incompatible with and unavailable to Replication Servers earlier than version 11.5.

- You can receive information about and subscribe to only one replication definition per table; however, when a Replication Server version 11.5 or later primary table has multiple replication definitions, the first replication definition created for the table can be marked and propagated to a Replication Server of a previous version, if it is compatible; that is, has the same primary and replicate table names, same primary and replicate column names, and does not include table owner name.

If you drop that replication definition, the next oldest 11.0.x-compatible replication definition created for that table is available for Replication Server version 11.0.x.

If subscriptions exist from Replication Server version 11.0.x, you must not alter an 11.0.x-compatible replication definition so that it is no longer compatible with 11.0.x. If you do so, that replication definition is no longer available to 11.0.x Replication Servers and the next oldest 11.0.x-compatible replication definition (if any) is available to the 11.0.x Replication Servers.

- You cannot create multiple replication definitions per table or customize them for destination tables.

- Unicode datatypes require Replication Server version 12.5 or later. If you are using Unicode datatypes in a mixed-version environment, see the *Replication Server Design Guide*.

See also “Mixed-version replication systems” on page 18 for more information.

## Marking tables for replication

After you create a replication definition for a table, use `sp_setreptable` to mark the table for replication. After a table is marked for replication, RepAgent begins forwarding the table’s log records to the Replication Server.

If you have marked a table for replication, you do not need to mark it again for another replication definition.

See “Subscription example” on page 382 for an example of setting up replication for one table.

---

**Note** Refer to your Replication Agent documentation for instructions on marking tables for replication in non-Sybase data servers.

---

## Using the `sp_setreptable` system procedure

To designate a primary Adaptive Server table for replication, use `sp_setreptable`. To use `sp_setreptable`, you must be the Database Owner or the System Administrator for the data server.

Refer to Chapter 5, “Adaptive Server Commands and System Procedures” in the *Replication Server Reference Manual* for more information about `sp_setreptable` command.

## Enabling replication

To mark a table for replication, log in to the Adaptive Server managing the database for that table and enter:

```
sp_setreptable table_name, 'true'
```

Marking the table in this way specifies that the table name must be unique.

---

**Note** Do not mark a table for replication unless you also create a replication definition for the table in the Replication Server managing that database. The RepAgent will begin forwarding to the Replication Server data for transactions for the affected table. If a replication definition does not exist, Replication Server may report message 32032 and its error log file may fill up. In addition, Replication Server performance may be significantly reduced. Warm standby applications, which do not require replication definitions, are not subject to this problem.

---

## Checking replication status

To check replication status for the table, enter:

```
sp_setreptable table_name
```

To check replication status for all tables in the database, enter:

```
sp_setreptable
```

## Enabling replication with *owner\_on* status

---

**Note** Refer to your Replication Agent documentation to see if your non-Sybase data server allows user tables with the same name but different owners.

---

User tables may have the same name but different owners. Adaptive Server allows you to mark a table for replication and specify that table owner information should be considered when identifying the table.

To mark the table for replication with the “owner on” status, log in to Adaptive Server and enter:

```
sp_setreptable table_name, 'true', owner_on
```

At the Replication Server, the replication definition for the table must identify the table owner. For example, if you set owner status for a table to “owner on” with `sp_setreptable`, you must include an owner name when you create the replication definition or Replication Server will be unable to find the correct table at the replicate database.

The owner of the source table and the owner of the destination table can be different.

---

**Note** If you specify “owner off” status for a table, Replication Server does not send table owner information to the *replicate* site. However, if you are replicating to a *standby* database, Replication Server sends “dbo” as the table owner.

---

Modifying the owner status of a table

You can change the owner status of a table previously marked for replication by using the `sp_setrepdefmode` system procedure.

To change the status of a table already marked for replication to “owner on,” log in to Adaptive Server and enter:

```
sp_setrepdefmode table_name, owner_on
```

To change the status of a table already marked for replication to “owner off,” log in to Adaptive Server and enter:

```
sp_setrepdefmode table_name, owner_off
```

You must reflect a change in owner status by including owner information in the replication definition. Use `create replication definition` at the Replication Server to create a new replication definition that includes the table owner.

Checking the owner status of a table

To check the owner status of a table, enter:

```
sp_setreptable table_name
```

## Disabling replication

To turn off replication for the table, enter:

```
sp_setreptable table_name, 'false'
```

---

**Note** Refer to your Replication Agent documentation for instructions on disabling replication in non-Sybase data servers.

---

## Replicating Java columns

You can replicate Java columns stored in your primary database to your standby and replicate databases. Replication Server passes Java objects through the replication system in serialized format without altering the Java objects in any way.

Refer to *Java in Adaptive Server Enterprise* for complete information about Java classes in the Adaptive Server database.

## Restrictions

Although you prepare replication definitions and subscriptions for Java columns in the usual manner, certain restrictions apply:

- Both the primary and replicate databases must be Sybase Adaptive Server version 12.0 or later.
- Replication Server does not replicate stored procedures that have Java objects as parameters. However, the effect of such a stored procedure can be duplicated through normal table replication.
- You cannot use Java columns as part of the primary key.
- You cannot evaluate Java columns in subscription expressions because Java columns are not searchable.

## Upgrade considerations

After you have upgraded the current Replication Server and set its site version to the current release, the Replication Manager route upgrade feature copies the replication definitions with Java columns from upstream Replication Servers to the current Replication Server.

Although Replication Server does not propagate replication definitions with Java columns to pre-12.0 version Replication Servers, you can replicate Java columns to older Replication Servers by manipulating function strings. See “Using function strings to replicate Java columns to older Replication Servers” on page 275 for more information.

## Java datatypes in Replication Server

Java columns pass through the replication system as one of two Replication Server datatypes:

- As `rawobject`, in which the information is stored in the database in a separate location in the same way that image data is stored. The base datatype of `rawobject` is `image`. This is the default datatype for Java columns in Replication Server. Replication Server handles `rawobject` data in the same way it handles image data.

Refer to “Replicating text, unitext, image, and `rawobject` columns” on page 277 for information about replication for `rawobject` columns.

- As `rawobject in row`, in which the information is stored in the database on consecutive data pages allocated to the table in the same way that, for example, `char` data is stored. The base datatype of `rawobject in row` is `varbinary(255)`. Replication Server handles `rawobject in row` data in the same way it handles `varbinary(255)` data.

`rawobject` and `rawobject in row` are compatible only with their base datatypes. They are not compatible with each other; that is, you cannot replicate `rawobject` to `rawobject in row` or vice versa.

The Replication Server reconciliation utility `rs_subcmp` treats Java datatypes as their base datatypes. Refer to the *Replication Server Reference Manual* for more information about `rs_subcmp`.

## Creating replication definitions for Java columns

You can create replication definitions for Java columns using `create replication definition` and the `rawobject` and `rawobject in row` datatypes.

When creating a replication definition:

- `rawobject` values have replication status. You can choose whether they are always replicated or replicated only if changed. They also have null status.

See “Replicating text, unitext, image, and `rawobject` columns” on page 277 for information about replication for `rawobject` columns.

- `rawobject in row` values do not have replication or null status.

`rawobject` and `rawobject in row` values:

- Cannot be part of the primary key.



- Cannot be evaluated in subscription expressions. Java columns are not searchable, and thus they cannot be used in a subscription or article where clause.

This example creates a sample replication definition p1 for a table t that contains Java columns.

```
create replication definition p1
  with primary at ds.db
  with all tables name t
  (c1 int,
   c2 rawobject null,
   c3 rawobject not null,
   c4 rawobject in row)
  primary key (c1)
  replicate_if_changed (c2)
  always_replicate (c3)
```

Columns c2 and c3 are rawobject columns; they have replication and null status. Column c4 is a rawobject in row column; it does not have replication or null status. Columns c2, c3, and c4 are neither part of the primary key nor are they searchable.

## Function strings for Java columns

Replication Server uses the `rs_raw_object_serialization` function string to pass Java columns to the replicate database in serialized format, which allows Replication Server to update Java columns directly. `rs_raw_object_serialization` is contained in `rs_sqlserver_function_class` and `rs_default_function_class`.

When a replication definition references the `rawobject` datatype, Replication Server creates `rs_get_textptr`, `rs_textptr_init`, `rs_datarow_for_writetext`, and `rs_writetext` function strings for each `rawobject` column just as it does for image data.

## Using function strings to replicate Java columns to older Replication Servers

Replication Server version 12.0 does not propagate replication definitions with Java datatypes to pre-12.0 Replication Servers. However you can replicate Java columns through older Replication Servers if you use the corresponding base datatype (`image` and `varbinary(255)`) and manipulate the `rs_usedb` and `rs_insert` function strings.

The following example illustrates the method.

- 1 Create tables containing Java columns in the primary and replicate databases:

```
create table tInfo
  (c1 integer,
   c2 Name rawobject in row,
   c3 Address rawobject null,
   c4 AccountInfo rawobject not null)
```

Name, Address, and AccountInfo are Java classes; c2, c3, and c4 are Java columns.

- 2 Create a replication definition for table tInfo.

If at least one of the Replication Server is pre-12.0, you must create a replication definition using the base datatypes for rawobject in row (varbinary(255)) and rawobject (image):

```
create replication definition tInfo1
with primary at DS-1.dbase
with all tables name TInfo
(c1 integer,
 c2 varbinary(255),
 c3 image null,
 c4 image not null,
 primary key (c1)
 ...
```

If the primary and replicate databases are managed by Replication Servers version 12.0 or later, a replication definition could be:

```
create replication definition tInfo
with primary at DS-1.dbase
with all tables named tInfo
(c1 integer,
 c2 rawobject in row,
 c3 rawobject null,
 c4 rawobject not null)
primary key (c1)
 ...
```

- 3 Alter the rs\_usedb and rs\_insert function strings for both the primary and replicate database connections. Refer to “Altering function strings” on page 41 in the *Replication Server Administration Guide Volume 2* for general information about customizing function strings.

- For rs\_usedb:

```
alter function string rs_usedb
for function_string_class_name
```

```
output language
`use ?rs_destination_db!sys_raw? set
raw_object_serialization on'
```

This change tells Adaptive Server to return Java column data as serialized binary values at subscription materialization. It also allows Replication Server to insert and update Java columns with serialized binary values.

- For `rs_insert`:

```
alter function string tInfo1.rs_insert
for function_string_class_name
output language
`insert tInfo(c1, c2, c4)
values (?c1!new?, ?c2!new?, 0xaced000574000130)'
```

This change alters `rs_insert` for `tInfo1` to insert the special binary value `0xaced000574000130` in column `c4`. If you do not alter `rs_insert`, the default value may cause Adaptive Server to return a serialization error.

So, you can create two replication definitions for the same table where the columns between the two replication definitions have different primary (declared) datatypes. If the primary Replication Server is version 12.0 or later, you can create both replication definitions `tInfo` and `tInfo1` for table `tInfo`. In this case, replicate Replication Servers version 12.0 and later can subscribe to `tInfo` and Replication Servers version pre-12.0 can subscribe to `tInfo1`.

---

**Note** You cannot use this method to replicate Java columns to standby databases. The standby connection uses the function-string class `rs_default_function_class`, which cannot be altered.

---

## Replicating *text*, *unitext*, *image*, and *rawobject* columns

Replication Server lets you replicate columns that use the Adaptive Server datatypes `text`, `unitext`, `image` and `rawobject`.

- When you replicate `text`, `unitext`, `image`, and `rawobject` columns you must specify a compatible replication status for each `text`, `unitext`, `image`, and `rawobject` column in both the replication definition and in Adaptive Server.

- You cannot include text, unitext, image, or rawobject columns as part of the primary key or as searchable columns.
- A unique set of columns must be identified so that text, unitext, image, or rawobject columns replication will only affect one row at the target database. This column or set of columns must be included in the primary key if a replication definition is used.
- To replicate text, unitext, image, and rawobject columns, follow these steps:
  - Use create replication definition to create a replication definition for a table that contains text, unitext, image, or rawobject columns.  
Refer to Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information about create replication definition command.
  - Use sp\_setreptable to mark the table for replication.  
See sp\_setreptable in Chapter 5, “Adaptive Server Commands and System Procedures” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.
- sp\_setreptable sets the replication status of text, unitext, image, or rawobject columns to always\_replicate.
- If you do not want to replicate some of the text, unitext, image, or rawobject columns, use sp\_setrepcol to change the replications status of those columns.  
See “Changing column status for text, unitext, image, or rawobject columns” on page 281 for instructions.
- Use create subscription to make subscriptions for the replication definition and begin replicating the text, unitext, image, or rawobject data.  
See “Using the create subscription command” on page 374.

---

**Note** When you execute an update at the primary database, you can update a text, unitext, image, or rawobject column and a non-text, non-image, or non-rawobject column--a char column, for example--with a single command. When those updates are copied to the replicate database, however, Replicate Server executes two commands, one for text, unitext, image, and rawobject updates and one for other datatype updates. If you choose to have DSI ignore certain replication errors, only a portion of the row may be replicated, which creates an inconsistent replicate table.

---

## Replicating large objects to non-ASE servers using DirectConnect Anywhere

Replication Server replicates large objects such as text and image to non-ASE servers by passing a `writetext` command to DirectConnect Anywhere™, where it is converted to an update statement. The `writetext` command include large-object pointers that an update statement uses to search and propagate the replicate database. Most data servers have their own unique implementation of updating large objects. Therefore, large-object replication to these servers can become slow and inefficient, often requiring a full table scan of the replicate database for a single update.

Replication Server provides an option to include primary keys with `writetext` commands sent to DirectConnect Anywhere. With the primary keys, DirectConnect Anywhere can create update statements that can efficiently search and replicate the replicate database.

Replication Server introduces the Data Server Interface (DSI) configuration parameter `dsi_alt_writetext`. You can use the `dsi_alt_writetext` to instruct the Replication Server to include a text pointer or a set of primary keys with the `writetext` command.

---

**Note** You need a version of ECDA 15.0 ESD #2 to use this feature.

---

See the *Replication Server Reference Manual* for more information.

## Creating a *text*, *unitext*, *image*, or *rawobject* replication definition

- When you create a table replication definition for *text*, *unitext*, *image*, or *rawobject* columns, use these guidelines:
- Include each *text*, *unitext*, *image*, or *rawobject* column that you want to replicate in the column list.
- Specify the datatype for each *text*, *unitext*, *image*, or *rawobject* column.
- Specify whether a null is allowed for the column in destination tables. This setting must be consistent with the way the source and destination tables are defined.
- Include each column in the optional clauses `replicate_if_changed` or `always_replicate`.

## Specifying a null value for *text*, *unitext*, *image*, and *rawobject* columns

To specify whether or not a null value is allowed in the replicate table for each text, unitext, image, or rawobject column, specify null or not null after the datatype for the column in the replication definition.

This setting must be consistent with the way the primary and replicate tables are defined. For text, unitext, image, and rawobject columns, the default is not null, meaning that the replicate table does not accept null values.

If you are using multiple replication definitions, the null value setting should be the same for all replication definitions on a primary table.

Do not specify null or not null for columns using datatypes other than text, unitext, image, or rawobject. Columns with null values cannot be searchable.

The following example replication definition for the table `au_pix` includes a column `pic` of datatype `image`, for which null values are allowed in replicate tables. The `pic` column is included in the `replicate_if_changed` clause.

```
create replication definition au_pix
with primary at TOKYO_DS.pubs2
(au_id char(11),
pic image null)
primary key (au_id)
replicate_if_changed (pic)
```

## Marking tables with *text*, *unitext*, *image*, or *rawobject* columns

Use `sp_setreptable` to set the initial replication status for text, unitext, image, and rawobject columns in Adaptive Server when you mark the table for replication. `sp_setreptable` sets the replication status of text, unitext, image, or rawobject columns to `always_replicate`.

---

**Note** If you do not want to replicate text, unitext, image, and rawobject columns, use `sp_setreptable` to mark the table for replication, which sets the replication status of text, unitext, image, and rawobject columns to `do_not_replicate`.

---

If you use `sp_setreptable` to mark a table for replication and the table includes text, unitext, image, or rawobject columns, an internal operation needs to be completed for every text, unitext, image, or rawobject column in every data row of the table. This internal modification is performed in a single transaction, and for large tables, this operation may be time-consuming and involve a significant amount of data.

Before you use `sp_setreptable` on a large table that has text, unitext, image, or rawobject columns, be sure that you have enough log space for this operation. You may also want to choose a time that will be least disruptive for client applications or replication system administration.

You can speed up the process of marking a table with text, unitext, image or rawobjects, if you use the option `use_index`. When using this option, Adaptive Server creates an internal nonclustered index for each text, unitext, image or rawobject, in the table. For example:

```
sp_setreptable aux_pix, true, null, use_index
```

See Chapter 5, “Adaptive Server Commands and System Procedures” in the *Replication Server Reference Manual* for more information about `sp_setreptable` command.

---

**Note** Refer to your Replication Agent documentation for instructions on marking tables for replication in non-Sybase data servers.

---

## Changing column status for *text*, *unitext*, *image*, or *rawobject* columns

When you mark a table with text, unitext, image, or rawobject columns for replication, `sp_setreptable` sets the replication status of text, unitext, image, or rawobject columns to `always_replicate`.

The replication status is the same for all replication definitions of a primary table. If you change the replication status for one replication definition with `alter replication definition`, you change the replication status for all replication definitions on the same primary table.

If you do not want to replicate some of the text, unitext, image, or rawobject columns in a marked table (or you marked the table using `sp_setreptable`, which sets the replication status of text, unitext, image, and rawobject columns to `do_not_replicate`), use `sp_setrepcol` to adjust the replication status. You can set the replication status for one or all columns to `always_replicate`, `do_not_replicate`, or `replicate_if_changed`. Table 9-2 describes each status.

**Table 9-2: text, unitext, image, or rawobject column replication status**

Status clause	Description
<code>always_replicate</code>	Adaptive Server logs replication information for the text, unitext, image, or rawobject column <i>whenever</i> any column in the row changes.

Status clause	Description
replicate_if_changed	Adaptive Server logs replication information for the text, unitext, image, or rawobject column <i>only</i> when the column data changes.
do_not_replicate	Adaptive Server does not log replication information for the text, unitext, image, or rawobject column.

To use `sp_setrepcol`, you must be the Database Owner or System Administrator for the data server.

---

**Note** If you have marked the database for replication to a standby database using `sp_reptostandby` and marked database tables for replication to a replicate database using `sp_setreptable`, Replication Server copies text, image, and rawobject columns to standby and replicate databases as `always_replicate`. If you want to copy text, unitext, image, and rawobject columns as `replicate_if_changed`, use `sp_setrepcol` to adjust the replication status. See “Replicating text, unitext, image, and rawobject data” on page 70 of the *Replication Server Administration Guide Volume 2*, for more information about replicating text, unitext, image, and rawobject columns in a warm standby application.

---

See `sp_setrepcol` in Chapter 5, “Adaptive Server Commands and System Procedures” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Enabling column replication

To mark a column with an image or rawobject datatype for replication, enter:

```
sp_setrepcol table, column, status
```

For example, to mark the `pic` column of datatype `image` for replication in the table `au_pix`, enter one of the following:

```
sp_setrepcol au_pix, pic, always_replicate
sp_setrepcol au_pix, pic, replicate_if_changed
sp_setreplcol au_pix, pic, replicate_if_changed,
use_index
```

## Disabling column replication

To turn off replication for the an image or rawobject column, enter:

```
sp_setrepcol table, column, do_not_replicate
```



For example, to disable replication of the pic column of datatype image for replication in the table au\_pix, enter:

```
sp_setrepcol au_pix, pic, do_not_replicate
```

## Enabling or disabling replication for all columns

To mark all text, unitext, image, and rawobject columns in the table with the same replication status, enter “null” instead of a column name. For example, to mark all text, unitext, image, and rawobject columns with the replicate\_if\_changed status:

```
sp_setrepcol table, null, replicate_if_changed
```

Execute sp\_setrepcol with the table name and a text, unitext, image, or rawobject column name to display the replication status of the specified column. For example:

```
sp_setrepcol table, column
```

Execute sp\_setrepcol with a table name to display the replication status of all of the text, unitext, image, and rawobject columns in the table. For example:

```
sp_setrepcol table
```

## Altering replication status for *text*, *unitext*, *image*, and *rawobject* columns

When you replicate text, unitext, image, and rawobject columns you must specify a compatible replication status for each column in both the replication definition and in Adaptive Server.

If you change the replication status of text, unitext, image, and rawobject columns in one table replication definition, the replication status automatically changes in all other replication definitions for the same table that includes those text, unitext, image, and rawobject columns.

## Changing from *replicate\_if\_changed* to *always\_replicate*

To change the replication status of a text, unitext, image, or rawobject column from replicate\_if\_changed to always\_replicate:

- 1 Let all transactions with a replicate\_if\_changed status finish processing.

- 2 Use `sp_setrepcol` to change the status of the column in Adaptive Server to `always_replicate`.
- 3 Use `alter replication definition` to change the status of the column to `always_replicate`.

### Changing from *always\_replicate* to *replicate\_if\_changed*

To change the replication status of a text, unitext, image, or rawobject column from `always_replicate` to `replicate_if_changed`:

- 1 Use `alter replication definition` to change the status of the column to `replicate_if_changed`.
- 2 Use `sp_setrepcol` to change the status of the column in Adaptive Server to `replicate_if_changed`.

### Resolving inconsistencies in replication status

The replication status for text, unitext, image, and rawobject columns in the Adaptive Server database is carried in the data modification commands that the RepAgent sends to the Replication Server. If the status is different at the Adaptive Server than in the replication definition, problems may result:

- *Scenario 1:* If a text, unitext, image, or rawobject column has a status of `replicate_if_changed` at the Adaptive Server database and `always_replicate` in the replication definition, Replication Server detects the inconsistency when the modification is being replicated, and RepAgent may shut down.
- *Scenario 2:* If a text, unitext, image, or rawobject column has a status of `do_not_replicate` at the Adaptive Server database and the replication definition includes that column for replication, processing continues and the Replication Server sends the modifications to the replicate database without the text, unitext, image, or rawobject data. The Replication Server also issues a warning message.

The following procedures enable you to resolve inconsistencies in the replication status of text, unitext, image, and rawobject columns for the two conflict scenarios described above and to resume replication operations.

#### Scenario 1

- Adaptive Server text, unitext, image, and rawobject status:  
`replicate_if_changed`

- Replication text, unitext, image, and rawobject definition status:  
always\_replicate

When RepAgent shuts down because a text, unitext, image, or rawobject column has a status of `replicate_if_changed` at the Adaptive Server database and `always_replicate` in the replication definition, take the following steps to resolve inconsistencies:

### **Setting `replicate_if_changed`**

To replicate text, unitext, image, or rawobject columns *only* when their values change:

- 1 Execute the alter replication definition command and change the status of the text, unitext, image, or rawobject columns to `replicate_if_changed`. Wait for the modified replication definition to arrive at the replicate sites.
- 2 Restart RepAgent.

### **Setting `always_replicate`**

To *always* replicate text, unitext, image, or rawobject columns:

- 1 Stop updates at the primary table.
- 2 Execute the alter replication definition command, and change the status of the text, unitext, image, or rawobject columns to `replicate_if_changed`. Wait for the modified replication definition to arrive at the replicate sites.
- 3 Restart RepAgent to let transactions with a `replicate_if_changed` status finish processing.
- 4 Execute the `sp_setrepol` system procedure at the Adaptive Server and change the status to `always_replicate`.
- 5 Execute alter replication definition and change the status of the text, unitext, image, or rawobject columns to `always_replicate`. Wait for the modified replication definition to be replicated to the replicate sites.
- 6 Resume updates to the primary table.

## **Scenario 2**

- Adaptive Server text, unitext, image, and rawobject status: `do_not_replicate`
- Replication definition text, unitext, image, and rawobject status:  
`always_replicate` or `replicate_if_changed`

When the Replication Server reports that the status of a text, unitext, image, or rawobject column is `do_not_replicate` at the Adaptive Server database and the replication definition includes that column for replication and specifies either `always_replicate` or `replicate_if_changed`, take the following steps to resolve inconsistencies.

### Setting `do_not_replicate`

If you *do not* want to replicate text, unitext, image, or rawobject columns:

- 1 Stop updates to the primary table.
- 2 Drop subscriptions to the replication definition.
- 3 Drop the replication definition.
- 4 Re-create the replication definition without the text, unitext, image, or rawobject columns, and re-create subscriptions.
- 5 Resume updates to the primary table.

### Setting `always_replicate` or `replicate_if_changed`

If you *do* want to replicate text, unitext, image, or rawobject columns:

- 1 Execute `sp_setrepcol` at the Adaptive Server database and change the status of the text, unitext, image, or rawobject column to `always_replicate` or `replicate_if_changed`. It should match the status in the replication definition.
- 2 Wait for subsequent transactions that modify the text, unitext, image, or rawobject column to be processed by the Replication Server.
- 3 Consider correcting any inconsistencies with the `rs_subcmp` program. See “Verifying subscription consistency” on page 391 for more information.

## Subscription issues for `replicate_if_changed` status

If you create subscriptions for replication definitions with text, unitext, image, or rawobject columns that have the status `replicate_if_changed`, see “Bulk materialization” on page 357 and “Materializing text, unitext, image, and rawobject data” on page 386.

## Function strings for replicating *text*, *unitext*, and *image* data

If you replicate columns with *text*, *unitext*, or *image* datatypes to a non-Adaptive Server database, you must create `rsdatarow_for_writetext`, `rs_get_textptr`, `rs_textptr_init`, and `rs_writetext` function strings for each *text*, *unitext*, or *image* column. The function string name must be the *text*, *unitext*, or *image* column name for the replication definition.

---

**Note** You cannot replicate *rawobject* or *rawobject* in row columns to non-Sybase databases unless you replicate these columns as their base datatype. The base datatype of *rawobject* is *image*; the base datatype of *rawobject* in row is *varbinary*.

---

Refer to Chapter 4, “Replication Server System Functions” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Replicating new large-object (LOB) datatypes

Replication Server supports the replication of Microsoft SQL Server 2005 datatypes *varchar(max)*, *nvarchar(max)*, and *varbinary(max)*. These datatypes can each store up to 2,147,483,647 bytes of data.

Replication Server introduces LOB datatypes as user-defined datatypes (UDDs) in the table-level replication environment. Replication Server also supports database-level replication for new LOB datatypes. The new LOB datatypes are directly mapped to *text*, *unitext*, and *image* datatypes.

The base type of UDDs is:

New LOB datatype	Base type
<i>varchar(max)</i>	<i>text</i>
<i>nvarchar(max)</i>	<i>unitext</i>
<i>varbinary(max)</i>	<i>image</i>

### Limitations

The new LOB datatypes have these limitations:

- You cannot define as a primary key a LOB column in the replication definition.
- You cannot define as searchable a LOB column in the replication definition.

- You cannot replicate stored procedures that include one of the new LOB datatypes as a parameter.
- You cannot use text pointers to manipulate the data of the new LOB datatypes.

In a mixed-version environment, the primary and replicate Replication Server must have a site version of 15.1 and an LTL version of 710.

For more information about the new LOB datatypes, see the *Replication Server Reference Manual*.

#### Partial update of LOB datatypes

Partial-update transaction directly writes a character string at a user-defined position of a table column without issuing a delete and replace command, as would happen in a full update.

Use the new `rs_updatetext` LTL command to implement partial update:

```
{distribute|_ds} command_tags {applied|_ap} 'table'.rs_updatetext
{partialupd|_pu} [{first|_fi}] [last] [{changed|_ch}] [with log]
[{{withoutp|_wo}}] [{{offset|_os}}=offset {deletelen|_dln}=deletelength]
[{{textlen|_tl}}=length] text_image_column
```

Partial update does not support multiple character set conversion. Its support is restricted to Microsoft SQL Server 2005.

For more information about partial update, see the *Replication Server Design Guide*.

## Replicating computed columns

Computed columns allow you to create an expression and place the result of the expression in a table column. A computed column is:

- **Materialized** — when its value is computed for each insert or update. A materialized computed column is stored in the same way as regular columns.
- **Virtual** — when its value is computed only when referenced in a query. A virtual computed column is stored in the table or index page.

A computed column expression can be:

- **Deterministic** — when its value is the same each time it is evaluated.
- **Nondeterministic** — when its value may be different each time it is evaluated (for example, a date stamp).

See the *Adaptive Server Enterprise System Administration Guide* for more information about creating and managing computed columns.

Replication Server replicates materialized computed columns in DML statements in the same way it replicates other columns; it does not replicate virtual computed columns.

The replication of computed columns is supported by function strings. With Replication Server version 15.0 and later, the class-level function string `rs_set_dml_on_computed` is applied at the replicate database DSI when a connection is established. It issues `set dml_on_computed "on"` after the `use database` statement. If the replicate Adaptive Server is 12.5.x or earlier, the command is ignored.

When you are creating or altering replication definitions for tables containing:

- Deterministic columns – you can choose whether or not to include those columns in the replication definition. Because deterministic columns always realize the same value, you can create the replication definition without them and allow each replicated insert and update to compute values at the replicate database.
- Nondeterministic columns – you must include nondeterministic computed columns in the replication definition to ensure that the primary and replicate databases remain synchronized.

## Replicating encrypted columns

As of version 15.0, Replication Server supports replication of encrypted columns in Adaptive Server. Replication Server replicates the encrypted columns from the primary Adaptive Server database, in binary format as ciphertext values, rather than clear text values.

The encryption keys for the primary and the replicate databases must be identical. Use replication to create the encryption key at the replicate database, or use a dump and load command to ensure that the encryption keys are identical.

Replication Server in a warm standby and in an MSA environment replicates the create, alter, and drop commands of the encryption keys. It also replicates alter table to encrypt or decrypt a column. To replicate the create, alter, and drop encryption key DDL commands, the `system_encr_passwd` must be identical for both the primary and the replicate databases.

If the encryption keys are stored in a separate database, ensure that it is synchronized at the same time as the database containing the encrypted columns using those encryption keys.

If data has diverged between the primary and the replicate databases because of earlier encryption keys or because of differences between the initialization vector and the padding, manually sync the data to avoid failures of update and delete statements.

Restrictions

Replicating encrypted columns has these restrictions:

- Text and image columns cannot be encrypted.
- Encrypted columns cannot be used in a where clause because Replication Server receives the value in ciphertext and cannot compare that value to a clear text value. The encrypted columns cannot be searchable columns.
- If an encrypted column is used in a primary key, the encryption key must be defined with INIT\_VECTOR NULL and PAD NULL.

The purpose of an initialization vector and padding is to randomize the ciphertext so that two like values encrypted by the same key result in two differing ciphertext strings. If the ciphertext for encrypted data at the primary and the replicate sites differ, then any attempt by the Replication Server to match the before-image from the primary site with the data at the replicate site fails.

If no initialization vector is used, the ciphertext at the source and the target databases exactly match. The matching is required because Replication Server issues a where clause on the update/delete statements using the ciphertext of the encrypted columns.

- If a table replication definition is not used to replicate the data in a warm standby or MSA environment for a table, all the encrypted columns in that table must be encrypted with keys defined as INIT\_VECTOR NULL and PAD NULL.

---

**Note** rs\_subcmp supports replication of encrypted columns in Adaptive Server.

---



## Working with special datatypes

This section describes how to use the special datatype `rs_address`, `identity`, and `timestamp` columns in replication definitions.

### Using the `rs_address` datatype

The `rs_address` special datatype makes a unique subscription resolution technique possible: bitmaps of the `rs_address` datatype (based on the underlying `int` datatype) are compared with a bitmask in a subscription's `where` clause to determine whether a row should be replicated.

To use this subscription resolution method:

- 1 Create tables that use columns of the `int` datatype.
- 2 Create a replication definition that includes these columns in the column list, but declare the datatype as `rs_address` instead of `int`.

You must include any columns that use the `rs_address` datatype in the `searchable columns` clause of the replication definition in order to perform bitmap comparison using the `where` clause.

See “Bitmap subscriptions” on page 388 for more information on using `rs_address` columns for bitmap subscription resolution.

### Replicating *identity* columns

`identity` columns store sequential numbers (such as invoice numbers, employee numbers, or record numbers) that are generated automatically. The value of the `identity` column uniquely identifies each row in a table.

`identity` columns use the numeric underlying datatype with scale 0, between 1 and  $10^{38} - 1$ , inclusive.

`identity` columns are never updated by the `update` command. `update` applied to primary data from a replicate site (using a request function) can never update the `identity` column with `identity` data.

## Specifying an *identity* column in a replication definition

To create a replication definition for a table that contains an identity column, specify “identity” as the declared datatype for the column or use a column-level translation to specify “identity” as the published datatype for the column.

A replication definition, or multiple replication definitions for the same table, may not publish more than one column that has the datatype identity.

Note that if one replication definition publishes a column as “identity,” another replication definition may publish the column as numeric and avoid having the extra commands sent with an insert for subscribers to the second replication definition.

## How Replication Server replicates *identity* columns

Replication Server applies the following command to the replicate table before insert:

```
set identity_insert table_name on
```

Replication Server applies the following command to the replicate table after insert:

```
set identity_insert table_name off
```

For any table containing an identity column, the maintenance user must be the owner of the table (or must be the “dbo” user or aliased to the “dbo” login name) at the replicate database in order to use the Transact-SQL `identity_insert` option.

## Replicating *timestamp* columns

Replication Server adds `timestamp` as a Replication Server datatype. `timestamp` is defined as `varbinary(8)` with a status bit indicator that differentiates it from `varbinary`. This allows the replication of `timestamp` columns to replicate, standby, and MSA databases.

You can also define `timestamp` as a primary key in a replication definition, and a searchable column in a replication definition and a function replication definition.

The `send_timestamp_to_standby` configuration parameter is also added to support `timestamp` replication. When `send_timestamp_to_standby` is enabled and there are no replication definitions, `timestamp` columns are sent to the replicate database.

For any table containing timestamp column, the maintenance user must be the owner of the table or must be the “dbo” user of aliased to the “dbo” login name at the replicate database.

### Specifying a *timestamp* column in a replication definition

To create a replication definition for a table that contains a timestamp column, specify “timestamp” as the declared datatype for the column or use a column-level translation to specify “timestamp” as the published datatype for the column.

A replication definition, or multiple replication definitions for the same table, may not publish more than one column that has the datatype timestamp.

---

**Note** The replicate Adaptive Server must be version 15.0.2 or later to support timestamp in replication definition.

---

See the *Replication Server Reference Manual* for more information about the new timestamp datatype.

## Modifying replication definitions

This section provides information on viewing, altering, and dropping replication definitions. It also describes how Replication Server supports table changes resulting from the alter table command when the table:

- Has subscriptions from a replicate site, or
- Is replicated to the standby database using a replication definition with a send standby replication definition columns clause.

---

**Note** Adaptive Server Enterprise version 12.0 allows users to alter existing tables— add non-nullable columns, drop columns, and modify column datatypes.

---

See “alter table support for warm standby” on page 111 in the *Replication Server Administration Guide Volume 2* for more information about alter table changes that affect warm standby tables with no subscriptions.

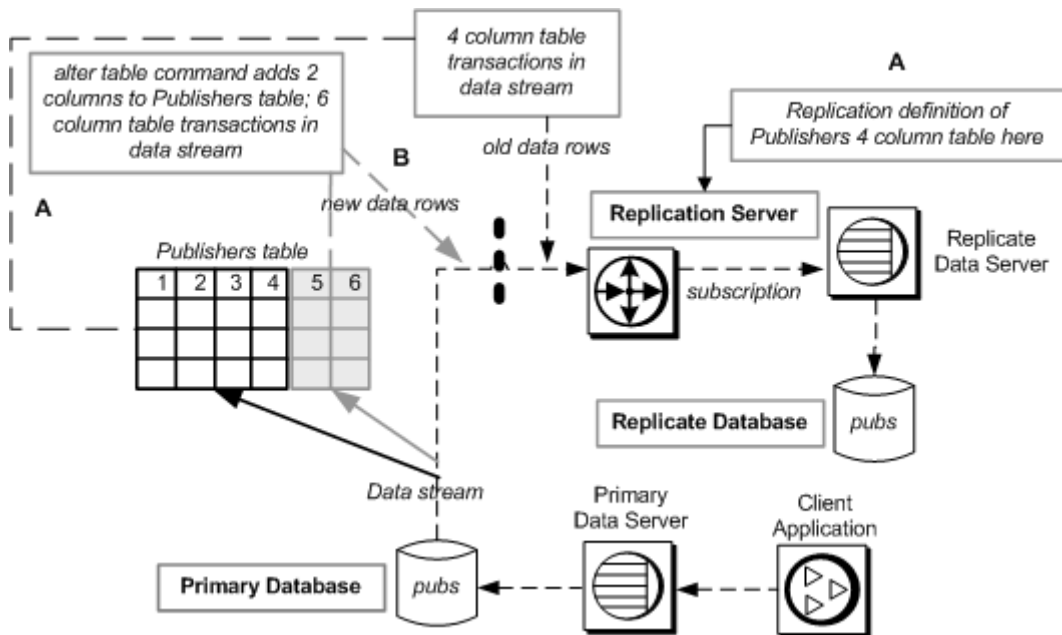
For descriptions of procedures that require altering or dropping replication data, see “Modifying replicated data” on page 305.

## Maintaining table schema

Replication Server stores information about table schema in a table’s replication definition. During alter table operations, new or modified data rows may reach Replication Server while old data rows are still being processed farther down the data stream. When replication definitions exist for a table, the discrepancies between the columns in the table and the columns in the replication definition may cause Replication Server threads (executor, distributor, or DSI) to shutdown.

**Note** See the section on alter table in the *Adaptive Server Enterprise Reference Manual* (version 12.0 or later) for syntax and details on how alter table works in Adaptive Server.

**Figure 9-2: Replicate Table Schema Inconsistency**



As Figure 9-2 illustrates, because the replication definition cannot describe the old data rows and the new data rows (Figure 9-2, A & B) at the same time, discrepancies between a replication definition and its corresponding table may cause Replication Server to behave incorrectly; that is, not able to read or write data rows to inbound and outbound queues.

For example:

If Replication Server receives the following from RepAgent:

```
old_datarow1
old_datarow2
...
alter table command
new_datarow1
new_datarow2
...
```

both the old and new data rows need to be replicated with the correct number of columns and the correct column datatype.

If alter table drops columns, old data rows still have these columns replicated while the new data rows do not.

If alter table adds new columns, the new columns need to be included only in the new data rows. Figure 9-2 illustrates that when you add new columns to the Publishers table using alter table (“B”), because the new rows are not in the table’s replication definition, the new rows will not be replicated, causing you to lose data.

If alter table alters a column datatype, both the old and new data rows need to be replicated in their own column types. When you modify primary table column datatypes, there is also a period of time when the replication definition column datatype does not match the table column datatype. This mismatch may cause problems in Replication Server when column datatypes are used.

## Migration procedure

The only way to guarantee consistency between tables and replication definitions and ensure that replication works correctly, is to use the steps in the following procedure when you want to add or modify columns in a primary table within a replication system.

---

**Note** This procedure is required only when a table has subscriptions, or when send standby replication definition columns is used to replicate to a standby database.

---

❖ **Altering a primary table that is part of a replication system and avoid data inconsistency or Replication Server thread shutdown:**

- 1 Stop all primary database activity.
- 2 Send an intentional fake “update” command. When this transaction’s results appear at the replicate site, you know that all operations have been completed by the Replication Server.
- 3 Use the Transact-SQL dump transaction command to make a copy of the primary database’s transaction log (syslog) and remove the inactive portion.

See the *Adaptive Server Enterprise Reference Manual* for instructions.

- 4 Quiesce the replication system. Make sure that the last update (step 2) has reached the replicate.

See the “Quiescing a replication system” on page 104 for instructions.

- 5 Use alter table to change the primary table schema.

See the alter table command in the *Adaptive Server Enterprise Reference Manual* (version 12.0 or later) for instructions.

- 6 Use alter replication definition to change the corresponding replication definitions. Verify that the replication definition changes reach all destination RSSDs.

---

**Note** If the alter table changes involve columns that are used in a subscription or article where clause, drop the subscription (without purge) or article before you alter the replication definition. If you use alter table to drop columns that are not used in a where clause, replication definition changes are not necessary.

---

See “Altering replication definitions” on page 298 for details.

- 7 If you dropped subscriptions in step 6, recreate them using create subscription without materialization or define subscription.  
  
See create subscription and define subscription in Chapter 3 of the *Replication Server Reference Manual* (version 11.5 or later) for instructions.
- 8 Change the replicated table schema if necessary.
- 9 Resume activity in the primary database.

### **alter table support for replicate databases**

Prior to version 12.0, when Replication Server received a data row, the columns that were defined in the replication definition but missing in the data row were sent as null. This could cause the Data Server Interface (DSI) to shutdown when the columns were dropped from the replicate or standby table.

Beginning with version 12.0, to support alter table [add | drop | modify] column, Replication Server sends only the values that are received from the data server. Columns that are defined in the replication definition but missing in the data row are ignored instead of receiving a null value. The exception is when you use custom function-strings. If a missing column value is expected in a custom function-string, then Replication Server will continue to send null for the column.

If you use a column in a subscription or article where clause, you must drop the subscription or article before you can change or drop that column. If you do not use a column in a subscription or article where clause, then you do not need to drop the subscription.

---

**Warning!** You always need to follow the manual procedure specified in “Migration procedure” on page 296 for alter table [add | drop | modify] columns when subscriptions are involved, or when you use send standby replication definition columns to replicate to a standby database. Failure to do so may cause data loss or Replication Server thread problems.

---

---

**Note** See “alter table support for warm standby” on page 111 in the *Replication Server Administration Guide Volume 2* for information on how Replication Server support Adaptive Server version 12.0 enhancements to alter table.

---

## Recovery procedures

This section discusses the recovery procedures to use if data loss or problems occur as a result of alter table changes.

Recovering from executor thread problems

Executor thread problems require no extra recovery. Data may be discarded when there is a normalization error in an executor thread.

When datatype conversion has completed, and if the table has been altered such that the executor thread cannot normalize the data, the data row may be discarded. Use the “Migration procedure” on page 296 to avoid data loss.

Recovering from inbound queue problems

If data in the inbound queue is incompatible with the column datatype in a replication definition, the distributor thread may shut down. The resume distributor command has been extended to allow you to skip one transaction:

```
resume distributor ds.db
skip transaction
```

Recovering from outbound queue problems

When there is bad data in the outbound queue, use `resume connection skip [n] transaction` to skip the bad data. In the case of replicate (not standby) Data Server Interface (DSI) threads, you may be able to alter the replication definition and resume the DSI to recover the data.

## Viewing existing replication definitions

To display information about existing replication definitions, use the Adaptive Server procedures `rs_helpuser` and `rs_helptable`. See `rs_helprep` and `rs_helpuser` in Chapter 6, “Adaptive Server Stored Procedures” in the *Replication Server Reference Manual* for complete information about this command.

## Altering replication definitions

You may need to alter a replication definition after a column has been added to a primary table or if a destination database requires a column that was not specified in the original replication definition.



In most instances, you alter replication definitions in conjunction with changing database schema in the source or destination table. Be sure to coordinate schema changes between the source and destination sites. See “Modifying replicated data” on page 305.

---

**Warning!** When you alter a replication definition, it may take a while for the changes to reflect in the replicate RSSD. If you manipulate the data too soon before or after running alter replication definition command, Replication Server may use the wrong replication definition to process the data.

---

This section describes how to use alter replication definition to modify a replication definition. You can alter the replication definition in one of the following ways:

- Provide a different replicate table name
- Add columns to the columns list
- Provide different replicate column names
- Change the specifications for replicating text, unitext, image, or rawobject columns
- Add columns to the primary keys column list
- Remove columns from the primary keys column list
- Add columns to the searchable columns list
- Drop columns from the searchable columns list
- Change declared or published column datatypes
- Change the specification for replicating minimal columns
- Change how the replication definition is used in replicating to a standby database

---

**Note** Function strings with replication definition scope are not automatically altered when you add columns to a table or to a replication definition. In certain cases, you must alter the function strings manually. See “Managing function strings” on page 32 in the *Replication Server Administration Guide Volume 2*.

---

Use alter replication definition at the primary Replication Server where you created the replication definition. See “Creating replication definitions” on page 250 for more information about what you can include in a replication definition.

- To rename primary or replicate tables, drop and re-create the replication definition. See “Renaming replicated tables” on page 305 for more information about how to accomplish this task.
- To drop or rename primary columns or change column datatypes, drop and re-create all the replication definitions that have the primary columns. See “Deleting columns in a source or destination table” on page 306 for more information about how to accomplish this task.

Refer to Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information about alter replication definition command.

Examples follow for different scenarios of altering replication definitions using alter replication definition.

Normally, you should quiesce the system and shut down the RepAgent before altering a replication definition.

## Providing a different replicate table name

To replicate data from a source table into a destination table with a different name, alter the replication definition. For example:

```
alter replication definition publishers
with replicate table named publishers2
```

## Changing the specified columns

Following are examples of how to add or change a column for the primary and destination tables.

### Adding a column

To add a char column named zip (for zip code information) to the source and destination copies of the publishers table:

- 1 Use the Transact-SQL alter table command to add the column to the tables in Adaptive Server. See the *Adaptive Server Enterprise Reference Manual* for more information.
- 2 Use alter replication definition to add the same column to the publishers\_rep:

```
alter replication definition publishers_rep
add zip char(10)
```

- 3 If the column you added to the destination table has a different name than the source column, enter a command like this:

```
alter replication definition publishers_rep
add zip as rep_zip char(10)
```

See “Adding columns in source and destination tables” on page 306 and alter replication definition in Chapter 3, “Replication Server Commands” *Replication Server Reference Manual* for more information.

### Dropping a searchable column

You can drop searchable columns from a replication definition only if they are not used in subscription or article where clause.

- 1 Use drop subscription to remove any subscriptions in which you want the where clause to exclude the searchable columns you are dropping. See “Using the drop subscription command” on page 380.
- 2 Use alter replication definition to drop the searchable column. For example:

```
alter replication definition publishers_rep
drop searchable columns zip
```

(This example removes the zip searchable column from the publishers\_rep replication definition.)

See alter replication definition in Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information.

- 3 Use create subscription to re-create subscriptions to the altered replication definition. See “Using the create subscription command” on page 374.

### Adding or dropping primary keys

Replication Server depends on primary keys to find the correct rows at the replicate or standby table. To add a the column zip as a primary key to the replication definition, enter:

```
alter replication definition publishers_rep
add primary key zip
```

To drop a primary key, enter:

```
alter replication definition publishers_rep
drop primary key zip
```

To replace all primary key columns, first alter the corresponding replication definition to add the new primary keys, then drop the old primary key columns in the table.

---

**Warning!** If all primary key columns are missing from the primary table, the DSI will shut down.

---

### Altering column datatypes

- You cannot change the declared column datatype (the datatype in the primary table) if it is used in a subscription or article where clause.
- You cannot change the `rs_address` datatype.
- You can change the column datatype to `text`, `unitext`, `image`, `rawobject`, or `rawobject in row` only if it is not a primary key or searchable column.
- To change the published (replicate) datatype, you must include the declared (primary) datatype of a column (whether it is being changed or not) and the `[map to]` clause.
- Altering a column's datatype and nullability affects the same column across all replication definitions for a table.

However, changes between a `rawobject` or `rawobject in row` and its base datatype, affects only the current replication definition. See “Translating datatypes using HDS” on page 317 for more information about HDS.

- Use column nullability changes for `text`, `unitext`, `image`, and `rawobject` columns only.

### Providing a different replicate column name

To replicate data for the source column `zip` into a destination column named `rep_zip2`, enter:

```
alter replication definition publishers_rep
alter columns with zip as rep_zip
```

Enter such a command when:

- You alter the existing destination table to add column `rep_zip`.
- You drop and re-create the destination table to contain the column `rep_zip` in place of the original column `zip`.

## Changing *text*, *unitext*, *image*, and *rawobject* replication status

To change the replication status of *text*, *unitext*, *image*, and *rawobject* columns in a replication definition, use `alter replication definition`.

See “Altering replication status for *text*, *unitext*, *image*, and *rawobject* columns” on page 283 for more information.

## Adding a searchable column

A searchable column lets you create subscriptions based on values in the column.

To add and take advantage of a searchable column:

- 1 Use `drop subscription` to remove any subscriptions in which you want the `where` clause to include the added searchable column. See “Using the `drop subscription` command” on page 380.

- 2 Use `alter replication definition` to add the searchable column. For example:

```
alter replication definition publishers_rep
add searchable columns zip
```

(This example makes the `zip` column searchable.)

See `alter replication definition` in Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information.

- 3 Use `create subscription` to re-create subscriptions to the altered replication definition. See “Using the `create subscription` command” on page 374.

See “Changing searchable columns” on page 307 for more information.

## Changing minimal column replication

To specify that Replication Server use the minimal number of columns (as opposed to all columns in each row) when it copies update and delete operations, enter a command like this:

```
alter replication definition publishers_rep
replicate minimal columns
```

## Altering a replication definition for warm standby replication

To change whether a replication definition will be used to replicate data into a standby database in a warm standby application, use `alter replication definition`. See `alter replication definition` in Chapter 3, “Replication Server Commands” of the *Replication Server Reference Manual*.

You can specify which replication definition to use to replicate data into a standby database in a warm standby application. You can also specify whether to replicate all the columns in the table or only the replication definition’s columns.

See “Using replication definitions with warm standby applications” on page 260 for more information.

## Dropping replication definitions

Before you drop a replication definition, first drop all subscriptions and articles that reference that replication definition. See Chapter 11, “Managing Subscriptions” for details on dropping subscriptions. See “Dropping articles” on page 316 for details on dropping articles.

To access a list of existing subscriptions for a specified replication definition, use `rs_helpsub`. See `rs_helpsub` in Chapter 6, “Adaptive Server Stored Procedures” in the *Replication Server Reference Manual* for more information.

To access a list of existing subscriptions for all replication definitions, use `rs_helprep`. See `rs_helprep` in Chapter 6, “Adaptive Server Stored Procedures” in the *Replication Server Reference Manual* for more information.

Enter `drop replication definition` at the primary Replication Server. For example, to drop the `publishers_rep` replication definition, enter a command like this:

```
drop replication definition publishers_rep
```

Refer to Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information about `drop replication definition` command.

## Modifying replicated data

This section describes how to modify replicated data and perform related operations to maintain replication. Chapter 11, “Managing Subscriptions” for subscription-specific commands.

Before you modify replicated data, carefully review the issues raised in “Planning a replication system” on page 243. When attempting to modify replicated data, refer to this section for any dependencies that may exist, and for the sequence of tasks required to perform the procedure.

## Adding a new table

To add a new source table, or add a new destination copy for an existing source table, follow the procedure outlined in “Replication procedure” on page 246.

## Renaming replicated tables

To rename a replicated table:

- 1 In Adaptive Server, use `sp_setreplicate` to disable replication for the table. See `sp_setreplicate` in Chapter 5, “Adaptive Server Commands and System Procedures” in the *Replication Server Reference Manual*.
- 2 Use `drop subscription` to drop subscriptions to all of the table’s replication definitions. See “Using the drop subscription command” on page 380.
- 3 Use `drop replication definition` to drop all of the table’s replication definitions. See “Dropping replication definitions” on page 304.
- 4 Rename the destination table.

Follow the steps in the “Replication procedure” on page 246. Be sure to specify the table names correctly, as described under “Specifying the replication definition name and table names” on page 252.

## Dropping a replicated table

To drop a replicated table, follow these steps:

- 1 Use the Transact-SQL `drop table` command to drop the table at the primary database.

See the *Adaptive Server Enterprise Transact-SQL User's Guide* for drop table syntax.

- 2 Use drop subscription to drop the subscriptions to the table. See “Using the drop subscription command” on page 380.
- 3 Use check subscription to confirm that the subscriptions are dropped. See “Using the check subscription command” on page 379.
- 4 Use drop replication definition to drop the replication definition to the table at the primary Replication Server. See “Dropping replication definitions” on page 304.
- 5 Use `rs_helprep` to confirm that the replication definition is dropped at all Replication Servers in the replication system. See “Viewing existing replication definitions” on page 298.

## Adding columns in source and destination tables

To add columns to source and destination tables in a warm standby only setup, follow the instructions in “alter table support for warm standby” on page 111 in the *Replication Server Administration Guide Volume 2*.

To add columns to source and destination tables that are replicated through subscriptions, use the “Migration procedure” on page 296.

## Deleting columns in a source or destination table

To delete columns in source and destination tables in a warm standby only setup, follow the instructions in “alter table support for warm standby” on page 111 in the *Replication Server Administration Guide Volume 2*.

To delete columns in source and destination tables that are replicated through subscriptions, use the “Migration procedure” on page 296.

---

**Note** If you drop columns from a table, it is unnecessary to drop those columns from replication definitions unless they are used in a subscription or article where clause. However, the columns need to be dropped from the searchable columns list and the primary key list.

If you drop table columns that are used in a subscription or article where clause, you need to drop the subscription or article, then recreate it.

---



## Changing searchable columns

To add searchable columns to a replication definition, see “Adding a searchable column” on page 303.

## Dropping a searchable column

You can drop searchable columns from a replication definition only if they are not used in subscription or article where clause.

- 1 Use drop subscription to remove any subscriptions in which you want the where clause to exclude the searchable columns you are dropping. See “Using the drop subscription command” on page 380.
- 2 Use alter replication definition to drop the searchable column. For example:

```
alter replication definition publishers_rep
drop searchable columns zip
```

(This example removes the zip searchable column from the publishers\_rep replication definition.)

See “alter replication definition“ in Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information.

- 3 Use create subscription to re-create subscriptions to the altered replication definition. See “Using the create subscription command” on page 374.

## Changing column datatypes in a source or destination table

To change column datatypes in a primary and replicate table in a warm standby only setup, follow the instructions in “alter table support for warm standby” on page 111 in the *Replication Server Administration Guide Volume 2*.

To change column datatypes in source and destination tables that are replicated through subscriptions, use the “Migration procedure” on page 296.

## Using publications

A publication lets you collect replication definitions for the same or related tables and/or stored procedures and then subscribe to them as a group. You collect replication definitions in a **publication** at the source Replication Server and subscribe to them with a **publication subscription** at the destination Replication Server.

With publications, you monitor the status of one publication subscription for a set of tables and procedures.

The following steps summarize the procedure for replicating data using publications.

- 1 Create or select the replication definitions to include in the publication.
- 2 Create the publication.
- 3 Create articles that reference the replication definitions you have chosen.
- 4 Validate the publication.
- 5 Create a subscription for the publication.

---

**Note** A replicate database can subscribe to different replication definitions of the same primary table directly or through publications—as long as each replication definition references a different table in the replicate database.

---

To use publications, the primary Replication Server must be version 11.5 or later. To use publication subscriptions, the replicate Replication Server and the route from the primary Replication Server and the replicate Replication Server must be version 11.5 or later.

When you use publications, you create and manage the following objects:

- *Articles* – replication definition extensions for tables or stored procedures that let you put table or function replication definitions in a publication. Articles may or may not contain where clauses, which specify a subset of rows that the replicate database receives.
- *Publications* – groups of articles from the same primary database.
- *Publication subscriptions* – subscriptions to a publication. When you create a publication subscription, Replication Server creates a subscription for each of the publication's articles. Publication subscriptions do not contain where clauses.

In general, you manage publications and publication subscriptions in the same way as you do replication definitions and subscriptions. However, when you create a publication, you can specify the subset of rows that the replicate table receives by including a where clause in the article—not in the subscription.

You can create and manage publications using the command line. The following sections provide detailed instructions for creating publications at the command line.

Refer to Chapter 10, “Managing Replicated Functions” for more information about publications for stored procedures. Refer to Chapter 11, “Managing Subscriptions” for information about creating and managing publication subscriptions.

## Using publications to replicate data at the command line

This section describes how to create a publication at the command line and prepare it for subscription. It also contains information on modifying dropping a publication and its associated articles and replication definitions.

### Commands for creating and managing publications

Table 9-3 lists the RCL commands for working with publications. All of these commands, except check publication, are executed at the source Replication Server, where they require create object permission. Anyone can execute check publication at the source Replication Server—or at the destination Replication Server if the user has the same login and password at both servers.

Table 11-5 on page 396 lists the RCL commands for working with publication subscriptions.

**Table 9-3: Commands for managing publications**

Command	Task
create publication	Creates a publication for a group of tables or stored procedures that is to be replicated to one or more subscribing databases.
create article	Creates an article for a publication, allowing you to add one or more where clauses to specify a subset of rows to send to the destination database. The publication and the replication definition on which the article is based must exist before you create an article.
validate publication	Checks that the publication contains at least one article and marks the publication as VALID and ready for subscription.
check publication	Displays the status of the publication and the number of articles it contains.

Command	Task
drop publication	Removes the publication from the rs_publications system table. You can drop the replication definitions associated with the publication if they are <i>not</i> included in other publications or subscriptions.
drop article	Removes the article from the publication and from the rs_articles system table. You can drop the replication definition associated with the article if it is <i>not</i> included in other articles or subscriptions.
rs_helppubs	Displays information about publications and articles.

## Creating publications and articles at the command line

The following procedure describes the RCL procedure for preparing a publication for subscription and creating a subscription against it.

At the source  
Replication Server

- 1 Create or select replication definitions for the tables or stored procedures from which you want to copy data.

The replication definition specifies the source and destination tables or stored procedure and the columns or parameters that are sent to the subscribing database. Refer to “Creating replication definitions” on page 250 for details.

- 2 Use create publication to create the publication that groups the replication definitions.

Publication information is stored in the rs\_publications system table in the source Replication Server RSSD. It includes the name of the publication, data server, and database. The publication name must be unique for the source Replication Server and database.

The following example creates a publication named pubs2\_pub. The primary database is pubs2 managed by the TOKYO\_DS data server.

```
create publication pubs2_pub
with primary at TOKYO_DS.pubs2
```

Publication information is not copied to the destination Replication Server until you create a subscription against the publication at the destination Replication Server.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

- 3 Use create article to create articles for the publication.

Each article specifies the publication to which it belongs and the table or function replication definition with which it identifies. A publication can contain articles based on the same or different replication definitions. The replication definition and publication must exist when you create the article.

An article includes the names of the publication, the replication definition, and the source data server and database. Article information is stored in the `rs_articles` and `rs_whereclauses` system tables. Each article name must be unique within the publication.

The following example creates an article named `titles_art` based on the replication definition `titles_rep` for the publication `pubs2_pub`.

```
create article titles_art
  for pubs2_pub with primary at TOKYO_DS.pubs2
  with replication definition titles_rep
```

An article can include where clauses that specify the rows or parameters to be sent to subscribing databases. Refer to “Specifying a where clause with the create article command” on page 312 for more information.

Creating an article invalidates the publication, which makes it ineligible for subscription. After you create an article, you must change the status of the publication to `VALID`, using `validate publication`, before you can create subscriptions against it.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

- 4 Use `validate publication` to change the status of the publication to `VALID`.

When you validate a publication, Replication Server checks that the publication contains at least one article and marks the publication ready for subscription.

Whenever you add or drop an article from a publication, Replication Server invalidates the publication. To mark the publication `VALID`—and ready for subscription—you must execute `validate publication`.

After you validate a publication, you can create a publication subscription against it.

The following example validates the `pubs2_pub` publication. The source database is `pubs2` managed by the `TOKYO_DS` data server.

```
validate publication pubs2_pub
  with primary at TOKYO_DS.pubs2
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

At the destination  
Replication Server

Use `create subscription` to create the publication subscription.

When you create a publication subscription, Replication Server creates a subscription for each article in the publication.

See “Using publication subscriptions” on page 395 for information about creating and managing publication subscriptions.

### Specifying a *where* clause with the *create article* command

You can include one or more *where* clauses in an article. A *where* clause sets criteria for the column or parameter values that are to be replicated. If you omit the *where* clause, Replication Server copies all rows for columns specified in the table replication definition or all parameters specified in the function replication definition.

The *where* clause syntax for articles is:

```
[where (column_name | @param_name)
  {< | > >= | <= | = | &} value
 [and {column_name | @param_name}
  {< | > >= | <= | = | &} value]...]
[or where (column_name | @param_name)
  {< | > >= | <= | = | &} value
 [and {column_name | @param_name}
  {< | > >= | <= | = | &} value]...]
...]
```

Each column name in a *where* clause must be listed in the searchable columns list of the table replication definition. The value for each column must have the same datatype as the column to which it is compared.

---

**Note** Each *where* clause in an article is joined by the `or` operator. However, the `!=`, `!<`, `!>`, and `or` operators are not supported *inside* a *where* clause. The `&` operator is supported only on `rs_address` columns. For details on using the `rs_address` datatype, see “Using the `rs_address` datatype” on page 291 and “Bitmap subscriptions” on page 388.

---

The following example creates an article named `titles_art` for the publication named `pubs2_pub`, using a *where* clause that limits replication to rows where the value in the `type` column is ‘`popular_comp`.’

```
create article titles_art
```

```
for pubs2_pub with primary at TOKYO_DS.pubs2
with replication definition titles_rep
where type = 'popular_comp'
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Viewing publication information

You can view information about publications and articles with the `check publication` command and the `rs_helppub` stored procedure.

### Display publication status and number of articles

To display the number of articles in a publication and its current status, use `check publication`.

Any user can execute `check publication` at either the primary or replicate Replication Server. If you execute `check publication` at the replicate Replication Server, you must have the same login and password at the primary and replicate servers.

The following example displays the status and number of articles in the `pubs2_pub` publication.

```
check publication pubs2_pub
with primary at TOKYO_DS.pubs2
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines and sample output.

### Display publication and article information

To display information about a publication and its articles, use the `rs_helppub` stored procedure at either the primary or replicate Replication Server RSSD.

---

**Note** Although you can execute `rs_helppub` at the primary or replicate site, `rs_helppub` only displays publication information stored at the site at which it is executed. For example, if you execute `rs_helppub` at the primary site, `rs_helppub` displays information about all publications created at that site. If, however, you execute `rs_helppub` at the replicate site, `rs_helppub` only displays information about publications for which subscriptions have been created at that site.

---

Here are some examples of using `rs_helppub`:

- To list all publications at a site, enter:

```
rs_helppub
```

The display output includes publication name, status, the primary Replication Server and database names, the number of articles, and the date of the latest change to the publication.

- To display detailed information about a particular publication, enter:

```
rs_helppub publication_name, primary_dataserver,  
primary_db
```

The display output includes the above information and the names of associated articles, replication definitions, and primary and replicate tables. If subscriptions have been created for the publication, the display includes names of the subscriptions, replicate databases, owners, and the date of the latest change to the subscription.

- To display information about a particular article, enter:

```
rs_helppub publication_name, primary_dataserver,  
primary_db, article_name
```

The output display includes the name of the publication to which the article belongs, associated replication definitions, status information, and where clauses and subscriptions, if any.

Refer to the *Replication Server Reference Manual* for complete syntax and usage guidelines and sample output.

## Altering publication information

Normally, if you want to alter an article or publication, you must drop the article or publication and re-create it.

If you want to make the where clauses in an article more selective, you can:

- Drop the article and re-create it with altered where clauses, or
- Create another article (for the same replication definition), tailoring the where clauses to the new row or parameter selection.

Refer to “Dropping publications” on page 315 and “Dropping articles” on page 316.



## Adding articles to a publication

To add articles to an existing publication, follow these steps:

At the source  
Replication Server

- 1 Create or select the replication definitions on which the articles are to be based.
- 2 Use `create article` to create new articles.
- 3 Use `validate publication` to validate the publication so that subscriptions can be created for the new articles.

At the destination  
Replication Server

To create subscriptions for the new articles, enter `create subscription` or `define subscription` and include the `for new articles` clause. Refer to “Using publication subscriptions” on page 395 for more information.

## Dropping publications

Use `drop publication` to remove a publication and all of its articles from the system tables.

Before you drop a publication, you must, at the replicate Replication Server, drop all subscriptions created against it. See “Dropping subscriptions for publications and articles” on page 401.

Execute `drop publication` at the Replication Server that manages the source database. You must have `create object` permission.

The following example drops the `pubs2_pub` publication and the articles it contains.

```
drop publication pubs2_pub
with primary at TOKYO_DS.pubs2
```

Publication information is dropped immediately from the primary Replication Server; it is not dropped from the replicate Replication Server until:

- You attempt to create a subscription against the dropped publication, or
- You enter `check publication` at the replicate Replication Server.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Dropping associated replication definitions

To drop replication definitions associated with the publication, include the `drop_repdef` clause when you execute `drop publication`. Replication Server drops all replication definitions associated with the publication that are not referenced by other publications or subscriptions.

For example, to drop all replication definitions associated with `pubs2_pub`, enter:

```
drop publication pubs2_pub
  with primary at TOKYO_DS.pubs2
  drop_repdef
```

## Dropping articles

Use `drop article` to remove an article from a publication.

Before you drop an article, you must drop subscriptions created against it at the replicate Replication Server. See “Dropping subscriptions for publications and articles” on page 401.

Execute `drop article` at the Replication Server that manages the source database. You must have `create object` permission.

The following example drops the `titles_art` article for the `pubs2_pub` publication.

```
drop article titles_art
  for pubs2_pub with primary at TOKYO_DS.pubs2
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Dropping the associated replication definition

To drop a replication definition associated with an article, include the `drop_repdef` clause when you execute `drop article`. Replication Server drops the replication definition if it is not referenced in other publications or subscriptions.

For example, to drop the `pubs2_pub` article and the replication definition it references, enter:

```
drop article titles_art
  for pubs2_pub with primary at TOKYO_DS.pubs2
  drop_repdef
```

## Translating datatypes using HDS

In a heterogeneous replication system, when information is replicated from one data server to another, values stored at the primary data server must often be altered so that they can be copied successfully to a different datatype at the replicate data server.

User-created function strings can produce these datatype translations, but require significant user input and are limited by the capabilities of the replicate data server.

To make datatype translations more readily available for different data servers, Replication Server provides heterogeneous datatype support (HDS), an easy-to-apply methodology for translating datatypes at the Replication Server. HDS supports selected datatype translations between these data servers:

- Adaptive Server Enterprise
- DB2
- Oracle
- Microsoft SQL Server
- UDB

When you use HDS, you can choose which columns and datatypes in the primary database are to be translated, and which replicate data servers will receive the translations.

Sources of information:

- See the *Heterogeneous Replication Guide*.
- See the *Replication Server Configuration Guide* for your platform for instructions for installing and setting up the objects that enables HDS.
- See the *Replication Server Reference Manual* for descriptions of the function strings.

## Overview

You can use HDS capabilities when replicating between:

- Adaptive Server databases – one Adaptive Server datatype to another Adaptive Server datatype
- Like non-Sybase databases – for example, DB2 TIMESTAMP to DB2 DATE

- Heterogeneous non-Sybase databases – Oracle to DB2, for example
- Adaptive Server and non-Sybase databases – Adaptive Server to Oracle, for example

If you are replicating information between Adaptive Servers, datatype translations are normally unnecessary. However, you can use HDS to perform datatype translations when datatypes differ in the primary and replicate databases.

HDS handles incompatibilities between the datatypes of the primary data server and the replicate data server. In general, these incompatibilities are of three types:

- Incompatible ranges – for example, the range of acceptable dates for Sybase datetime is January 1, 1753 through December 31, 9999. DB2, however, allows dates from January 1, 0001 through December 31, 9999.
- Incompatible formatting – for example, the primary data server date format is “CCYY-MM-DD,” but the replicate data server requires a date format of “MM/DD/CCYY.”
- Incompatible delimiters – for example, Sybase delimits binary data with an “0x” prefix, whereas Oracle surrounds binary data with single quotation marks.

The Replication Agent for each data server delivers replicate values to Replication Server in a datatype format that Replication Server understands, which includes the literal value, delimiter information, and other datatype attributes. Replication Server handles the value as its base datatype—one of the native Replication Server datatypes described in the *Replication Server Reference Manual*.

You can implement datatype translations in two ways:

- *Class-level translations* – translate all instances of a datatype for a particular connection.
- *Column-level translations* – translate all instances of a column described by a table replication definition.

## Getting started

- 1 Review the datatype translations available for your primary and replicate data servers. Determine the translations you want and the methods for delivering them:

- Class-level translations  
For lists of supported class-level datatype translations, see the *Heterogeneous Replication Guide*.
  - Column-level translations  
See Table 9-4, Table 9-5, and Table 9-5 for lists of supported datatypes and data servers.
  - A combination of class-level and column-level translations
- 2 Set up the environment and run the scripts that enable HDS for your system. Refer to the *Replication Server Configuration Guide* for your platform for instructions.
  - 3 Set up class-level and column-level translations using the procedures described in the following sections.

For information about creating subscriptions for replication definitions, refer to Chapter 11, “Managing Subscriptions”. For information about using function replication definitions for class-level translations, see Chapter 10, “Managing Replicated Functions”.

## Creating class-level translations

Class-level translations ensure that each time a value of a certain datatype is replicated from the primary to a particular data server, the datatype of that value is changed. Sybase provides the function strings and function-string classes necessary to produce these translations.

To set up class-level translations, follow these steps:

- 1 Set up and configure the replicate database gateway server.  
Refer to the *Replication Server Configuration Guide* for your platform for instructions.
- 2 Set up the database objects and run the scripts that install the function strings and function-string classes for your primary to replicate data server connections.  
Refer to the *Replication Server Configuration Guide* for your platform for instructions.
- 3 Create or alter the connections to specify the function-string class.

- If you are creating a new connection, Sybase provides a sample script that you can use to create the connection and specify the appropriate function-string class. You will need to modify the script for your installation.

Refer to the *Replication Server Configuration Guide* for your platform for instructions.

- If you are adding class-level translations to an existing connection, use the alter connection command as described in “Adding class-level translations to an existing connection” on page 321.

Sybase provides function-string classes for several Sybase and non-Sybase data servers:

- Adaptive Server Enterprise – `rs_sqlserver_function_class`
- DB2 – `rs_db2_function_class`
- Microsoft SQL Server – `rs_msss_function_class`
- Oracle – `rs_oracle_function_class`
- UDB – `rs_udb_function_class`

Each of these function classes contains function strings. For example, for the DB2 database, HDS provides these translations:

- DB2 to Adaptive Server and Adaptive Server to DB2
- DB2 to Microsoft SQL Server and Microsoft SQL Server to DB2
- DB2 to Oracle and Oracle to DB2

With the exception of `rs_db2_function_class`, each function class inherits from `rs_default_function_class`.

In general, you cannot add to, delete, or change any of these function-string classes or the functions they contain. You can modify `rs_sqlserver_function_class` for compatibility with earlier releases of Replication Server, but you cannot modify or alter any of its datatype translations. Although you can create classes that inherit from these classes, the classes you create cannot inherit any class-level translations from the parent class.

You are installing translations in the RSSD when you run the class-level installation scripts. If you do not run the scripts, no default class-level translations take place. Running the scripts replaces default Adaptive Server translations that would otherwise be inherited from `rs_default_function_class` with translations designed for a particular data server.

You activate class-level translations for a connection by specifying the function-string class when you create or alter the connection. When the function-string class is activated, all subsequent data replicated via that connection is translated according to the translations defined for that function-string class.

If a class-level translation is not specified for a published datatype (the datatype of the replicate data server), Replication Server simply translates the value from the Replication Agent to its base datatype format in the usual manner. For example:

- If no translation is specified for the Sybase datetime datatype, no translation is performed; the base datatype of datetime is datetime.
- If no translation is specified for rs\_db2\_timestamp, any rs\_db2\_timestamp value routed through the connection is translated to char(26), its base datatype.

Replication Server performs class-level translations after column-level translations and after subscription resolution, but before values are mapped to function strings. You can display a list of active function-string classes using `admin show_function_classes`. See “Using class-level and column-level translations together” on page 326.

## **Adding class-level translations to an existing connection**

If you want to add class-level translations to an existing connection, use the `alter connection` command. Follow these steps:

- 1 Run the appropriate install scripts. Refer to the *Replication Server Configuration Guide* for your platform.
- 2 Use the `alter connection` command to set the function class for the connection.

For example, to enable the translations for `rs_db2_function_class`, enter this command from the replicate Replication Server:

```
alter connection to db2_gateway1.db2_subsystem1
set error class to ansi_error
set function string class to rs_db2_function_class
...
```

- 3 Suspend and then resume the connection to activate the translations.

---

**Note** If you already have a DB2 database configured as a replicate database with an earlier version of Replication Server, continue to use the earlier version with Replication Server 12.0 and later and its HDS feature. The 12.0 and later function strings may not be compatible with earlier function string versions.

---

## System-defined variables

Class-level translations change the datatype of system-defined variables as well as column values.

For example, if a class-level translation changes `datetime` to `rs_db2_timestamp`, the `rs_origin_begin_time` system-defined variable, which is `datetime`, is translated to `rs_db2_timestamp` for that connection.

## Creating column-level translations

Column-level translations affect each replicated instance of a particular column (datatype) and table. They are defined using the `create replication definition` or `alter replication definition` command.

To set up column-level translations, you simply create or alter the replication definition, identifying the column to be translated and its initial and final datatypes using the `map to` option.

- If you are creating a new replication definition, use `create replication definition`.

For lists of supported datatype translations, see the *Heterogeneous Replication Guide*.

- If you are adding or altering a column in an existing table, use `alter replication definition`.

Sybase provides a set of datatype definitions and datatype classes that you can use to modify the datatype of the replicated columns. Each datatype class contains datatype definitions for a particular data server:

- Adaptive Server – `rs_sqlserver_dt_class`
- DB2 – `rs_db2_dt_class`
- Microsoft SQL Server – `rs_mssql_dt_class`



- Oracle – rs\_oracle\_dt\_class
- UDB – rs\_udb\_dt\_class

Datatype classes are not replicated and cannot be modified. Column-level translations are implemented after subscription resolution and before class-level translations. See “Using class-level and column-level translations together” on page 326 for more information.

You can activate a column-level translation for a particular column when you create or alter a table replication definition. The syntax for create replication definition with column and datatype variables specified for HDS is:

```
create replication definition replication_definition
with primary at data_server.database
...
(column_name [as replicate_column_name]
declared_datatype [null | not null]
[map to published_datatype])
...
```

where:

- The **declared datatype** depends on the datatype of the value delivered to the Replication Server from the Replication Agent:
  - If the Replication Agent delivers a native Replication Server datatype, such as datetime, to the Replication Server, the declared datatype is the native datatype.
  - Otherwise, the declared datatype must be the datatype definition for the original datatype at the primary database.

For example, the Replication Agent delivers a value in the DB2 TIMESTAMP datatype, as a character string with delimiters, to Replication Server. In this case, the declared datatype is the datatype definition rs\_db2\_timestamp. See Table 9-4, Table 9-5, and Table 9-5 for a list of datatype definitions and their datatype equivalents.

- The **published datatype** is the datatype of the column after the column-level translation (and before a class-level translation, if any). The published datatype is normally either a Replication Server native datatype or a datatype definition for the datatype in the replicate database. If the published datatype is omitted from the replication definition, it defaults to the declared datatype.

Both declared and published datatypes have a base datatype. For example, the datatype `rs_db2_timestamp` has a base datatype of `char(26)`; the native datatype `char(26)` also has a base datatype of `char(26)`. A datatype definition describes a non-Sybase datatype in terms of a Replication Server native datatype. The base datatype fixes the maximum and minimum length to be associated with the datatype definition and provides defaults for other datatype attributes. The base datatype defines the delimitation of values for the datatype definition when a value of that type is delivered to Replication Server either in Log Transfer Language (LTL) or in a command executed by a Replication Server administrator such as `create subscription`.

---

**Note** Native datatypes include all datatypes supported by Replication Server. However, you cannot use `text`, `unitext`, `image`, `rawobject`, and `rawobject` in row datatypes for defining a datatype definition; neither can you use these datatypes as the source or target of a translation.

---

For example, to create a table replication definition `ase_employee_repdef_for_db2` that translates values in the `birthdate` column from `datetime` (`birthdate`'s primary table datatype) to DB2 `DATE` datatype for the replicate database, log in to the primary Replication Server and enter:

```
create replication definition
ase_employee_repdef_for_db2
with primary at ase_server.ase_database
with all tables named 'employee'
(empid int,
first_name char(20),
last_name char(20),
...
birthdate datetime map to rs_db2_date,
salary money,
...)
```

In this example, `birthdate` is the column name, `datetime` is the declared datatype, and `rs_db2_date` is the published datatype. Because the declared datatype is a native datatype, the native and base datatype are the same. That is, the base datatype of `datetime` is `datetime`. The published datatype `rs_db2_date` is a datatype definition for DB2, and its base datatype is `char(10)`.

## How datatype definitions work

Datatype definitions allow you to translate from one datatype to another without losing valuable information.

When used as the declared datatype, a datatype definition provides the mechanism for capturing both the literal value and its datatype attributes—such as delimiters, range information, precision, scale, length, and maximum and minimum values—and translating them into a native datatype format that Replication Server can process.

When used as a published datatype, a datatype definition takes the value in Replication Server native datatype format, including its attribute information, and translates that information into a datatype format acceptable to another database, retaining as much information as the published datatype can accommodate.

When data definitions are used for both the declared and published datatypes, both translations take place.

The following tables list the available datatype definitions for each supported non-Sybase datatype.

---

**Note** Microsoft SQL Server does not directly support the new unsigned integer types in 15.0 and requires to use a map to clause in their replication definitions.

---

Table 9-4 lists the supported DB2 datatypes and their datatype definition equivalents.

**Table 9-4: Datatype definitions for DB2 datatypes**

<b>DB2 datatype</b>	<b>Datatype definition</b>
CHAR FOR BIT DATA	rs_db2_char_for_bit
DATE	rs_db2_date
TIME	rs_db2_time
TIMESTAMP	rs_db2_timestamp
VARCHAR FOR BIT DATA	rs_db2_varchar_for_bit
TINYINT	rs_db2_tinyint
DECIMAL	rs_db2_decimal
NUMERIC	rs_db2_numeric

Table 9-5 lists supported Oracle datatypes and their datatype definition equivalent.

**Table 9-5: Datatype definitions for Oracle datatypes**

Oracle datatype	Datatype definition
RAW	rs_oracle_binary
DATE	rs_oracle_date
DATE (with time)	rs_oracle_datetime
NUMBER (INTEGER)	rs_oracle_int
NUMBER (FLOAT)	rs_oracle_float
NUMBER (DECIMAL)	rs_oracle_decimal

## Column-level translations and multiple replication definitions

In general, a column declared in multiple replication definitions must use the same *declared datatype* in each replication definition—although *published datatypes* can differ.

rawobject and rawobject in row (Java) columns declared in multiple replication definitions, however, can use either the rawobject (or rawobject in row) datatype *or* its base datatype for the declared datatype. For example, you can use rawobject and image or rawobject in row and varbinary in multiple replication definitions for the same Java column. See *Java in Adaptive Server Enterprise* for detailed information about Java columns in Adaptive Server.

## Using class-level and column-level translations together

If you activate class- and column-level datatype translations for the same column, both are applied. Column-level translations are performed after subscription resolution and before class-level translations, just prior to delivery to the replicate database.

This order of execution ensures that column-level translations supersede class-level translations. That is, translations for a particular connection (class-level translations) do not affect translations defined for a particular table and column (column-level translations).

## Verifying translations

You can verify how translations alter values before you set up column- or class-level translations. Use the `admin translate` command to view the results of a particular translation. `admin translate` accepts a value and a source and target datatype and returns the target value. It is most useful with the diagnostic version of Replication Server, which, if the translation fails, allows you to trace the reason for the failure.

The syntax is:

```
admin translate, value, source_datatype, target_datatype
```

where:

- *value* is the literal representation of the value being translated—including delimiters as required by the base datatype of the source datatype.
- *source\_datatype* is the datatype definition or datatype for the value you want to translate.
- *target\_datatype* is the datatype definition or datatype for the value after translation.

If the base datatype of either the source or target datatype requires a length specification, such as `char(26)`, enclose the datatype name in quotes.

For example, to verify the translation of a date from `db2_date` to `datetime`, log in to Replication Server and enter:

```
admin translate, '04/29/1989', db2_date, datetime
```

In this example, *value* is the character string “04/29/1989,” and you must enclose it in single quotes. Refer to the *Replication Server Reference Manual* for a complete description of `admin translate` and further examples.



# Managing Replicated Functions

This chapter describes how to replicate the execution of a stored procedure from the source database to the destination database using replicated functions.

Topic	Page
Prerequisites and restrictions	330
Using replicated functions	334
Implementing an applied function	337
Implementing a request function	340
Marking stored procedures for replication	344
Subscribing to replicated functions	345
Modifying or dropping replicated functions	345
Using publications for stored procedures	349

When you use function replication, Replication Server replicates the execution of a stored procedure to the destination database. That is, when a stored procedure is executed at the source database, the replication server invokes the execution of another stored procedure at the destination database. The two stored procedures need not have the same name nor perform the same tasks.

Refer to the *Replication Server Design Guide* for information about replication system design issues that concern replicated stored procedures.

This chapter covers the distribution of stored procedures via function replication definitions. The distribution of stored procedures associated with table replication definitions is described in Appendix A, “Asynchronous Procedures,” in the *Replication Server Administration Guide Volume 2*. The request function distribution with version earlier than 15.1 without subscription is described in Appendix C, “Pre-15.1 Request Function Replication” in the *Replication Server Administration Guide Volume 2*.

You identify the stored procedure at the source and the information that is to be passed to the destination by creating a function replication definition, which specifies:

- The names of the stored procedures at the source and destination databases (if they are different)
- The datatypes and parameters that are to be passed to the destination stored procedure

To satisfy the requirements of distributed applications, Replication Server provides two ways to implement replicate functions. Use:

- An **applied function** to deliver a transaction to a replicate database by the maintenance user. See “Applied functions” on page 335 for more information.
- A **request function** to deliver a transaction to a replicate database by the same user who invokes the stored procedure at the primary database. See “Request functions” on page 336 for more information.

The `maint_user` runs the transaction at the replicate database if the function is replicated through applied function replication definition. The `origin_user` runs the transaction if the function is replicated through request function replication definition at the replicate database.

## Prerequisites and restrictions

Before you implement applied or request functions in your replication system, be sure that you have met the prerequisites discussed below, and that you understand the restrictions on the use of replicated stored procedures.

### Replicated function prerequisites

- Understand how you will use applied or request functions to meet your application needs. Refer to the *Replication Server Design Guide* for more information.
- Set up a RepAgent at the primary Replication Server. See Chapter 4, “Managing a Replication System” and the *Replication Server Configuration Guide* for details.
- Set up routes from the primary Replication Server to the replicate Replication Server. See Chapter 6, “Managing Routes” to learn how to set up routes.



- Replicated functions can be used with applications that involve fragmented primary data. To do this, create a function replication definition and a stored procedure for each primary fragment. Refer to the *Replication Server Design Guide* for more information about working with fragmented primary data.

In general, the information in this chapter assumes Replication Server basic primary copy model, where a single source database distributes data to one or more destination databases. Refer to “Replication Server basic primary copy model” on page 6 for a detailed description of this model.

## Replicated function restrictions

- The names of all replication definitions, including function replication definitions, must be unique in the replication system.
- When you create an applied function replication definition for a primary function in your replication system, make sure that the function does not have an existing function replication definition that satisfies both these conditions:
  - The function replication definition is created using the create function replication definition command.
  - The function replication definition is used for the request function replication without subscription in Replication Server 15.0.1 and earlier version.

Otherwise, the existing request function replication will be disabled. See Appendix C, “Pre-15.1 Request Function Replication” for more information in the *Replication Server Administration Guide Volume 2*.

- Replication Server does not support nested transactions—those containing begin or commit statements—within replicated stored procedures.

If stored procedures with nested stored procedures are marked for replication:

- The RepAgent forwards only the outer stored procedure call to the Replication Server.
- The RepAgent shuts down.
- An error message appears in the Adaptive Server error log.

When the `maint_user` or the replicate database replicates a stored procedure, using `sp_setreproc` or `sp_setreplicate`, Adaptive Server always executes the stored procedure within a transaction. Even if you have not explicitly executed the replicated stored procedure within a transaction at the primary database, Adaptive Server places an implicit begin transaction at the start of the procedure when it is applied by the `maint_user` in the replicate database.

For more information, see `dsi_max_xacts_in_group`, in “Connection parameters that affect performance” in the *Replication Server Administration Guide Volume 2*. If the replicated stored procedure contains such commands as `begin transaction`, `commit transaction`, or `rollback transaction`, errors may result when you execute the procedure. For example, a `rollback transaction` command might roll back to the start of the transaction group, rather than to the nested `begin transaction` command that was the intended rollback point.

- Replicated functions, like Adaptive Server stored procedures, cannot contain parameters with text and image datatypes. Refer to the *Adaptive Server Enterprise Reference Manual*.
- Adaptive Server logs a replicated stored procedure invocation in the database in which the enclosing transaction was started:
  - If the user does not begin a transaction explicitly, Adaptive Server begins one in the user’s current database before the stored procedure is invoked.
  - If the user begins the transaction in one database and then executes a replicated stored procedure in another database, the execution is still logged in the database where the transaction began.
- If a single transaction invokes one or more request functions and executes applied functions or contains data modification language, or a mixed-mode transaction, Replication Server processes the request functions after all the other operations have completed, together in a separate transaction.
- When you use replicated functions and heterogeneous datatype translations:
  - You cannot alter the datatype of a parameter value using `create applied/request function replication definition` or `alter applied/request function replication definition`. However, you can use datatype definitions to declare parameters for applied function replication definitions, which are then subject to class-level translations.

- Replication Server does not perform translations on parameter values for request functions. However, during function-string mapping, the delimiters defined for the parameter values of their declared datatype are used to generate the SQL.

---

**Warning!** Do not put a commit statement inside a replicated function as this may cause a duplicate key and make Replication Server recovery fail.

---

## Commands for managing function replication definitions

Table 10-1 lists the Replication Server commands used to work with function replication definitions.

**Table 10-1: Commands for managing function replication definitions**

Command	Task
drop function replication definition	Removes a function replication definition from the replication system. You must drop all subscriptions for a function replication definition before you can drop the replication definition. See “Modifying or dropping replicated functions” on page 345.
create applied replication definition	Creates an applied function replication definition that describes the stored procedure and its parameters, for both the primary and replicate databases. It also describes the location of the primary data. The maint_user applies the applied function at the replicate site. See “Implementing an applied function” on page 337.
create request replication definition	Creates a request function replication definition that describes the stored procedure and its parameters, for both the primary and replicate databases. It also describes the location of the primary data. The same user running the stored procedure at the primary site applies the request function at the replicate site.
alter applied replication definition	<p>Modifies an applied function replication definition, which is created with create applied function replication definition command. For example, it:</p> <ul style="list-style-type: none"> <li>• Specifies a different name for the primary stored procedure invoked at the source database.</li> <li>• Specifies a different name for the stored procedure invoked at the destination database.</li> <li>• Adds parameters or searchable parameters.</li> <li>• Changes how the replication definition is used in replicating to a standby database.</li> </ul> <p>See “Modifying or dropping replicated functions” on page 345</p> <p>If parameters are added, the change applies to all applied function replication definition created for this primary function.</p>

Command	Task
alter request replication definition	<p>Modifies a request function replication definition, which is created with create request function replication definition command. For example, it:</p> <ul style="list-style-type: none"> <li>• Specifies a different name for the primary stored procedure invoked at the source database.</li> <li>• Specifies a different name for the stored procedure invoked at the destination database.</li> <li>• Adds parameters or searchable parameters.</li> <li>• Changes how the replication definition is used in replicating to a standby database.</li> </ul> <p>See “Modifying or dropping replicated functions” on page 345</p> <p>If parameters are added, the change applies to all request function replication definitions created for this primary function.</p>
create function replication definition	<p>Creates a function replication definition that describes the stored procedure, and its parameters, for replication. It also describes the location of the primary data. This command is deprecated and is replaced by create applied function replication definition and create request function replication definition commands</p> <p>See “Implementing an applied function” on page 337 and “Implementing a request function” on page 340.</p>
alter function replication definition	<p>Modifies a function replication definition. For example, it:</p> <ul style="list-style-type: none"> <li>• Specifies a different name for the stored procedure invoked at the destination database</li> <li>• Adds parameters or searchable parameters</li> <li>• Changes how the replication definition is used in replicating to a standby database</li> </ul> <p>This command can only be used to modify the function replication definition created with create function replication definition command. See “Modifying or dropping replicated functions” on page 345.</p>

Also, see Table 9-1 on page 249 and Table 11-3 on page 370.

## Using replicated functions

A **replicated stored procedure** is an Adaptive Server stored procedure that you have marked for replication using either `sp_setrepproc` or `sp_setreplicate`.

A **function replication definition** describes the primary and the replicated stored procedure, its parameters, and its location. You can use these three commands to create a function replication definition:

- create applied function replication definition
- create request function replication definition
- create function replication definition (deprecated)

When you create a function replication definition, Replication Server creates a **function**, which contains the information in the function replication definition.

When a replicated stored procedure that has its own **function replication definition** is invoked, its function is transferred from the source to a destination Replication Server. In most cases, the replicated stored procedure is invoked at the primary database and delivered to the replicate database. The only exception is the request function replication with a version earlier than 15.1 without subscription and with such replication definition, where the stored procedure is invoked at the replicate database and delivered to the primary database. In all cases, the primary Replication Server is always the Replication Server where the replication definition is created. This Replication Server controls the primary database. See Appendix C, “Pre-15.1 Request Function Replication” in the *Replication Server Administration Guide Volume 2*.

The function passes parameters to the corresponding stored procedure that is, in turn, invoked in the destination database. A **function string** translates the function to a syntax that a subscribing database can interpret. When used correctly, function replication can dramatically improve performance because it can encapsulate multiple operations in a single function. Replicated stored procedures do not have to modify any data in order to be replicated.

## Applied functions

Use an **applied function** to distribute operations performed in a primary database to replicate databases. Applied functions allow you to realize important performance benefits. For example, if a client application must update a large number of row changes, you can create an applied function that changes many rows, rather than replicating the rows individually.

To use an applied function, you first create a stored procedure in the primary database and a corresponding stored procedure in the replicate database. Use `sp_setrepproc` command to mark the stored procedure to be replicated. At the primary Replication Server, you create an applied function replication definition for the stored procedure. Replicate Replication Servers can subscribe to the function replication definition. When the stored procedure in the primary database is invoked, the replicate Replication Server in turn executes the stored procedure in the subscribing replicate database.

Replication Server does not know in advance what data is needed by the stored procedure at the replicate databases until the execution of the stored procedure is subscribed, thus you must use bulk materialization or the no-materialization method when you subscribe to a function replication definition.

Replication Server executes the stored procedure in the replicate database as the maintenance user, which is consistent with normal data replication.

See “Implementing an applied function” on page 337 for step-by-step instructions.

## Request functions

Use a **request function** to deliver a replicated stored procedure from a primary database to the replicate database through the original user, the same user who invokes the stored procedure at the primary database. This type of function replication is usually used to enable the remote site to make changes to the central data with the authorized user. For example, a client application at a remote location needs to make changes to the central data. The client application first executes a stored procedure at the remote site—a procedure that may or may not make changes at the remote database. When the stored procedure executes, the replicate Replication Server passes a request function to the central site, where a corresponding stored procedure is invoked that updates the central data. In this example, the remote database is the primary database, while the central database is the replicate database of this request function.

With the primary copy model, a single central database contains all the latest updates. A client application at a remote site can update the central data using request functions. As updates occur at the central table, Replication Server captures the updates and sends them to replicate data servers through applied functions. Execution of stored procedures are stored in the Replication Server stable queues until they can be delivered to the appropriate databases.

To use a request function, create a stored procedure in the remote database and a corresponding stored procedure in the central database. Then, create a request function replication definition at the Replication Server that controls the remote database. The Replication Server that controls the central database can subscribe to this request function replication definition. When the stored procedure in the remote database is invoked, it invokes the stored procedure in the central database.

The Replication Server that manages the central database executes the stored procedure in the central database as the user who executed the stored procedure in the remote database. This guarantees that only authorized users can change central data.

In an application, Replication Server may replicate some or all of the data that is changed in the central database. The changes are distributed to the remote databases managed by Replication Servers that have subscriptions to table replication definitions or as separate applied functions. Either way, the effect of a transaction arrives at the central and then remote databases.

When you use request functions, all updates are made at the central database. This preserves Replication Server primary copy data model and protects the replication system from network failure and excess traffic.

See “Implementing a request function” on page 340 for step-by-step instructions.

## Implementing an applied function

The applied and request function are very similar. The difference is that the maintenance user executes the applied function at the replicate site, while the same user who executes the stored procedure at the primary database executes the request function at the replicate site.

To implement an applied function:

- 1 Review the requirements described in “Prerequisites and restrictions” on page 330.
- 2 Set up replicate databases containing replicate tables that the stored procedure will modify.
- 3 In the primary database, create the stored procedure. The stored procedure may or may not modify primary data. For example, this stored procedure uses the @pub\_name parameter to update the pub\_name column of the publishers table:

```
create proc update_pubs
@pub_id char(4), @pub_name varchar(40),
as
update publishers
set pub_name = @pub_name
where pub_id = @pub_id
```

- 4 In the primary database, mark the stored procedure for replicated function delivery, using the sp\_setrepproc system procedure. For example:

```
sp_setrepproc update_pubs, 'function'
```

See “Marking stored procedures for replication” on page 344 for details.

- 5 In the replicate database, create a stored procedure with the same parameters and datatypes as the stored procedure in the primary database. Typically, the two stored procedures perform the same operations. For example:

```
create proc update_pubs
pub_id char(4), @pub_name varchar(40),
as
update publishers
set pub_name = @pub_name
where pub_id = @pub_id
```

---

**Note** The stored procedure created in the replicate database does not have to have the same name, but must have the same parameter name and datatype.

---

**Warning!** A stored procedure invoked in a replicate database in applied function delivery is invoked inside a user-defined transaction. See the *Adaptive Server Enterprise Transact-SQL User’s Guide* for information about operations that are not allowed inside user-defined transactions (for example, the dump transaction and dump database commands).

---

Do not mark this stored procedure as replicated. In applied function delivery, only the stored procedure in the primary database is marked as replicated.

However, if the replicate database modifies a standby database, mark the stored procedure in the active and standby replicate databases as replicated if you want to use stored procedure replication to the standby.

- 6 In the replicate database, grant execute permission on the stored procedure to the maintenance user. For example:

```
grant execute on update_pubs to maint_user
```

- 7 In the primary Replication Server, create an applied function replication definition for the stored procedure. For example:

```
create applied function replication definition
update_pubs_rep
with primary at TOKYO_DS.pubs2
with all functions named update_pubs
(@pub_id char(4), @pub_name varchar(40),
@state char(2))
```



```
searchable parameters (@pub_name, @state)
```

The function replication definition must use the same parameter names and datatypes as the stored procedure in the primary database. You have the option to include only the parameters you want to replicate. If the function replication definition has 0 parameters, you must still include the parentheses for this clause.

If you specify searchable parameters, you can subscribe to function invocations based on the value of the function's parameters. In the preceding example, @pub\_name and @state are searchable parameters. Thus, for example, they can subscribe only to "CA" updates.

If you want to replicate the Adaptive Server timestamp datatype, declare the datatype binary(8) in the function replication definition.

Refer to Chapter 3, "Replication Server Commands," in the *Replication Server Reference Manual* for more information about create applied function replication definition command.

See "Modifying or dropping replicated functions" on page 345 for information about changing function replication definitions.

- 8 When you create a function replication definition, Replication Server automatically creates a corresponding function in the default function-string class. See "User-defined functions" on page 14 in the *Replication Server Administration Guide Volume 2* for more information.

If you are not using a default function-string class or a class inherited from the default or if you want to customize the function's invocation, you need to create a function string for the user-defined function. See "Creating or modifying a function string for a replicated function" on page 348 for more information.

- 9 In the replicate Replication Server, create a subscription to the function replication definition, using create subscription and the no-materialization method or define subscription and the other bulk materialization commands.

---

**Note** You must use the no-materialization method or bulk materialization—instead of atomic or nonatomic materialization—because Replication Server cannot determine in advance what data is needed for the stored procedure at the replicate site.

---

For example:

```
create subscription pubs_sub
```

```
for update_pubs_rep
with replicate at SYDNEY_DS.pubs2
where @state = 'CA'
without materialization
```

If you specified searchable parameters in the function replication definition, you can subscribe to function invocations based on the value of the function's parameters. In this example, the subscription only receives rows if the value of the @state parameter is equal to CA.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about create subscription command. See also “Using create subscription for no materialization” on page 376.

- 10 Verify that all Replication Server and database objects in steps 1 through 9 exist at the appropriate locations. You should now be able to execute the applied function.

Refer to Chapter 6, “Adaptive Server Stored Procedures,” in the *Replication Server Reference Manual* for information about stored procedures, such as rs\_helpfunc, that you can use to query the RSSD for information about the replication system.

## Implementing a request function

The applied and request function are very similar. The difference is that the maintenance user executes the applied function at the replicate site and the same user who executes the stored procedure at the primary database executes the request function at the replicate site.

To implement a request function:

- 1 Review the requirements described in “Prerequisites and restrictions” on page 330.
- 2 In the replicate Adaptive Server, create a login name and password for the user who will execute the stored procedure at the replicate Adaptive Server.

See Chapter 8, “Managing Replication Server Security” for details.

- 3 In the replicate database, create a replicate stored procedure that updates the real data. For example:

```
create proc update_pubs
@pub_id char(4), @pub_name varchar(40)
as
update publishers
set pub_name = @pub_name
where pub_id = @pub_id
```

---

**Warning!** A stored procedure invoked in request function delivery is invoked inside a user-defined transaction. See the *Adaptive Server Enterprise Transact-SQL User's Guide* for information about operations that are not allowed inside user-defined transactions (for example, the dump transaction and dump database commands).

---

Do not mark this stored procedure as replicated; however, if this database is also part of a warm standby application, then mark the stored procedure in the active database as replicated if you want to replicate stored procedures to the standby database.

- 4 In the replicate database, grant execute permission on the stored procedure to the same user for whom you created a login name and password in step 2. When the request function is replicated in the replicate database, this user executes it. For example:

```
grant execute on update_pubs to pubs_user
```

- 5 In the primary database, create a request primary stored procedure with the different name, but the same parameters and datatypes as the stored procedure in the replicate database. The new stored procedure should either do nothing or should display a message to indicate a pending update. Typically, the purpose of this stored procedure is to send a request to other databases, instead of performing any data changes on its own database. For example:

```
create proc update_pubs_request
@pub_id char(4), @pub_name varchar(40)
as
```

```
print "Transaction accepted."
```

---

**Note** Use a different name for the stored procedure you create in the replicate and primary databases. In the typical applications, the function will replicate back to the primary database later as an applied function. When you create the request function replication definition in step 8, you must specify the name of the stored procedure in the primary and replicate databases.

---

- 6 In the primary database, mark the stored procedure for replicated function delivery using the `sp_setrepproc` system procedure. For example:

```
sp_setrepproc update_pubs_request, 'function'
```

See “Marking stored procedures for replication” on page 344 for details.

- 7 In the primary database, grant execute permission on the stored procedure to the primary Replication Server user who will invoke it. For example:

```
grant execute on update_pubs_request to pubs_user
```

- 8 In the *primary* Replication Server, which manages the request primary stored procedure, create a request function replication definition for this stored procedure. For example:

```
create request function replication definition
  update_pubs_request_rep
with primary at TOKYO_DS.pubs2
with primary function named update_pubs_request
with replicate function named update_pubs
  (@pub_id char(4), @pub_name varchar(40)),
  @state char(2))
searchable parameters (@state)
```

The request function replication definition must use the same parameter names and datatypes as the stored procedure in the replicate database. You have the option to include only the parameters you want to replicate.

See Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about create request function replication definition command.

- 9 When you create a function replication definition, Replication Server automatically creates a corresponding user-defined function.

If you are not using a default function string or wish to customize the function's invocation, you need to create a function string for the user-defined function. See "Creating or modifying a function string for a replicated function" on page 348 for more information.

- 10 In the replicate Replication Server, create a subscription to the request function replication definition, using create subscription and the no materialization method or define subscription and the other bulk materialization commands. For example:

```
create subscription pubs_sub
for update_pubs_request_rep
with replicate at SYDNEY_DS.pubs2
where @state = 'CA'
without materialization
```

If you specified searchable parameters in the function replication definition, you can subscribe to function invocations based on the value of the function's parameters. In this example, the subscription only receives rows if the value of the @state parameter is equal to "CA".

See Chapter 3, "Replication Server Commands," in the *Replication Server Reference Manual* for more information about create subscription command. See also "Using create subscription for no materialization" on page 376.

---

**Note** You must use the no-materialization method or bulk materialization, instead of atomic or nonatomic materialization because the Replication Server cannot determine in advance what data is needed for the stored procedure at the replicate site.

---

- 11 Verify that all Replication Server and database objects in steps 1 through 10 exist at the appropriate locations. You should now be able to execute the request function at the primary database.

Refer to Chapter 6, "Adaptive Server Stored Procedures," in the *Replication Server Reference Manual* for information about stored procedures, such as rs\_helpfunc, that you can use to query the RSSD for information about the replication system.

## Marking stored procedures for replication

The system procedure `sp_setreproc` is used to mark stored procedures for replication. The syntax is:

```
sp_setreproc [proc_name [, {'false' | 'table' | {'function' [, {'log_current' | 'log_sproc'} ] } } ]]
```

where:

*proc\_name* – the name of a stored procedure in the current database.

'function' – enables replication for a stored procedure associated with a function replication definition.

'table' – enables replication for a stored procedure associated with a table replication definition. For information on replicating stored procedures associated with table replication definitions, see Appendix A, “Asynchronous Procedures,” in the *Replication Server Administration Guide Volume 2*.

'false' – disables replication for the stored procedure.

'log\_current' – logs the execution of the stored procedure you are replicating in the current database, not in the database where the stored procedure resides.

'log\_sproc' – logs the execution of the stored procedure you are replicating in the database where the stored procedure resides, not in the current database. 'log\_sproc' is the default parameter.

Use `sp_setreproc` according to these guidelines:

- To list all replicated objects in the database, enter `sp_setreproc` with no parameters.
- To determine the replication status of the stored procedure, enter `sp_setreproc` with the stored procedure name only.
- Enter `sp_setreproc` with the stored procedure name and 'function', 'table', or 'false' to enable each type of replication or to disable replication for the stored procedure. You must be the System Administrator or the Database Owner to use `sp_setreproc` to change the replication status of a stored procedure.
- To log the execution of a replicated stored procedure in the database you choose, enter `sp_setreproc` with 'log\_current', to log execution in the current database, or 'log\_sproc', to log execution in the database where the stored procedure resides.

For either applied or request function replication, specify 'function' to indicate the type of replication definition associated with the stored procedure.

For more information on `sp_setreproc`, see the *Replication Server Reference Manual*.

## Subscribing to replicated functions

You must create subscriptions to function replication definitions for either applied or request functions using `create subscription` and the no-materialization method or `define subscription` and the other commands for bulk materialization: `activate subscription`, `validate subscription`, and `check subscription`.

The only exception is the function replication definitions with versions earlier than 15.1 used for request functions without subscription. See Appendix C, “Pre-15.1 Request Function Replication” for information.

If you specified searchable parameters in the function replication definition, you can subscribe to a function based on the value of its parameters.

You drop subscriptions to function replication definitions using `drop subscription`. They are dropped without purging the replicate data associated with the function. You do not need to specify the `without purge` option.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual*, and related commands, for the full syntax for bulk materialization commands. Also see “Bulk materialization” on page 357.

## Modifying or dropping replicated functions

This section explains how to modify or drop replicated functions.

### Before modifying a function replication definition

- 1 Change the stored procedure at the primary or replicate data server and provide defaults for new parameters, if necessary.
- 2 As a precaution, quiesce the system. Altering functions while updates are in process can have unpredictable results.

See Chapter 4, “Managing a Replication System” for information on how to quiesce the system.

## Altering a function replication definition

- 1 Quiesce the replication system using the Sybase Central Replication Manager plug-in or the procedure described in the *Replication Server Troubleshooting Guide*.

Quiesce first the primary updates and ensure that all primary updates have been processed by the replication system. If you are unable to do that, then the old updates in the primary log will not have values for new parameters, and the replication system will use nulls. Take this into account when altering function strings in step 4.

- 2 Alter the stored procedure at the primary and the replicate sites.
- 3 Alter the function replication definition. Wait for the modified function replication definition to arrive at the replicate sites.
- 4 Alter any function strings pertaining to the function replication definition, if necessary. Wait for the modified function strings to arrive at the replicate sites.
- 5 Modify subscriptions on the function replication definition at replicate sites, if necessary. To modify a subscription, drop and re-create it using drop subscription and create subscription (with no materialization option).

Altering a request or applied replication definition does not affect current subscriptions. If new parameters are added to the function replication definition, they are replicated with any new updates for all existing subscriptions.

- 6 Resume updates to the data at the primary database.

## Modifying a function replication definition

To add new parameters, add new searchable parameters, or change the name of the destination stored procedure, use alter applied function replication definition and alter request function replication definition commands to alter the function replication definition. The syntax for this command is:

```
alter applied function replication definition function_applied_rep_def
{with replicate function named 'proc_name' |
```



```
add @param_name datatype[, @param_name datatype]... |
add searchable parameters @param_name[, @param_name]... |
send standby {all | replication definition} parameters}
```

```
alter request function replication definition function_request_rep_def
{with replicate function named 'proc_name' |
add @param_name datatype[, @param_name datatype]... |
add searchable parameters @param_name[, @param_name]... |
send standby {all | replication definition} parameters}
```

These two commands are used to change the function replication definitions created by create applied function replication definition and create request function replication definition command respectively. See Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about alter applied function replication definition and alter request function replication definition commands.

See Appendix C “Pre-15.1 Request Function Replication” in the *Replication Server Administration Guide Volume 2* for information on modifying pre-15.1 function replication definition.

See “Creating or modifying a function string for a replicated function” on page 348 for information about function strings for function replication definitions.

To add new searchable parameters to the where clause of a define subscription command, drop and re-create the subscription for the function replication definition. For more information about subscribing to function replication definitions, see “Implementing an applied function” on page 337 and “Implementing a request function” on page 340.

## Dropping a function replication definition

To change or remove parameters, or to rename a function replication definition, use the drop function replication definition command to drop it. Then re-create it. The syntax for this command is:

```
drop function replication definition function_rep_def
```

When you drop a function replication definition, the associated user-defined function and function string are also dropped. Subscriptions to a function replication definition must be dropped first. You can re-create the subscriptions after you re-create the function replication definition.

## Creating or modifying a function string for a replicated function

When you create or alter a function replication definition, Replication Server automatically creates or alters the corresponding user-defined function. You must, however, create a function string for the user-defined function if you are not using a class that inherits function strings from `rs_default_function_class`, either directly or indirectly.

See “User-defined functions” on page 14 in the *Replication Server Administration Guide Volume 2* for more information.

Create a function string for a user-defined function in the function-string class assigned to the destination database for the replicated function. Use `create function string` at the primary Replication Server to create a function string for a user-defined function.

See “Function strings and function-string classes” on page 33 in the *Replication Server Administration Guide Volume 2* for more information.

When you drop a function replication definition, Replication Server always drops the user-defined function and function strings.

You can customize function strings in function-string classes that allow it. In a typical application, the replicated user-defined function passes stored procedure parameter values to the destination Replication Server, and the function string executes the stored procedure with these values in the destination database.

To change the default function string to perform some other action, such as inserting data into an audit log, use the `alter function string` command at the primary Replication Server for the replicated function. The function-string class assigned to the destination database for the replicated function must allow you to customize function strings.

See Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2* for information on creating and altering function strings. Also refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual*, for more information about `create function string` command.

## Using publications for stored procedures

You can use publications to select stored procedures and/or tables, along with their replication definitions, and subscribe to all of them as a group.

Publications let you organize your replication definitions and subscriptions and then monitor their status with a single command.

Refer to “Using publications” on page 308 for procedures for creating and managing publications. Refer to “Using publication subscriptions” on page 395 for procedures for creating and managing publication subscriptions.



# Managing Subscriptions

This chapter describes setting up and managing subscriptions for replicated data.

Topic	Page
Overview	351
Subscription materialization methods	353
Dematerialization processing	365
Monitoring materialization and dematerialization	366
Before you create subscriptions	368
Using subscription commands	370
Subscription example	382
Materializing text, unitext, image, and rawobject data	386
Subscriptions for columns with heterogeneous datatypes	387
Bitmap subscriptions	388
Obtaining subscription information	390
Using publication subscriptions	395

## Overview

Subscriptions resemble SQL select statements. They identify the replication definition or publication to which you are subscribing, the source and destination databases and data servers, and the **materialization** method by which the initial information is to be copied. You can use a where clause to specify a subset of rows or parameters that the destination database receives from the source database. This chapter describes how to materialize subscription data and manage subscriptions.

Materialization is 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. Initialization from media is called bulk materialization. You use one of four materialization methods, depending on how you want materialization to affect the replication system. See “Subscription materialization methods” on page 353 for more information.

Subscriptions for database replication definitions instruct Replication Server to replicate database objects from the primary to the replicate database. You can choose to replicate or not replicate individual tables, transactions, functions, system stored procedures, and data definition language (DDL). This method requires only a single database replication definition for each primary database and a single subscription for each subscribing database. See Chapter 12, “Managing Replicated Objects Using Multisite Availability,” for detailed information about database replication definitions and database subscriptions.

Subscriptions for table replication definitions instruct Replication Server to replicate data from primary tables into specified replicate tables. After you have created a replication definition for a primary table, replicate sites must subscribe to the replication definition at the primary database to receive updates.

Subscriptions for function replication definitions require you to use the no-materialization or the bulk materialization methods. See “No materialization” on page 357 and “Bulk materialization” on page 357. See also Chapter 10, “Managing Replicated Functions” for information about replicated functions.

You can subscribe to a group of replication definition articles by subscribing to a publication. Publication subscriptions cannot contain where clauses. To subscribe to a subset of rows in an article, you must include a where clause when you create the article. See “Using publication subscriptions” on page 395 for information about subscribing to publications.

You create subscriptions at the Replication Server managing the database where the replicate data is to be maintained. Your previously created replication definition provides the location of the primary data and defines the structure of the primary table and optionally, of the replicate table, where they differ.

## Subscription materialization methods

Materializing a subscription copies the requested data from the primary database to the replicate database, thereby initializing the replicate table. Subscriptions are added to the `rs_subscriptions` system table for both the primary and the replicate Replication Server. The materialization method you select determines how you create subscriptions.

Because a subscription can replicate a large set of rows, materialization can burden the network or impede applications that use the primary or replicate data. Replication Server offers four methods for creating subscriptions, so you can regulate the effects of materialization on the replication system.

Table 11-1 summarizes the materialization methods you can use to create subscriptions, including commands required for the process.

**Table 11-1: Subscription materialization methods**

Method	Description
Atomic materialization (default)	<p>This method, invoked using the default form of the <code>create subscription</code> command, copies subscription data through the network in a single atomic operation. Replication Server executes the <code>rs_select_with_lock</code> function to retrieve the primary data.</p> <p>This method provides complete consistency throughout the materialization process, but may temporarily obstruct transactions using the primary or replicate data. Do not use this method for large subscriptions if a long-running transaction is unacceptable in the primary database.</p> <p>For details, see “Atomic materialization” on page 354.</p>
Nonatomic materialization	<p>This method, invoked using the <code>create subscription</code> command with the <code>without holdlock</code> clause, is similar to the atomic method, except that consistency constraints during materialization are relaxed to allow clients at the primary database to process transactions during materialization. Replication Server executes the <code>rs_select</code> function to retrieve the primary data. Subscription data is copied in a series of transactions.</p> <p>Because users are allowed to update primary data, this method may result in transactional inconsistency and incomplete data during materialization. When materialization is complete, all inconsistencies are fully corrected. Autocorrection for the replicate table must be enabled to resolve inconsistencies.</p> <p>For details, see “Nonatomic materialization” on page 355.</p>
No materialization	<p>This method, invoked using the <code>create subscription</code> command with the <code>without materialization</code> clause, allows you to create a subscription when the subscription data already exists at the replicate database.</p> <p>You can use this method to create subscriptions to table replication definitions, function replication definitions, and database replication definitions.</p> <p>For details, see “No materialization” on page 357.</p>

Method	Description
Bulk materialization	<p>This method is appropriate when there is too much data to copy through the network. This is a “manual” materialization method that allows you to load the subscription data from media such as magnetic tape.</p> <p>Use this method for subscriptions to database replication definitions and to function replication definitions when data must be initialized at the replicate database.</p> <p>The commands used for bulk materialization are <code>define subscription</code>, <code>activate subscription</code>, and <code>validate subscription</code>. For more details, see “Bulk materialization” on page 357.</p>

## Atomic materialization

Atomic materialization is the default materialization method. It is the easiest method to execute and maintains complete data consistency throughout the materialization process.

During atomic materialization, Replication Server logs in to the primary data server as the user creating the subscription and with the password defined at the replicate Replication Server. Therefore, the user must be defined at both the replicate Replication Server and primary database with the same password. The user also needs the same login name and password as the primary Replication Server.

Logged in to the primary data server, the Replication Server selects the subscription rows using a `select with holdlock` operation specified by the `rs_select_with_lock` function. The holdlock performs a repeatable read, preventing other transactions at the primary site from updating the data until the select transaction has completed. The rows are transferred to a materialization queue at the replicate site, where they are applied to the replicate database. You must provide the stable queue with adequate partition space to handle the operation.

Atomic materialization is best for smaller subscriptions where the `select with holdlock` operation does not last long enough to disturb client applications using the primary database. If the subscription selects a large number of rows, you may choose to use nonatomic or bulk materialization, so that clients at the primary database are not affected.

When data already exists at the replicate database, you can use the non-materialization method.

Atomic materialization allows changes to the primary table but effectively delays data server changes until the activation phase of materialization has completed.



## Incremental atomic materialization

You can avoid long-running transactions at the replicate database by using the `incrementally` option. The `incremental` option sends materialization data to the replicate database in a series of transactions, rather than in one large transaction. Otherwise, incremental and non-incremental atomic materialization are identical. Subscription data is available but incomplete until materialization has completed and the subscription is validated.

Rows are removed from the stable queue after they have been successfully inserted, so less partition space is required. You can truncate the database transaction log during materialization, if necessary.

Users at the replicate site will see partial subscription data during materialization, which may invalidate some queries. However, they will have access to inserted rows sooner, which may be beneficial.

The `publishers_rep` replication definition presented in Chapter 9, “Managing Replicated Tables” is used in the following example to create a subscription. The `create subscription` command in the example has no `where` clause, so the subscription causes Replication Server to replicate all the rows in the replication definition. The `incrementally` keyword ensures that the replicate database transaction log does not become full. Clients at the replicate site can be suspended or warned that the `publishers` table is materializing and will contain incomplete data until the process has completed.

```
create subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
incrementally
```

## Nonatomic materialization

Nonatomic materialization, using the `without holdlock` option of the `create subscription` command, is the same as atomic materialization, except for the following:

- The data is selected from the primary database without a holdlock. Clients at the primary site can update the data while the `select` operation is in process.
- Transactions are always applied incrementally at the replicate database.

---

**Note** If the replicate minimal columns feature is set for the replication definition, you cannot create new subscriptions using nonatomic materialization.

---

In nonatomic materialization, Replication Server inserts rows into the replicate database incrementally in 10-row transactions. Clients at the replicate site that are using the table will see partial subscription data during materialization. This may invalidate some queries. Since the subscription is activated before the data is copied to the replicate database, primary table changes may be applied twice to the replicate table in some circumstances. You must enable autocorrection when you use nonatomic materialization. Autocorrection ensures that a second application of data does not result in an error. See “Using autocorrection” on page 356 for details.

## Using autocorrection

To enable autocorrection, issue the `set autocorrection` command with the `on` option for each replication definition to which you plan to subscribe using nonatomic materialization. When using autocorrection, if Replication Server updates or inserts a row in a primary table, it converts the update or insert into a delete followed by an insert, so that the update or insert operation cannot fail because of an existing row.

During nonatomic subscription materialization, Replication Server selects data without a holdlock. After adding the data to the replicate database, Replication Server applies replicated commands. If you enable autocorrection, Replication Server corrects certain temporary inconsistencies that may be caused by selecting the data using the `without holdlock` option.

However, if you execute replicated stored procedures that change subscription data during materialization, autocorrection does not always correct the replicate database. During function calls, autocorrection does not protect against inconsistencies.

After a subscription that uses nonatomic materialization has materialized, you can disable autocorrection for better performance. If you disable autocorrection, you can also specify minimal column replication. See “Replicating the minimal set of columns” on page 258 for more information.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about the `set autocorrection` command.

## No materialization

You can use `create subscription` with the `without materialization` clause to activate a subscription when materialization has already occurred. To use this method:

- The subscription data must already exist at the replicate database
- The primary and replicate tables must be synchronized
- Activity must be stopped on the primary table so that there are no further updates in the Replication Server stable queues

When creating a subscription with the `without materialization` clause, Replication Server logs in to the primary Replication Server as the user creating the subscription. The user who executes `create subscription` must have the same login and password at the primary and replicate Replication Servers.

You can also use `create subscription` with the `without materialization` clause to subscribe to function replication definitions.

## Bulk materialization

With bulk materialization, you manually transfer subscription data between databases. Use bulk materialization when a subscription is too large to copy through the network. Bulk materialization has very little effect on primary database clients or on the network.

You can use bulk materialization to create subscriptions for function replication definitions. See Chapter 10, “Managing Replicated Functions” for more information about replicated functions.

Bulk materialization uses these commands, which are executed at different points in the materialization process: `define subscription`, `activate subscription`, `validate subscription`. Use the `check subscription` command to check the status of the subscription.

When you use bulk materialization, you must coordinate:

- The dump to media of the subscription data at the primary site.
- The load from media into the table at the replicate site.

- The application of updates made at the primary site after you make the media dump.

---

**Note** Bulk materialization may require special handling if the primary and replicate databases differ in, for example, table or column names.

---

Three bulk-materialization methods are available to ensure data consistency between the primary and replicate sites. The method you use depends mainly on whether applications using the primary data can tolerate interruptions.

You can use any of these methods for subscriptions to either table or function replication definitions. With subscriptions to function replication definitions, it may not be obvious which replicate tables will be affected by stored replicated procedures executing in the replicate database.

Before you initiate bulk materialization, you must consider these issues in relation to the existing data in the replicate database.

Table 11-2 summarizes the three bulk materialization methods.

**Table 11-2: Summary of bulk materialization methods**

Method	Summary of process
Stop updates to the primary table and take a snapshot of the data	<p>Stop all applications from updating the primary data and then retrieve the subscription data from the primary database with a select statement or database dump.</p> <p>Define the subscription and activate it with an option that leaves the DSI suspended for the replicate database. Clients can resume updates to the primary data.</p> <p>After you load the subscription data into the replicate database, you can resume the DSI and validate the subscription. For details on this procedure, see “Stop updates at the primary database and take a snapshot” on page 359.</p>
Simulate atomic materialization	<p>Allow client applications to continue executing transactions against the primary data while the subscription data is retrieved. After defining the subscription, you lock the primary data, retrieve the subscription data, and activate the subscription. The activate subscription command leaves the DSI for the replicate database suspended.</p> <p>After you load the subscription data into the replicate database, you can resume the DSI and validate the subscription. For details on this procedure, see “Simulate atomic materialization” on page 361.</p>
Simulate nonatomic materialization	<p>This method is the same as simulating atomic materialization, except that you activate the subscription first, and then retrieve the data from the primary database without locking the data. Because of this, the data at the replicate database may be inconsistent with the data at the primary database until the subscription is validated and you are required to enable autocorrection for the replicate data. For details on this procedure, see “Simulate nonatomic materialization” on page 363.</p>

## Stop updates at the primary database and take a snapshot

To stop updates at the primary and take a snapshot, you can use either of two bulk materialization methods:

- Using the Adaptive Server mount command
- Using the Adaptive Server dump and load, select, or bcp command

Use these methods to retrieve data from the primary database if you are able to suspend updates to the primary data. To maintain consistency, all updates to the primary database are suspended for the duration of the materialization.

### ❖ **Retrieving data from the primary database using the Adaptive Server mount command**

This procedure uses mount to retrieve data from the primary database. You can use this method only if you are using Adaptive Server version 12.5.1 or later, and your primary and replicate databases are identical.

- 1 Verify that the entire replication system is working. See Chapter 1, “Verifying and Monitoring Replication Server” in the *Replication Server Administration Guide Volume 2* for details.
- 2 Suspend updates to the data in the primary database by stopping client applications that generate transactions against the primary data directly or indirectly through Replication Servers.
- 3 Quiesce the replication system components involved with replicating data from the primary Replication Server to the replicate Replication Server.  
Use `admin quiesce_for_rsi` at the primary and replicate Replication Servers and any intermediate Replication Servers.
- 4 Execute the Adaptive Server command `quiesce database tag_name hold db_name list [for external dump] to manifest_file [with override]` to generate the manifest file. See the *Adaptive Server Enterprise Reference Manual* for more information.
- 5 Take a snapshot of the subscription data from the primary database by creating a data dump of both the database and log devices. You can create a data dump using utilities such as tar or zip, or the UNIX dd command.
- 6 Use `mount database` to begin loading the snapshot data into the replicate database.
- 7 Resolve the mismatch of user information between the master database and the loaded user database.

- 8 Use `rs_init` to add the replicate database to the replication system if it is not already there.
- 9 Execute `define subscription` at the replicate Replication Server.
- 10 Use `check subscription` at the primary and at the replicate Replication Servers to verify that the subscription has been defined. When the subscription status is `DEFINED` at both servers, continue to step 11.
- 11 Execute `activate subscription` at the replicate Replication Server.
- 12 Use `check subscription` at the primary and at the replicate Replication Server to verify that the subscription has been activated. When the subscription status is `ACTIVE` at both servers, continue to step 13.
- 13 Execute `quiesce release` to resume updates to the primary data.
- 14 Execute `validate subscription` at the replicate Replication Server.
- 15 Use `check subscription` at the primary and at the replicate Replication Server to verify that the subscription is `VALID` at both servers.

When you have completed this procedure, the subscription is created, the replicate data is consistent with the primary data, and replication is active.

❖ **Retrieving data from the primary database using the Adaptive Server dump and load, select or bcp commands**

This procedure retrieves data from the primary database using the Adaptive Server dump and load, select, or bcp commands and utilities.

- 1 Verify that the entire replication system is working. Refer to Chapter 1, “Verifying and Monitoring Replication Server” in the *Replication Server Administration Guide Volume 2* for details.
- 2 Suspend updates to the data in the primary database by stopping client applications that generate transactions against the primary data.
- 3 Quiesce the replication system components involved with replicating data from the primary Replication Server to the replicate Replication Server.  
  
Use `admin quiesce_force_rsi` at the primary and replicate Replication Servers and at any intermediate Replication Servers.
- 4 Execute `suspend log transfer` for the primary database.
- 5 Take a snapshot of the subscription data from the primary database using a `select` statement or a database dump.
- 6 Execute `define subscription` at the replicate Replication Server.

- 7 Use check subscription at the primary and at the replicate Replication Servers to verify that the subscription has been defined. When the subscription status is DEFINED at both servers, continue to step 9.
- 8 Execute the activate subscription command, using the with suspension clause, at the replicate Replication Server.
- 9 Use check subscription at the primary and at the replicate Replication Server to verify that the subscription has been activated. When the subscription becomes active at the replicate Replication Server, the DSI connection to the replicate Replication Server is suspended.  
  
When the subscription status is ACTIVE at both servers, continue to step 11.
- 10 Execute resume log transfer from the primary database at the primary Replication Server.
- 11 Begin loading the snapshot data into the replicate database.

---

**Note** While you wait for the data to finish loading in the replicate database, you can continue with the next step.

---

- 12 Execute validate subscription at the replicate Replication Server to validate the subscription.
- 13 Use check subscription at the primary and at the replicate Replication Server to verify that the subscription status is VALID for both servers.
- 14 When the snapshot data has finished loading in the replicate database, execute resume connection to resume the connection to the replicate database.

When you have completed this procedure, the subscription is created, the replicate data is consistent with the primary data, and replication is active.

## Simulate atomic materialization

Use this bulk materialization method when you cannot suspend updates to the primary database.

This method ensures replicated data consistency by retrieving the subscription data, activating the subscription, and suspending the DSI connection to the replicate database all in one transaction at the primary data server.

Use select with holdlock and the rs\_marker stored procedure, as in this example:

```
begin transaction
select from table with holdlock
where search_conditions
execute rs_marker
'activate subscription subid
with suspension'
commit transaction
```

*subid* is an integer that identifies the subscription. The *subid* for a subscription can be found in the *subid* field of the *rs\_subscriptions* system table in the RSSD. After the subscription is defined, you can find its *subid* by executing the following query in the RSSD of the primary or replicate Replication Server:

```
select subid from rs_subscriptions
where subname = 'subscription'
and dbid in (select dbid from rs_databases
where dbname = 'replicate_database'
and dsname = 'replicate_data_server')
```

Here are the steps to follow to simulate atomic materialization:

- 1 Verify that the entire replication system is working. Refer to Chapter 1, “Verifying and Monitoring Replication Server” in the *Replication Server Administration Guide Volume 2* for details.
- 2 Execute the define subscription command at the replicate Replication Server.
- 3 Wait for the subscription to be defined 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 DEFINED.
- 4 Execute a single transaction as provided in the previous sample transaction that includes select with holdlock and the *rs\_marker* stored procedure. This action activates the subscription.
- 5 Wait for the subscription to become active at both the primary and replicate Replication Servers. Execute the check subscription command at the replicate Replication Server to verify that the subscription status is ACTIVE. When the subscription status is ACTIVE at the replicate Replication Server, the DSI connection to the replicate database will be suspended.
- 6 Begin loading the subscription data into the replicate database.
- 7 Resume the DSI connection to the replicate database using the resume connection command.



- 8 Execute the `validate subscription` command at the replicate Replication Server.
- 9 Wait for the subscription to become valid at both the primary and replicate Replication Server. Execute the `check subscription` command at the replicate Replication Server to verify that the subscription status is `VALID`.

Now the subscription is created and replication is active.

## Simulate nonatomic materialization

Use this bulk materialization method when you cannot suspend updates to the primary database or if you cannot lock the primary data during the `select` or `dump` operation that retrieves the subscription data.

This method allows a period of flux at the replicate site during which the replicate data may be inconsistent with the primary data. By the time the subscription becomes `VALID`, however, the data should be consistent. You must set autocorrection on during materialization so that inconsistencies resulting from continuing updates in the primary database can be resolved without errors.

---

**Warning!** Do not use this method if the replicate minimal columns feature is set for the replication definition or if you execute applied functions or applied stored procedures from the primary database to modify data in the replicate database. In both cases, autocorrection cannot resolve the inconsistencies.

---

- 1 Verify that the entire replication system is working. Refer to Chapter 1, “Verifying and Monitoring Replication Server” in the *Replication Server Administration Guide Volume 2* for details.
- 2 Execute the `define subscription` command at the replicate Replication Server.
- 3 Wait for the subscription to be defined 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 `DEFINED`.
- 4 Execute the `activate subscription` command, using the `with suspension` clause, at the replicate Replication Server.

- 5 Wait for the subscription to become active at both the primary and replicate Replication Servers. Execute the check subscription command at the replicate Replication Server 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 has been suspended.
- 6 As soon as the subscription becomes active at the primary Replication Server, retrieve the data from the primary database using a select or a database dump.
- 7 Find the ID number (*subid*) for the subscription by querying the rs\_subscriptions system table. See “Subscription example” on page 382 for more information.
- 8 Execute the rs\_marker stored procedure in the primary database:

```
rs_marker 'validate subscription subid'
```

---

**Warning!** Be sure that you execute the rs\_marker stored procedure with the correct subid number for the subscription. The subid column in the rs\_subscriptions system table contains the unique ID number for each subscription. Entering any other number or character string may cause serious problems.

For more information on rs\_marker see *Replication Server Reference Manual*.

---

- 9 Load the subscription data into the replicate database.
- 10 Enable autocorrection for the replication definition at the replicate database. See “Using autocorrection” on page 356 for more information.
- 11 Use the resume connection command to resume the database connection for the replicate database.
- 12 Wait for the subscription to become valid at both the primary and replicate Replication Servers. Execute the check subscription command at the replicate Replication Server to verify that the subscription status is VALID. Once the subscription status is VALID, the replicate data is consistent with the primary data.
- 13 Disable autocorrection for the replicate database. See “Using autocorrection” on page 356 for more information.

Now the subscription is created and replication is active.

## Dematerialization processing

Dematerialization removes subscriptions and, optionally, data from the replicate database. Dematerialization also removes subscription information from the RSSDs at the primary and replicate sites.

Dropping a subscription causes Replication Server to stop sending changes from a primary database to a replicate database. You can use the drop subscription command to drop subscriptions for either table or function replication definitions.

drop subscription removes the subscription from the RSSDs of the primary and replicate Replication Servers.

When you drop a subscription to a table replication definition, you can specify that Replication Server delete the subscription's rows from the replicate database. Or, you can delete the rows manually.

When you drop a subscription to a function replication definition, the replicate data associated with the function is not deleted from the replicate database.

There are two methods of dematerialization:

- with purge dematerialization, which selectively deletes rows not used by other subscriptions
- without purge dematerialization, which allows you to manually delete rows in replicate tables

In either case, the primary Replication Server stops sending data for the dropped subscription, if the data is not included in other subscriptions at the same replicate site.

---

**Note** For heterogeneous datatypes: Subscriptions that specify columns subject to class- or column-level translations in the where clause cannot be dematerialized automatically. You must use the bulk or no-materialization method.

---

## Dematerializing and purging rows

Use the with purge clause when you want to delete rows replicated by the subscriptions you are dropping. Use the incrementally option to delete rows in 10-row increments. The maintenance user for the replicate database must have select permission on the table to use this option.

Dematerializing a subscription and purging rows from the replicate table uses function strings for the `rs_select` or `rs_select_with_lock` system functions. You may be required to create a function string for these system functions.

- If the connection for the replicate database uses a function-string class with default-generated function strings or a function-string class inherited from such a class, Replication Server generates a corresponding default function string for the `rs_select_with_lock` or `rs_select` functions.
- If the connection uses any other function-string class, you must create the function string, with an input template that matches the subscription's where clause. Use the create function string command.

See “Function-string classes” on page 21 in the *Replication Server Administration Guide Volume 2* for details.

If you are using a function-string class in which you can customize function strings, you can replace an existing default or custom function string with one that performs a select operation that your application requires, using the alter function string command.

For more information on creating or altering `rs_select` and `rs_select_with_lock` function strings, see “Managing function strings” on page 32 in the *Replication Server Administration Guide Volume 2*.

## Dematerialization without purging rows

Dropping a subscription using the `without purge` option leaves the rows replicated by the subscription in the replicate table. Subscriptions to function replication definitions are dropped automatically using the `without purge` option. You do not need to specify this option. You must, however, specify this option if you want to keep the rows in the replicate table. If you want to manually delete rows, you must use the `with suspension` option as well.

## Monitoring materialization and dematerialization

Subscriptions pass through phases before they are fully set up or removed from the replication system. The phases for setting up a subscription are:

- *Definition* – create subscription or define subscription add the subscription to the RSSD for the primary and replicate Replication Servers.

- *Activation* – takes place after subscription resolution. The primary Replication Server adds the subscription to the Subscription Resolution Engine (SRE). The SRE compares log records to the current subscriptions to determine where changes to replicated tables must be distributed.
- *Materialization* – for atomic and nonatomic subscriptions, the primary Replication Server retrieves subscription data from the primary database and copies it to the replicate Replication Server to be applied to the replicate database.
- *Validation* – both the primary and replicate Replication Server completely materialize the subscription and verify it is consistent with the primary data.

The phases for removing subscription data, using the drop subscription command, are:

- *Dematerialization* – stops sending updates for the subscription to the replicate database and, if the with purge clause is specified, deletes the subscription data from the replicate database (if the data is not included in other subscriptions). If the without purge clause is specified, then Replication Server does not delete the data from the replicate database.
- *Removal* – deletes the subscription from the RSSD for both the primary and replicate Replication Servers.

Materialization or dematerialization can fail during any of these stages. This is why you need to monitor the progress of a subscription using the check subscription command. See “Using the check subscription command” on page 379 for more information. In addition to the check subscription command, you can use the admin who command to check the status of the Replication Server threads processing the subscription. For atomic and nonatomic materialization, Replication Server builds a materialization queue that contains rows to be added to the replicate table. The admin who, sqm command can monitor queue activity, and the admin who, dsi command can show you whether the DSI thread is running.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for information about executing admin who and interpreting its results.

Refer to the *Replication Server Troubleshooting Guide* for comprehensive troubleshooting information that details the status of a subscription and suggested actions.

## Before you create subscriptions

Before creating subscriptions, verify that the replication system is ready. Review each of the steps in this section that follow to ensure that you meet all requirements.

- 1 Verify that all components in the replication system are working. See “Verifying a replication system” on page 2 in the *Replication Server Administration Guide Volume 2* for details.
- 2 Make sure the following database objects and permissions exist:
  - One or more replication definitions exist for the primary table.
  - The primary table is marked as replicated with `sp_setreptable` or `sp_reptostandby` for warm standby applications.
  - A table corresponding to the replication definition exists in the replicate database. Its columns must match those specified for the replicate database in the replication definition. Its datatypes must match the corresponding primary columns.

This table must also be visible to the user creating the subscription and the user maintaining it. If an owner name is included in the replication definition, the table must be visible to all database users. If an owner name is not included in the replication definition, the easiest way to make the table accessible is to have the Database Owner create it.

- The replicate database maintenance user must have:  
select, insert, update, and delete permissions on the replicate table, and execute permission for functions used in replication.  
  
If the subscription for the table includes the subscribe to truncate table clause, the maintenance user must have `replication_role`, `sa_role`, or alias the Database Owner.
- 3 Make sure that you meet recommended guidelines for the character sets and sort orders used throughout your replication system. These play an important role in processing subscriptions, and they must be consistent everywhere for subscriptions to be valid. Refer to the *Replication Server Design Guide* for guidelines.
  - 4 Choose one of the subscription materialization methods described in “Subscription materialization methods” on page 353, and verify the following requirements for your chosen method:

- For nonatomic materialization, you must enable autocorrection for the replicate table. See “Using autocorrection” on page 356 for more information. Also refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for set autocorrection command details.

If the replicate minimal columns feature is set for the replication definition, you *cannot* create new subscriptions using nonatomic materialization.

- For atomic and nonatomic materialization:

A default function-string class or a function-string class inherited from a default function-string class generates default function strings for the `rs_select_with_lock` or `rs_select` functions. If you use other function-string classes, you must create function strings for the `rs_select_with_lock` or `rs_select` functions, with an input template that matches the subscription’s where clause.

To modify `rs_select` or `rs_select_with_lock`, use a function from the function string class associated with the primary database connection, not the functions in the replicate database connection.

See “Function-string classes” on page 21 and “Using input templates” on page 35 in the *Replication Server Administration Guide Volume 2* for details.

- 5 When you create subscriptions, use the login name of a regular user. Do not create subscriptions as the maintenance user.

Make sure the user creating the subscription has the following login names and permissions:

- Same login name and password at the replicate Replication Server, the primary Replication Server, and the primary data server. If you are using bulk materialization or the no-materialization method, you are not required to have a login name for the primary data server.
- select permission on the primary table. This does not apply if you are using bulk materialization or no materialization.
- execute permission on the `rs_marker` stored procedure in the primary database or no materialization.
- create object or sa permission in the replicate Replication Server.
- primary subscribe, create object, or sa permission in the primary Replication Server.

## Using subscription commands

You can use RCL commands or Sybase Central to:

- Create subscriptions for atomic and nonatomic materialization and for the no-materialization method.
- Define, activate, and validate subscriptions for bulk materialization.
- Check the status of subscriptions during the materialization process.
- Drop subscriptions to initiate the dematerialization process.
- Enable replication of the truncate table command when you create or define a subscription.

You can use a where clause to control which table rows or function invocations to replicate. The where clause can specify only the searchable columns or searchable parameters specified in the table or function replication definition. If you do not provide a where clause, all the rows of the replication definition's columns, or all the function invocations, are replicated. See “Using the where clause” on page 371 for more information.

If you are using Adaptive Server Enterprise version 11.5 or later, you can include the subscribe to truncate table keywords to reproduce execution of the truncate table command at the destination database. See “Enabling replication of truncate table” on page 373 for more information.

Table 11-3 lists the Replication Server commands for working with subscriptions. Also see Table 9-1 on page 249 and Table 10-1 on page 333.

**Table 11-3: Commands for managing subscriptions**

Command	Task
create subscription	<p>Creates a subscription that transfers the initial version of the replicated data using either:</p> <ul style="list-style-type: none"> <li>• Atomic materialization, which copies the initial version of the data for a subscription as a single transaction, or</li> <li>• Nonatomic materialization, which copies the data in a series of transactions. Users at the replicate site can see some of the data before it all arrives. Replication Server does not create a materialization queue for the entire set of subscription data.</li> </ul> <p>Use create subscription with the without materialization clause to activate a subscription for which the initial version of the replicated data already exists at the replicate database. You can also use create subscription to create subscriptions for table replication definitions. Use create subscription, with the without materialization clause, for function replication definitions.</p>



Command	Task
define subscription	The first step in bulk materialization defines a subscription. You can use <code>define subscription</code> and the other bulk materialization commands to create subscriptions for either table or function replication definitions. You must transfer data manually, as necessary. Data replication begins after materialization is complete and a subscription is activated and validated. Use <code>check subscription</code> to verify subscription status. See “Using the check subscription command” on page 379 for details. See Chapter 10, “Managing Replicated Functions.”
activate subscription	Second step in bulk materialization. Activates a subscription at both primary and replicate Replication Servers. This causes the primary Replication Server to start sending changes to the subscription’s data to the replicate Replication Server. See “Using the activate subscription command” on page 378 for details.
validate subscription	Third step in bulk materialization. Changes the subscription status at both the primary and replicate sites to <code>VALID</code> . See “Using the validate subscription command” on page 379 for details.
check subscription	Verifies the status of a subscription at both the primary and replicate sites. Use this command with all types of subscription materialization. See “Using the check subscription command” on page 379 for details.
drop subscription	Removes a subscription from the replication system. For subscriptions to table replication definitions, optionally removes subscription rows from the replicate table in a process known as dematerialization. See “Using the drop subscription command” on page 380 for details.

## Using the *where* clause

You can include one *where* clause in a subscription. The *where* clause syntax is a subset of the Transact-SQL *where* clause. It is supported by the `create subscription` and `define subscription` commands for subscriptions to replication definitions. The supported syntax is the same for both commands and allows you to create very selective subscriptions. It is designed for efficient processing by the Subscription Resolution Engine in Replication Server.

---

**Note** You cannot evaluate a Java column in a subscription expression. Thus, you cannot include a column of type `rawobject` or `rawobject in row` in a subscription *where* clause.

---

For subscriptions to table replication definitions, the *where* clause syntax is:

```
where column_name{< | > | <= | >= | = | &} value
[and column_name{< | > | <= | >= | = | &}
value]...
```

For subscriptions to function replication definitions, the *where* clause syntax is:

```
where @param_name
      {< | > | <= | >= | = | &} value
[and @param_name
     {< | > | <= | >= | = | &} value]...
```

Refer to “Datatypes” in Chapter 2, “Topics,” in the *Replication Server Reference Manual* for entry formats for values for different datatypes.

---

**Note** The !=, !<, !>, and or operators are not supported. You can create multiple subscriptions instead of using the or operator. The & operator is supported only on `rs_address` columns. For details on using the `rs_address` datatype, see “Using the `rs_address` datatype” on page 291 and “Bitmap subscriptions” on page 388.

---

Each column name in a where clause must be listed in the searchable columns list of the table or function replication definition. The *value* for each column must have the same datatype as the column to which it is compared.

For example, for table replication definition `publishers_rep`, you would enter:

```
create subscription publishers_sub1
for publishers_rep
with replicate at SYDNEY_DS.pubs2
where state = 'CA'
```

to specify that you want to subscribe to data where `state = CA`.

---

**Note** The maximum size of a where clause in a create subscription statement is 255 characters.

---

To subscribe to data in `publishers`, where `state = CA` or `state = MA`, you would need to create two subscriptions. In addition to the preceding command, you would enter:

```
create subscription publishers_sub2
for publishers_rep
with replicate at SYDNEY_DS.pubs2
where state = 'MA'
```

---

**Note** When you use a where clause with a subscription for heterogeneous datatype columns subject to class- or column-level translations, you must make sure that you use the correct datatype in the comparison. See “Subscriptions for columns with heterogeneous datatypes” on page 387.

---

## Enabling replication of *truncate table*

If you are using Adaptive Server Enterprise version 11.5 or later, you can enable replication of the `truncate table` command to particular destination database tables when you create or define a subscription.

The `truncate table` command can truncate one or more partitions. Replication Server will recreate the same command executed at the primary database. This requires the replicate site to have the same partition names, otherwise, DSI shuts down.

You have an option to skip the `truncate table` command and apply appropriate action at the replicate site, or use `rs_truncate` function string to customize the action in the replicate site. Replication Agent sends this command once the LTL version is set to 700.

To create or define a subscription that enables replication of `truncate table`, log in to Replication Server and enter:

```
create subscription subscription
  for table_rep_def
  with replicate at data_server.database
  ...
  subscribe to truncate table
```

When `truncate table` executes at the destination database, Adaptive Server deallocates whole data pages. It does not delete rows one at a time.

---

**Note** Replication Server executes `truncate table` at the replicate database as the maintenance user. Among the permissions granted to maintenance user is `replication_role`. If you revoke maintenance user's `replication_role`, you cannot replicate `truncate table` unless the maintenance user has been granted `sa_role`, the maintenance user owns the table, or the maintenance user is aliased as the Database Owner.

---

Warm standby applications can copy the execution of `truncate table` to standby databases without a subscription. See “Replicating `truncate table` to standby databases” on page 101 in the *Replication Server Administration Guide Volume 2* for information about using this feature.

See `define subscription` and `create subscription` in Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for complete command syntax and usage guidelines.

### Changing the Status of “subscribe to truncate table”

All subscriptions for a replicate table in a particular database must either support or not support replication of truncate table. You cannot create a subscription that enables replication of truncate table if all existing subscriptions for that table do not support replication of truncate table.

Use the `sysadmin apply_truncate_table` command to change the status of “subscribe to truncate table” for all subscriptions on a replicate table.

For example, to turn on replication of truncate table for all subscriptions to a replicate table, log in to the replicate Replication Server and execute this command at the `isql` prompt:

```
sysadmin apply_truncate_table, data_server,  
database, {table_owner|' ' |""}, table_name'on'
```

where `data_server` is the name of the replicate data server, `database` is the name of the replicate database managed by the data server, `table_owner` is the owner of the replicate table, and `table_name` is the name of the replicate table.

If you specified a replicate table owner in the replication definition, you must also specify a table owner with the `sysadmin apply_truncate_table` command. If you did not specify a replicate table owner in the replication definition, enter " (two single-quote characters) or "" (two double-quote characters) for the table owner name.

Refer to Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for more information about `sysadmin apply_truncate_table` command.

## Using the *create subscription* command

You use the `create subscription` command to replicate data by subscribing to a replication definition. There are three methods for creating a subscription:

- Atomic
- Nonatomic
- No materialization

You can use a `where` clause to replicate only certain rows from the primary table, based on values for the searchable columns specified in the table replication definition. If you do not provide a `where` clause, all rows are replicated. See “Using the `where` clause” on page 371 for more information.

If you are using Adaptive Server Enterprise version 11.5 or later, you can include the `subscribe to truncate table` keywords to reproduce execution of the `truncate table` command at the destination database. See “Enabling replication of truncate table” on page 373 for more information.

---

**Note** `create subscription` automatically truncates text, untext, and image data larger than 32K.

---

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for details on using the `create subscription` command. See Chapter 9, “Managing Replicated Tables” for more information on creating table replication definitions.

### Using *create subscription* for atomic materialization

To create a subscription with atomic materialization, execute the `create subscription` command at the Replication Server managing the database where the data is to be replicated. The syntax for the `create subscription` command, with atomic materialization, is:

```
create subscription subscription
  for table_rep_def
  with replicate at data_server.database
  [where search_conditions]
  [incrementally]
  [subscribe to truncate table]
```

where *subscription* is the name of the subscription to activate, *table\_rep\_def* is the name of the table replication definition you are subscribing to, and *data\_server.database* identifies the replicate database.

The *subscription* name must be unique for the replication definition and replicate database.

Subscribing to function replication definitions requires you to use `define subscription` (the bulk materialization method) or `create subscription` with the `without materialization` clause (the no materialization method).

If you use the optional keyword `incrementally`, Replication Server initializes the subscription by sending 10-row batches of inserts.

If you do not use the keyword `incrementally`, Replication Server inserts all of the subscription rows at the replicate database in a single transaction. All of the rows are held in a stable queue at the replicate Replication Server at one time, and there must be enough partition space to accommodate them. Also, the transaction log for the replicate database must have enough space to log the transaction.

## Using `create subscription` for nonatomic materialization

Use the `create subscription` command with the `without holdlock` clause to create a subscription with nonatomic materialization. The syntax is:

```
create subscription subscription
  for table_rep_def
  with replicate at data_server.database
  [where search_conditions]
  without holdlock
  [subscribe to truncate table]
```

where *subscription* is the name of the subscription to activate, *table\_rep\_def* is the name of the table replication definition you are subscribing to, and *data\_server.database* identifies the replicate database.

Nonatomic materialization is always incremental.

Clients at the replicate site should be suspended or warned that the data in the replicate table is incomplete and possibly inconsistent until all the subscription data has materialized.

See “Monitoring materialization and dematerialization” on page 366 for information about monitoring the materialization process.

## Using `create subscription` for no materialization

To create a subscription that does not initialize the subscription data, execute `create subscription` with the `without materialization` clause at the Replication Server managing the replicate database. The syntax for `create subscription` for no materialization is:

```
create subscription subscription
  for {table_rep_def | function_rep_def | publication pub |
      database replication definition db_repdef
      with primary at server_name.db }
  with replicate at server_name.db
  [where search_conditions]
  without materialization
  [subscribe to truncate table]
```

where *subscription* is the name of the subscription to create, *table\_rep\_def* is the name of the table replication definition the subscription is for, *function\_rep\_def* is the name of the function replication definition the subscription is for, *pub* is the name of the publication the subscription is for, *db\_repdef* is the name of the database replication definition the subscription is for, and *server\_name.db* identifies the primary or replicate database.

The without materialization clause activates the subscription without first initializing the subscription data. Use create subscription with the without materialization clause when there is no activity at the primary database and the data already exists in the replicate database.

## Using the *define subscription* command

To create a subscription with bulk materialization, execute the define subscription command at the Replication Server that is managing the database where the data is to be replicated. define subscription sets the subscription status to DEFINED.

The syntax for define subscription is:

```
define subscription subscription
for {table_rep_def | function_rep_def
     publication pub_name | database replication definition db_repdef
     with primary at data_server.db
with replicate at data_server.db
[where search_conditions]
[subscribe to truncate table]
```

where *subscription* is the name of the subscription to define, *table\_rep\_def* is the name of the table replication definition the subscription is for, *function\_rep\_def* is the name of the function replication definition the subscription is for, *pub\_name* is the publication the subscription is for, *db\_repdef* is the database replication definition the subscription is for, and *data\_server.db* identifies the primary or replicate database.

The *subscription* name must be unique for the replication definition and replicate database.

Refer to Chapter 9, “Managing Replicated Tables” for more information on creating table replication definitions. Refer to Chapter 10, “Managing Replicated Functions” for more information on creating function replication definitions. Also refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for details on using define subscription command.

## Using the *activate subscription* command

Use the activate subscription command during bulk materialization to start the distribution of updates from the primary to the replicate database for a subscription. activate subscription sets the subscription status to ACTIVE.

Execute active subscription at the Replication Server where you created the subscription using the define subscription command. The syntax for activate subscription is:

```
activate subscription subscription
  for { table_rep_def | function_rep_def | publication pub_name |
        database replication definition db_repdef
        with primary at data_server.db }
  with replicate at data_server.db
  [with suspension [at active replicate only]]
```

where *subscription* is the name of the subscription to activate, *table\_rep\_def* is the name of the table replication definition the subscription is for, *function\_rep\_def* is the name of the function replication definition the subscription is for, *pub\_name* is the publication the subscription is for, *db\_repdef* is the database replication definition the subscription is for, and *data\_server.db* identifies the primary or replicate database.

Use the with suspension clause to suspend the DSI after the subscription status changes to ACTIVE. This prevents the replicate Replication Server from sending updates for the replicated table before the subscription data is loaded. After loading the data at the replicate site, execute resume connection to apply the updates.

If you do not use with suspension, you should prohibit updates to the primary table until the subscription is materialized.

If the database is part of a warm standby application, the with suspension clause suspends the DSI for the active and standby databases. This let you load the data into both databases before allowing updates to the active database. If you load the data into the active database with logging, use the with suspension at active replicate only clause so that the standby DSI remains active. In this case, subscription data is replicated from the active database. The DSI for the active database in a warm standby application is suspended. The clause does not suspend the DSI for the standby database.

Refer to “Using the validate subscription command” on page 379 for more information about the with suspension and with suspension at active replicate only clauses. Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for detailed usage information for activate subscription.



## Using the *validate subscription* command

Use the `validate subscription` command to complete the bulk materialization process and set the subscription status to `VALID`.

Execute `validate subscription` at the Replication Server where you created the subscription. The syntax is:

```
validate subscription subscription
for { table_rep_def | function_rep_def | publication pub_name |
      database replication definition db_repdef
      with primary at data_server.db }
with replicate at data_server.db
```

where *subscription* is the name of the subscription to validate, *table\_rep\_def* is the name of the table replication definition the subscription is for, *function\_rep\_def* is the name of the function replication definition the subscription is for, *pub\_name* is the publication the subscription is for, *db\_repdef* is the database replication definition the subscription is for, and *data\_server.db* identifies the primary or replicate database.

## Using the *check subscription* command

The `check subscription` command reports the status of a subscription at the Replication Server where you enter the command. The subscription status at the primary and replicate Replication Servers often differs while the subscription is being created, so you should enter `check subscription` at both sites. If the primary and replicate databases are managed by a single Replication Server, `check subscription` displays the status of the subscription for both the primary and replicate databases.

The syntax for the `check subscription` command is:

```
check subscription subscription
for { table_rep_def | function_rep_def | publication pub_name |
      database replication definition db_repdef
      with primary at data_server.db }
with replicate at data_server.db
```

where *subscription* is the name of the subscription to check, *table\_rep\_def* is the name of the table replication definition the subscription is for, *function\_rep\_def* is the name of the function replication definition the subscription is for, *pub\_name* is the publication the subscription is for, *db\_repdef* is the database replication definition the subscription is for, and *data\_server.db* identifies the primary or replicate database.

The message returned by the command contains subscription status information. If the subscription had an error, the message directs you to the log where you should look for specific error messages.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for a list of the messages check subscription can return.

## Using the *drop subscription* command

Dropping a subscription causes Replication Server to stop sending changes from a primary database to a replicate database. You can use the drop subscription command to drop subscriptions for either table or function replication definitions.

Execute the drop subscription command at the replicate Replication Server. It requires create object permission at the replicate Replication Server and create object or primary subscribe permission at the primary Replication Server.

Here is the syntax:

```
drop subscription subscription
for {table_rep_def | function_rep_def | article article_name in pub_name |
     publication pub_name | database replication definition db_repdef
     with primary at data_server.db }
with replicate at data_server.database
[without purge
[with suspension [at active replicate only ] ] |
[incrementally] with purge]
```

If you choose the without purge dematerialization method, Replication Server does not delete subscription data from the replicate database.

If you choose the with purge dematerialization method, Replication Server logs in to the replicate database and selects data from it. If this data does not belong to any other subscriptions, the subscription data is deleted from the replicate database.

When you drop subscriptions to table replication definitions, you can purge subscription rows regardless of the materialization method you used when you created the subscription. Rows are removed only if they do not match another subscription.

You can use the check subscription command to view the progress of the drop subscription command. When the subscription status no longer exists at the primary and replicate Replication Servers, the command is complete.

Subscriptions to function replication definitions are always dropped without purging the replicate data associated with the function. You do not need to specify the `without purge` option.

When you are dropping subscriptions to table replication definitions, you have two basic methods to choose from. Because each method carries important implications, Replication Server requires that you explicitly choose one of these two methods:

- `with purge` – Replication Server removes, or dematerializes, the subscription's rows from the replicate database, if they do not belong in other remaining subscriptions. The Replication Server logs in as the maintenance user to perform the select operation. Use the `incrementally` option to specify that dematerialization occurs in 10-row increments of deletes per transaction.
- `without purge` – the subscription's rows remain at the replicate database. The `with suspension` option leaves the connection to the replicate database suspended when drop subscription has completed, so that you can manually remove the rows.

For warm standby applications, the option `with suspension at active replicate` only suspends the active replicate database but not the standby replicate database.

---

**Warning!** When removing rows manually, do not remove rows for remaining overlapping subscriptions that require those rows.

---

Example of dropping subscription with purge

To drop a subscription with purge, use a command like this:

```
drop subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
with purge
```

Examples of dropping subscription without purge

To drop a subscription without purge, use a command like this:

```
drop subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
without purge
```

To drop a subscription without purge and also suspend the DSI for the replicate database so that you can manually delete the rows for the subscription, use a command like this:

```
drop subscription publishers_sub
```

```
for publishers_rep
with replicate at SYDNEY_DS.pubs2
without purge
with suspension
```

If you have a warm standby application for the replicate database, you may want to suspend the connection for the active database only, and leave the standby DSI up. This way, Replication Server will replicate your row deletion transactions from the active replicate database to the standby database. In this case, use a command like this:

```
drop subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
without purge
with suspension at active replicate only
```

## Subscription example

This section contains an example that shows you how to replicate a table from a primary database to a replicate database by creating an atomic subscription to the table replication definition. You can do this in Sybase Central or with RCL commands. The example shows you the steps and RCL commands needed to replicate transactions for a table named publishers between two Adaptive Servers.

Following is a description of the replication system and procedures for setting up replication for the table.

### Description of replication system

#### Primary site

- The Replication Server for the primary site is named TOKYO\_RS.
- The primary version of the publishers table is in the pubs2 database of the Adaptive Server named TOKYO\_DS. You have added a connection from TOKYO\_RS to the pubs2 database using Sybase Central or rs\_init and set up a RepAgent for the database.
- The system database for TOKYO\_RS is named TOKYO\_RSSD and is managed by the TOKYO\_DS Adaptive Server.
- A route exists from TOKYO\_RS to SYDNEY\_RS.

- Replicate site
- The Replication Server for the replicate site is named SYDNEY\_RS.
  - The replicate copy of the publishers table will be in the pubs2 database of the Adaptive Server named SYDNEY\_DS. You have added a connection from SYDNEY\_RS to the pubs2 database using Sybase Central or rs\_init.
  - The system database for SYDNEY\_RS is named SYDNEY\_RSSD and is managed by the SYDNEY\_DS Adaptive Server.

## Procedures for replicating tables

Preparing to replicate tables      To check replication system components, use Sybase Central or isql to log in to the servers identified for the primary and replicate sites.

Preparing the primary table      In the TOKYO\_DS Adaptive Server, log in to the pubs2 database and ensure that the publishers table exists:

```
isql -Usa -P -STOKYO_DS
use pubs2
go
sp_help publishers
go
```

Preparing login names for user creating the subscription      You will create the subscription using the “pubs2\_user” login name. This user must exist in both Replication Servers.

In the TOKYO\_DS Adaptive Server, create this login name:

```
isql -Usa -P -STOKYO_DS
sp_addlogin pubs2_user, pubs2_pw, pubs2
go
```

In the TOKYO\_DS Adaptive Server, add the “pubs2\_user” login name to the pubs2 database, and grant the user select permission on the publishers table:

```
use pubs2
go
sp_adduser pubs2_user
go
grant select on publishers to pubs2_user
go
```

In the TOKYO\_RS Replication Server, create the “pubs2\_user” login name and grant primary subscribe permission to this login name:

```
isql -Usa -P -STOKYO_RS
create user pubs2_user
set password pubs2_pw
```

```
go
grant primary subscribe to pubs2_user
go
```

In the SYDNEY\_RS Replication Server, create the “pubs2\_user” login name and grant create object permission to this login name:

```
isql -Usa -P -SSYDNEY_RS
create user pubs2_user
set password pubs2_pw
go
grant create object to pubs2_user
go
```

Creating the replication definition

In the TOKYO\_RS Replication Server, create the replication definition publishers\_rep for the publishers table:

```
isql -Ujohn -P -STOKYO_RS
create replication definition publishers_rep
with primary at TOKYO_DS.pubs2
with all tables named 'publishers'
(pub_id char(4), pub_name varchar(40),
city varchar(20), state char(2))
primary key (pub_id)
searchable columns (pub_id, pub_name)
replicate minimal columns
go
```

In this example, the user “john” creates the replication definition. This user requires create object permission in TOKYO\_RS.

Marking the primary table for replication

In the TOKYO\_DS Adaptive Server, mark the publishers table for replication. To mark the table for replication with the sp\_setreptable system procedure, you must be the Database Owner or System Administrator for the data server. Enter the following command:

```
sp_setreptable publishers, 'true'
go
```

Verifying that the table exists in the replicate database

In the SYDNEY\_DS Adaptive Server, log in to the pubs2 database, and verify that the publishers table exists:

```
isql -Usa -P -SSYDNEY_DS
use pubs2
go
sp_help publishers
go
```

When you add the replicate pubs2 database using Sybase Central or rs\_init, the maintenance user is created and given replication\_role. The maintenance user must have replication\_role, sa\_role, or alias the Database Owner to replicate truncate table.

In SYDNEY\_DS, make sure the maintenance user has select, insert, delete, and update permissions on the publishers table:

```
grant all on publishers to SYDNEY_DS_maint
go
```

#### Creating the subscription

Log in to the SYDNEY\_RS Replication Server using the “pubs2\_user” login name and create the subscription publishers\_sub for the replication definition publishers\_rep:

```
isql -Upubs2_user -Ppubs2_pw -SSYDNEY_RS
create subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
subscribe to truncate table
go
```

This subscription uses atomic materialization, the default. No where clause is included, so all rows will be replicated. Execution of the truncate table command will be reproduced at the destination database.

#### Monitoring subscription materialization

While still logged into SYDNEY\_RS, use the check subscription command to monitor the status of the subscription:

```
check subscription publishers_sub
for publishers_rep
with replicate at SYDNEY_DS.pubs2
go
```

#### Verifying replication

You can also check if replication is occurring as expected by verifying that a row you insert is copied to the replicate table.

In the TOKYO\_DS Adaptive Server, insert a row into the publishers table:

```
isql -Usa -P -STOKYO_DS
use pubs2
go
insert publishers
values ('9950', 'Who Donut', 'Butler', 'CA')
go
```

In the SYDNEY\_DS Adaptive Server, verify that the row you inserted was replicated into the replicate copy of the publishers table:

```
isql -Usa -P -SSYDNEY_DS
```

```
use pubs2
go
select * from publishers
go
```

## Materializing *text*, *unitext*, *image*, and *rawobject* data

In general, you can use any materialization method for subscriptions for tables with columns that use the `text`, `unitext`, `image`, or `rawobject` datatypes. If you use atomic or nonatomic materialization, the Replication Server managing the replicate database selects all of the subscription data into a subscription materialization queue.

If you want to materialize `text`, `unitext`, `image`, or `rawobject` data, you can use automatic materialization *only* if the size of your data row is less than 32K. Otherwise, you must use bulk materialization.

If you are materializing many large data rows, make sure that the Replication Server has sufficient queue space for the data before you create the subscription. For tables with a large volume of `text`, `unitext`, `image`, and `rawobject` data, you may need to add temporary partitions to the Replication Server to complete the materialization.

## Nonatomic materialization

If you are using nonatomic subscription materialization and you have set the `replicate_if_changed` replication status for any `text`, `unitext`, `image`, or `rawobject` column, Replication Server displays a warning message in the error log file. You are cautioned that data may be inconsistent if applications modify the primary table during subscription materialization. Run the `rs_subcmp` program to reconcile the data in the replicate and primary tables.

## Row migration

Under certain conditions, `text`, `unitext`, `image`, and `rawobject` column data may be missing in a replicate table as a result of row migration.



Row migration occurs in a subscription that has a where clause. Updating a column specified in the where clause can cause a row to become valid for, or migrate into, the subscription. When this happens, Replication Server executes an insert in the replicate table. To insert a complete row, each insert would require values for all columns, including text, untext, image, and rawobject columns that did not change in the primary table.

If your application allows rows to migrate into a subscription and you have set any text, untext, image, or rawobject columns to the replicate\_if\_changed replication status, Replication Server displays a warning message in the error log. The message states that a row has migrated into the subscription but that its text, untext, image, or rawobject data is missing.

If a text, untext, image, or rawobject column with the replicate\_if\_changed status was not changed in an update operation at the primary table, and the update causes the row to migrate into a subscription, the inserted row at the replicate table will be missing the text, untext, image, or rawobject data. Run the rs\_subcmp program to reconcile the data in the replicate and primary tables.

## Subscriptions for columns with heterogeneous datatypes

You create subscriptions for table replication definitions in the normal manner when class-level or column-level translations are defined and active. However, certain restrictions apply to use of the where clause.

- Subscriptions that specify columns subject to class- and column-level translations in the where clause cannot be dematerialized automatically. You must use the bulk or no-materialization method.
- Take care creating or defining subscriptions that specify class- or column-level translations in the where clause. Make sure that the value in the where clause comparison is in the declared datatype format. HDS translations take place *after* the subscription is presented.

For example, if searchable column starttime is declared as datetime but published as rs\_db2\_time, then the comparison value in the where clause must be described using datetime format.

```
create subscription db2_time_sub
for table_rep_def XXXXX
with primary at AAAAA
```

```
with replicate at BBBBB
where starttime > '19000101 23:14:02'
```

and not “where starttime > '23:14:02,'” which is rs\_db2\_time format.

For a detailed discussion of heterogeneous datatype translations, see Chapter 9, “Managing Replicated Tables”.

## Bitmap subscriptions

Bitmap subscriptions allow you to create subscriptions that replicate rows based on bitmap comparisons. When you create a replication definition for a table, specify the datatype of your bitmap columns as `rs_address`. This special datatype tells Replication Server to treat these int columns as bitmaps.

The create subscription and define subscription commands support a bitmap comparison operator (&) in the where clause for `rs_address` columns or parameters.

In the Adaptive Server table, you use an int column to hold a bitmap, since Adaptive Server allows bitwise operators on integer values. An int column has 32 bits. You can have multiple `rs_address` columns in a replication definition if your application requires more than 32 bits.

When you create a subscription, specify bitmap comparisons by comparing each `rs_address` column to a bitmask using the & operator. Each subscription can have one comparison per `rs_address` column.

Bitmap subscription example

For example, consider an application that uses an `rs_address` column named `book_type` to record the categories of books customers are interested in reading. The book categories are mapped into the lower 8 bits of a bitmap column, as shown in Table 11-4:

**Table 11-4: Example bitmap comparison**

Bit number	Book category
0	Science fiction
1	Mystery
2	Business
3	Cooking
4	Popular computing
5	Computer science

Bit number	Book category
6	Psychology
7	Reference

If a bit is set, the customer has expressed interest in books of the corresponding category. The bits are numbered from least significant to most significant. For example, if the customer is interested in mystery, cooking, computer science, and psychology books, the least significant 8 bits are 01101010 and the 32-bit integer value is 106. The `book_type` column in the customer's row contains the value 106.

To create a subscription for customers who are interested in specified book categories, form a bitmask of the desired categories and compare it, using the `&` operator, to the `book_type` column in the `where` clause of the `create subscription` or `define subscription` command. The `&` operator performs a bitwise AND operation. If the result is non-zero, the row matches the subscription.

For `rs_address` columns *only*, the bitmap comparison operator `&` is supported in the `where` clause, as follows:

```
where rs_address_column1 & bitmask
[and rs_address_column2 & bitmask]
[and other_search_conditions]
```

For example, to create a subscription for all customers who are interested in mystery or business books, the lower 8 bits of the mask are 00000110. Converted to a 32-bit integer value, the bitmask is 6. For atomic or nonatomic materialization, you can create the subscription as follows:

```
create subscription mystery_or_business
for customers
with replicate at BRANCH_22.BOOK_DB
where book_type & 6
```

You can use a similar approach in the `define subscription` command, used for bulk materialization. For subscriptions to function replication definitions, which require the no-materialization method or bulk materialization, specify parameter names instead of column names.

See “Using the `where` clause” on page 371 for more information.

In addition to 32-bit integer values, you can also compare `rs_address` columns to 32-bit hexadecimal numbers in the `where` clause. If you use hexadecimal numbers, pad each number with zeros, as necessary, to create an 8-digit hexadecimal value.

---

**Warning!** Hexadecimal values are treated as binary strings by both Adaptive Server and Replication Server. Binary strings are converted to integers by copying bytes. The resulting bit pattern may represent different integer values on different platforms. For example, `0x0000100` represents 65,536 on platforms that consider byte 0 most significant, and represents 256 on platforms that consider byte 0 least significant. Because of these byte-ordering differences, bitmap subscriptions involving hexadecimal numbers might not work if a replication system involves different platforms. Be very cautious about comparing `rs_address` columns to hexadecimal numbers in the `where` clause of a subscription.

---

Replication Server does not replicate a row if the only changed columns are `rs_address` columns, unless the changed bits indicate that the row should be inserted or deleted at the replicate database. Because of this filtering, `rs_address` columns in replicate databases may not be identical to the corresponding columns at the primary database. This is an optimization for applications that use `rs_address` columns to specify the destination replicate databases.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for more information about creating bitmap subscriptions using `create subscription` and `create replication definition` commands.

Refer to the *Adaptive Server Enterprise Reference Manual* and the *Open Client and Open Server Common Libraries Reference Manual* for more information about conversions between datatypes.

## Obtaining subscription information

Once data is replicating, you may need to obtain information about the subscriptions or verify that data is replicating consistently. Replication Server provides stored procedures for obtaining information and a standalone utility for verifying consistency.

## Displaying subscription information

To display information about subscriptions at a Replication Server, you can use the `rs_helpsub` and `rs_helprepdb` stored procedures in the Replication Server RSSD.

Use `rs_helpsub` to display information about subscriptions at a Replication Server. The syntax is:

```
rs_helpsub [subscription_name
           [, replication_definition
           [, data_server, database]]]
```

Use the `rs_helprepdb` stored procedure to display information about databases with subscriptions for replication definitions in the current Replication Server. The syntax is:

```
rs_helprepdb [, data_server, database]
```

For parameter descriptions, refer to Chapter 6, “Adaptive Server Stored Procedures,” in the *Replication Server Reference Manual*.

## Verifying subscription consistency

After you create a subscription, Replication Server propagates transactions from the primary database to the replicate database. The replication system keeps the replicate copy of the table consistent with the primary copy.

The replicate data may become inconsistent with the primary version. For example, if you have not restricted update permissions on a replicate table to the maintenance user for the database, a client may update the replicate data directly, introducing inconsistencies.

Primary and replicate tables may be temporarily inconsistent because Replication Server takes some time to transfer updates from the primary table to the replicate table. However, as soon as the Replication Server applies the updates at the replicate database, inconsistency due to latency no longer exists.

There are three kinds of inconsistency that may occur between primary and replicate tables:

- Missing rows in the primary table are missing from the replicate table.
- Inconsistent rows in the primary table differ from the corresponding rows in the replicate table.
- Orphaned rows in the replicate table do not exist in the primary table or do not match subscriptions for the replicate table.

You need to differentiate between temporary inconsistencies caused by delay and real inconsistencies caused by the incorrect use of the system or by system failures. The `rs_subcmp` program described in the following section helps you make this distinction. You can correct inconsistencies by dropping and recreating subscriptions or by using `rs_subcmp`.

## Using `rs_subcmp` to locate and correct inconsistencies

For Sybase databases, the standalone executable program `rs_subcmp` compares a replicate table to the primary version of the table, finding—and correcting if you so choose—missing, orphaned, and inconsistent rows. On UNIX systems, the program is called `rs_subcmp`. On PC systems, the program is called `subcmp`.

The `rs_subcmp` program is located in the *bin* subdirectory of the Sybase release directory. Refer to the Replication Server installation and configuration guides for your platform for more information.

The program works by logging in to the primary data server and the replicate data server, and selecting and comparing rows from both tables.

Because some differences between primary and replicate data can be attributed to latency, `rs_subcmp` first identifies inconsistencies, and then performs iterations a specified number of times. `rs_subcmp` waits for any updates to be replicated before removing the corrected rows from its list.

It is best to use `rs_subcmp` when latency is low to avoid the program having to perform several iterations through the data.

You can instruct `rs_subcmp` to display inconsistent rows on the standard output, correct them, or both display and correct them.

Creating a configuration file avoids the need for complex command lines, which are prone to errors. Here is an `rs_subcmp` configuration file that compares the `sales` table in the `pubs2` database in the data servers `TOKYO_DS` and `SYDNEY_DS`:

```
PDS=TOKYO_DS
RDS=SYDNEY_DS
PDB=pubs2
RDB=pubs2
RTABLE=sales
RSELECT=select * from sales \
        order by stor_id, ord_num
RUSER=sa
KEY=stor_id
KEY=ord_num
```

```
RECONCILE=Y
RECONCILE_CHECK=Y
WAIT=15
NUM_TRIES=5
VISUAL=Y
```

The *PTABLE*, *PSELECT*, and *PUSER* parameters, which are used for the primary database, are not shown in this example. Their values are the same as those of corresponding parameters in the replicate databases, so they need not be included in the configuration file.

The *RSELECT* line and the *PSELECT* line (if used) must be entered on one line. To continue a line onto the next line (row), precede each newline character with a backslash as, for example:

```
RSELECT=select * from sales \
        order by stor_id, ord_num
```

---

**Note** Due to update filtering, columns of *rs\_address* datatype may not be identical between the primary and replicate databases. Do not select *rs\_address* columns using *RSELECT* or *PSELECT* parameters.

---

When you execute *rs\_subcmp*, you can override values in the configuration file with command line options. For example, rather than changing the name of the *TOKYO\_DS* data server to *TOKYO\_DS2* in the configuration file, you can specify it on the command line, using the *-S* flag, as the following example illustrates:

```
rs_subcmp -f sales_cmp -S TOKYO_DS2 > sales_badrows
```

In this example, the *-f* option specifies a configuration file name, *sales\_cmp*. If the *VISUAL* parameter is set to “Y” in the configuration file (equivalent to the *-V* command line option), a list of the inconsistent rows is generated. In this example, the output is redirected to a file.

#### Schema comparison

Schema comparison is useful in comparing schema between two databases that may have the same data but different schemas.

For example, if you want to compare all schemas between two databases using the *config.cfg* file:

```
rs_subcmp -f config.cfg
```

A report file which details the comparison result between two tables or two databases is created after every schema comparison. The report file is named *reportPROCID.txt*. If inconsistencies exist, `rs_subcmp` creates a reconciliation script named *reconcilePROCID.sql*. The report file and the reconciliation script are saved in the same directory where you issued the `rs_subcmp`.

---

**Note** Before running `rs_subcmp` for schema comparison, make sure that `ddlgen` is working on your environment.

---

See `rs_subcmp` in Chapter 7, “Executable Programs” of the *Replication Server Reference Manual* for detailed information in using schema comparison.

Manual data  
reconciliation

To verify the reconciliation of statements before execution, a reconciliation file can be created using the `rs_subcmp` command. You can use the command line option `-g` with `rs_subcmp` or you can set the configuration file parameter `RECONCILE_FILE` to “Y” to indicate the creation of a reconciliation file.

*rs\_subcmp*  
performance  
enhancement

Hash algorithm improves the performance of `rs_subcmp` and compresses the data in primary and replicated tables. The compressed data is then fetched by `rs_subcmp`.

Instead of taking the entire row of data during comparison between the primary table and replicated table, `rs_subcmp` now transfers only the compressed data of each data row from the primary or replicated tables, and then verifies or reconciles inconsistencies between them.

For an improve `rs_subcmp` performance, use the command line parameters `-h` or `-H` or their equivalent configuration file parameters `FASTCMP` or `HASH_OPTION`.

---

**Note** To support hash algorithm, `rs_subcmp` requires ASE 15.0.2 or later and cannot handle case-sensitive comparison. It also cannot handle text, unitext or image datatypes and does not allow the user to specify the precision for the float datatype (maximum precision is used). Also, Sybase suggests to set the ASE parameter default data cache to 128M or higher to get a better comparison performance.

---

The `rs_subcmp` program has a large number of options, which you can specify on the command line or in a configuration file. Refer to Chapter 7, “Executable Programs,” in the *Replication Server Reference Manual* for a list of these configuration file parameters and command line options.



## Using publication subscriptions

With publication subscriptions, you create subscriptions for a group of replication definitions using a single command. You collect replication definitions and their articles in a publication at the primary Replication Server. At the replicate Replication Server, you create a publication subscription against that publication.

When you create a publication subscription, Replication Server creates a subscription for each article in the publication.

Publication subscriptions and article subscriptions follow the rules and requirements for single subscriptions with one exception: They cannot contain where clauses. To specify a subset of rows that the replicate Replication Server receives, include where clauses in the article. Refer to “Specifying a where clause with the create article command” on page 312 for more information.

To use publications, the primary Replication Server must be version 11.5 or later. To use publication subscriptions, the replicate Replication Server and the route from the primary Replication Server and the replicate Replication Server must be version 11.5 or later.

The following restrictions apply:

- A valid publication must exist before you can create a publication subscription against it.
- The name of a publication subscription must be unique to the publication, to the destination data server, and to the destination database.
- You can include articles in one or more publications that reference different replication definitions for the same primary table. However, you cannot subscribe to more than one replication definition per primary table for each replicate table.

Use the command line to create and manage publication subscriptions.

Refer to “Using publications to replicate data at the command line” on page 309 for a list of steps for creating publications and publication subscriptions.

Refer to “Using publications” on page 308 for an overview of creating publications and publication subscriptions.

## Commands for creating and managing publication subscriptions

Table 11-5 lists the RCL commands for working with publication subscriptions. All of these commands, except check subscription, require primary subscribe or create object permission at the source Replication Server and create object permission at the destination Replication Server. Anyone can execute check subscription.

See Table 9-3 on page 309 for a list of RCL commands for working with publications.

**Table 11-5: Commands for managing publication subscriptions**

Command	Task
create subscription <i>sub_name</i> for publication <i>pub_name</i>	Creates a subscription for a publication and a subscription for each article in the publication. With create subscription, you can: <ul style="list-style-type: none"> <li>• Subscribe to table replication definitions using the atomic, nonatomic, or no-materialization method.</li> <li>• Subscribe to function replication definitions using the no-materialization method.</li> </ul> See “Using the create subscription command” on page 398.
define subscription <i>sub_name</i> for publication <i>pub_name</i>	Defines a subscription for a publication and a subscription for each article in the publication. Use with activate subscription and validate subscription. <p>With define subscription, you can subscribe to articles with table replication definitions or function replication definitions using the bulk materialization method. See “Creating publication subscriptions with bulk materialization” on page 399.</p>
activate subscription <i>sub_name</i> for publication <i>pub_name</i>	Activates a subscription for a publication and a subscription for each article in the publication. Use with define subscription and validate subscription for bulk materialization. See “Creating publication subscriptions with bulk materialization” on page 399.
validate subscription <i>sub_name</i> for publication <i>pub_name</i>	Validates a subscription for a publication and a subscription for each article in the publication. Use with define subscription and activate subscription for bulk materialization. See “Creating publication subscriptions with bulk materialization” on page 399.
check subscription <i>sub_name</i> for publication <i>pub_name</i>	Displays the status of the publication subscription and all of its article subscriptions. See “Displaying status information” on page 403.
check subscription <i>sub_name</i> for article <i>article_name</i> in <i>pub_name</i>	Displays the materialization status of an article subscription. See “Displaying status information” on page 403.
rs_helppubsub	Displays information about publication subscriptions.
drop subscription <i>sub_name</i> for publication <i>pub_name</i>	Removes the publication subscription and all of its article subscription from the rs_subscriptions system table at the primary and replicate sites. See “Dropping subscriptions for publications and articles” on page 401.

Command	Task
drop subscription <i>sub_name</i> for article <i>article_name</i> in <i>pub_name</i>	Removes the article subscription from the publication subscription and from the <code>rs_subscriptions</code> system table at the primary and replicate sites. See “Dropping subscriptions for publications and articles” on page 401.

## Enabling replication of the *truncate table* command

When you create, refresh, or define a publication subscription, you can enable replication of truncate table to the replicate table. If you do not, you must execute truncate table yourself at the replicate database.

For example, to create the publication subscription *pubs2\_sub* and enable replication of truncate table, enter this command at the destination Replication Server:

```
create subscription pubs2_sub
  for publication pubs2_sub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
  subscribe to truncate table
```

All subscriptions to the same replicate table must use truncate table consistently. If a replicate table has a subscription that does not enable replication of truncate table and you add another subscription that does enable replication of truncate table, the publication subscription fails.

You do not need to include subscribe to truncate table when you activate and validate the publication subscription.

See “Enabling replication of truncate table” on page 373 for more information.

## Creating publication subscriptions

Once a publication has been validated, you can create subscriptions against it. When you create a publication subscription, Replication Server creates a subscription for each article in the publication.

Publication subscriptions and article subscriptions specify the publication, the primary and replicate databases, and the materialization method. They do not contain where clauses. To specify a subset of rows to be replicated, include where clauses in the article description. Refer to “Specifying a where clause with the create article command” on page 312.

## Using the *create subscription* command

Use *create subscription* to create a publication subscription and an article subscription for each article in the publication. You can use *create subscription* to materialize source data at the destination database using the atomic, nonatomic, or no-materialization method.

Execute *create subscription* at the Replication Server that manages the destination database. Subscription information is stored in the *rs\_subscriptions* system tables at the primary and replicate sites.

The following example creates a subscription named *pubs2\_sub* for the publication *pubs2\_pub*. It also creates a subscription named *pubs2\_sub* for each article in *pubs2\_pub*. The source database is *pubs2* managed by the TOKYO\_DS data server. The destination database is also named *pubs2*; it is managed by the SYDNEY\_DS data server.

```
create subscription pubs2_sub
  for publication
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
```

---

**Note** The maximum size of a where clause in a *create subscription* statement is 255 characters.

---

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Specifying a materialization method

Specify materialization methods for publication subscriptions in the same way you specify materialization methods for regular subscriptions. When you use *create subscription*, you can specify atomic, nonatomic, or the no-materialization method. The default method is atomic materialization, using the *select with holdlock* operation.

Article subscriptions share the name of the parent subscription and generally, its materialization method. However, function replication definitions require the bulk or no-materialization method. If you use *create subscription*, and articles in the publication reference function replication definitions, Replication Server uses the no-materialization method for these article subscriptions—regardless of the materialization method specified in the publication subscription.

See “Subscription materialization methods” on page 353 for a description of the different materialization methods.

### **Refreshing publication subscriptions**

When you add articles to an existing publication, you must add article subscriptions to the existing publication subscription to subscribe to the new articles. Use `refresh subscription` for new articles to refresh the subscription. This clause instructs Replication Server to check the subscription against the publication and then to create subscriptions for any unsubscribed articles.

For example, to refresh the publication subscription `pubs2_sub`, enter this command at the destination Replication Server:

```
create subscription sub for publication pub
with primary at TOKYO_DS.pubs2
with replicate at SYDNEY_DS.pubs2
for new articles
```

Use `check subscription` to find out whether a subscription exists for each article in a publication. See “Displaying status information” on page 403 for more information about `check subscription`.

### **Creating publication subscriptions with bulk materialization**

Bulk materialization allows you to load subscription data from media such as magnetic tape. Use this method if the amount of data to be transferred is too large to copy through the network. You can also use this method to create subscriptions for function replication definitions.

When you create publication subscriptions with bulk materialization, you must use `define subscription`, `activate subscription`, and `validate subscription`. You use these bulk materialization commands to create publication subscriptions in the same way you create single subscriptions. You cannot include `where` clauses in publication subscriptions.

Refer to “Specifying a `where` clause with the `create article` command” on page 312 for information about adding `where` clauses to articles.

### **Using the *define subscription* command**

Use `define subscription` to create a publication subscription and a subscription for each article in the publication. `define subscription` always creates a subscription using bulk materialization.

Execute `define subscription` at the Replication Server that manages the destination database. Subscription information is stored in the `rs_subscriptions` system tables at the source and destination sites.

All subscriptions in the publication subscription are created at the same time.

The following example creates a subscription named `pubs2_sub` for the publication `pubs2_pub`.

```
define subscription pubs2_sub
  for publication pubs2_pub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
```

When you define a publication subscription with bulk materialization, you can enable replication of truncate table to the destination table. See “Enabling replication of the truncate table command” on page 397 for more information.

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

### Using the `activate subscription` command

Use `activate subscription` to activate a publication subscription and its subscription subset. Execute `activate subscription` at the Replication Server that manages the destination database.

Before you execute `activate subscription`, you must execute `define subscription`, and the publication subscription status must be `DEFINED`. Refer to “Displaying status information” on page 403 for information about displaying subscription status.

All subscriptions in the publication subscription are activated at the same time.

The following example activates every subscription in the publication subscription `pubs2_sub`.

```
activate subscription sub for publication pub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

### Using the `validate subscription` command

Use `validate subscription` to set the subscription status to `VALID` for the publication subscription, and its subscription subset. Execute `validate subscription` at the Replication Server that manages the replicate database.

Before you execute `validate subscription`, you must execute `activate subscription` and the publication subscription status must be `ACTIVE`. Refer to “Displaying status information” on page 403 for information about displaying subscription status.

All subscriptions in the publication subscription are validated at the same time.

The following example validates every subscription in the publication subscription `pubs2_sub`.

```
validate subscription sub for publication pub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

### Refreshing publication subscriptions using bulk materialization

When you refresh a publication subscription using bulk materialization, use the `for new articles` clause when you define the publication subscription. You do not need to repeat the clause when you activate and validate the subscription.

The following example refreshes the publication subscription `pubs2_sub`.

```
define subscription pubs2_sub
  for publication pubs2_pub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
  for new articles
```

See “define subscription” in the *Replication Server Reference Manual* for syntax and usage guidelines.

To check whether a subscription exists for each article in a publication, execute `check subscription` at the primary or replicate Replication Server. See “Displaying status information” on page 403 for more information about `check subscription`.

### Dropping subscriptions for publications and articles

Use `drop subscription` to drop a publication subscription and all of its article subscriptions, or to drop a single article subscription.

drop subscription removes information about the publication subscription and its article subscriptions from system tables at the source and destination servers. It does not remove publication information from the destination server. Thus, you can create another subscription against the publication, and Replication Server only needs to reload primary site information if it has been changed.

Include the without purge clause to retain existing rows replicated by the subscription to the destination database. The subscriptions are dropped all at once.

This example drops a subscription named *pubs2\_sub* for the publication *pubs2\_pub* using without purge.

```
drop subscription pubs2_sub
  for publication pubs2_pub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
  without purge
```

Include the with purge clause to delete existing rows replicated by the subscription to the destination database. The subscriptions are dropped one at a time.

This example uses with purge:

```
drop subscription pubs2_sub
  for publication pubs2_pub
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
  with purge
```

The following example deletes the article *pubs2\_art*, without removing rows replicated by the subscription.

```
drop subscription sub for article pubs2_art
  with primary at TOKYO_DS.pubs2
  with replicate at SYDNEY_DS.pubs2
  without purge
```

Refer to Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for complete syntax and usage guidelines.

## Viewing publication subscription information

You can view information about publication and article subscriptions with the check subscription command or the rs\_helppubsub stored procedure.



## Displaying status information

Use check subscription at the primary Replication Server or the replicate Replication Server to check the status of a publication subscription and its article subscriptions or to check the status of an article subscription.

check subscription returns a status (such as VALID, MATERIALIZING, or ACTIVE) along with a descriptive message. See “check subscription” in Chapter 3, “Replication Server Commands,” in the *Replication Server Reference Manual* for syntax and usage guidelines and a list of status messages.

- This example displays the subscription status of the publication subscription *pubs2\_sub*.

```
check subscription pubs2_sub
      for publication pubs2_pub
      with primary at TOKYO_DS.pubs2
      with replicate at SYDNEY_DS.pubs2
```

If the publication subscription is valid, Replication Server also checks whether the subscription is current. When you alter a publication after the subscription is created, the publication subscription is out of sync with the publication. To create subscriptions for new articles and make the subscription current, refresh the subscription using create subscription or define subscription.

- This example displays the subscription status of the article *pubs2\_art* in the subscription *pubs2\_sub*.

```
check subscription sub for article pubs2_art
      in pubs2_pub
      with primary at TOKYO_DS.pubs2
      with replicate at SYDNEY_DS.pubs2List
Publication and Article Subscription Information
```

To display information about a publication subscription and article subscriptions, use the `rs_helppubsub` stored procedure at either the primary or replicate Replication Server RSSD.

Here are some examples of using `rs_helppubsub`:

- To list all publication subscriptions at a site, enter:

```
rs_helppubsub
```

For each publication subscription known to the site, the display includes the names of the subscription and its associated publication, the names of the primary and replicate databases and data servers, status information, and the date of the latest change to the publication subscription.

- To display information about a particular publication subscription, enter:

```
rs_helppubsub subscription_name
```

The output displays the information described in the above example for all publication subscriptions named *subscription\_name*.

- To display information about a particular publication subscription and its article subscriptions, enter:

```
rs_helppub subscription_name, publication_name,  
primary_dataserver, primary_db,  
replicate_dataserver, replicate_db
```

The output displays the information described in the above examples for all publication subscriptions named *subscription\_name*. For each article subscription, the output displays subscription and article name, status information for the primary and replicate Replication Servers, replication definition name, autocorrection status, and the date of the latest change to the article subscription.

See the *Replication Server Reference Manual* for complete syntax and usage guidelines and sample output.

## Managing Replicated Objects Using Multisite Availability

This chapter describes how to set up and manage database replication definitions and subscriptions using multisite availability (MSA).

<b>Topic</b>	<b>Page</b>
Overview	406
Setting up an MSA system	408
Marking data for replication	414
Managing database replication definitions	415
Viewing information about database replication definitions	418
Using database, table, and function replication definitions concurrently	418
Managing database subscriptions	420
Viewing information about database subscriptions	423
Using database, table, and function subscriptions concurrently	423
Replicating the master database in an MSA environment	424
Replicating DDL and system procedures	426
Replicating user stored procedures	427
Customizing function strings	427

This chapter describes how to replicate database objects using database replication definitions and MSA.

- See Chapter 9, “Managing Replicated Tables,” for information about replicating individual tables using table replication definitions.
- See Chapter 10, “Managing Replicated Functions,” for information about replicating individual system stored procedures.

## Overview

MSA can make the process of setting up a replication system both faster and easier.

Some of the features that MSA provides are:

- A simple replication methodology that requires only one replication definition for the primary database and only one subscription for each subscribing database.
- A replication filtering strategy that lets you choose whether or not to replicate individual tables, transactions, functions, system stored procedures, and data definition language (DDL).
- Replication of DDL to any replicate database—including non-warm standby databases.
- Replication to multiple replicate sites—for warm standby as well as non-warm standby databases.

You can overlay MSA scenarios over your existing replication structure. The procedures for implementing MSA are similar to those you use to replicate to warm standby or replicate databases.

### Database replication

When you use table and function replication, you describe each piece of data that is to be replicated using individual table and function replication definitions and subscriptions. This methodology allows you to transform data and provides fine-grained control over the information being entered in the replicate database. However, it requires that you mark each table or function to be replicated, create a replication definition for each replicated table or function, and create subscriptions for each replication definition at each replicate database.

MSA lets you identify specific database objects: tables, functions, transactions, DDL, and system stored procedures in a single replication definition. You can choose to replicate the entire database, or you can choose to replicate—or not replicate—particular tables, functions, transactions, DDL, and system stored procedures in that database. If you do not need to replicate partial tables, MSA can provide replication while affording the advantages of simple setup and maintenance.

When the replicate is a warm standby database

In the non-MSA warm standby scenario, changes to the primary database are copied directly to the warm standby database without alteration. This methodology allows replication of DDL. To change or qualify the data sent, you must add table and function replication definitions. Each primary database can have one, and only one, standby database. See Chapter 3, “Managing Warm Standby Applications,” in the *Replication Server Administration Guide Volume 2* for a complete discussion of this warm standby application.

MSA provides all the features of the warm standby application described in Chapter 3, “Managing Warm Standby Applications,” in the *Replication Server Administration Guide Volume 2*. In addition, MSA:

- Enables replication to multiple standby databases
- Allows you to replicate or not replicate specific database objects

Bidirectional replication support for DDL in MSA

You can configure multisite availability (MSA) to set up a two-way replication of DDL transactions between two Adaptive Server databases.

Replication Server 15.0 and later supports this bidirectional replication using a Replication Server configuration parameter called `dsi_replication_ddl`. When `dsi_replication_ddl` is set to on, DSI sends set replication off to the replicate database, which instructs it to mark the succeeding DDL transactions available in the system log not to be replicated. Therefore, these DDL transactions are not replicated back to the original database, which enables DDL transactions replication in bidirectional MSA replication environment.

To set up bidirectional replication:

- 1 Create a bidirectional MSA replication environment. See “Creating a bidirectional replication environment” on page 70.
- 2 Grant “set session authorization” privilege to the maintenance user on the destination database, as shown in the following example:

```
grant set session authorization to maint_user
```

- 3 Alter the connection of the destination database to set `dsi_replication_ddl` configuration parameter to “on” to enable bidirectional DDL replication, as shown in the following example:

```
alter connection to dataserver.database set dsi_replication on
```

- 4 Replicate DDL transactions.

MSA mixed-version environment

In an MSA mixed-environment, the primary Replication Server filters the data features with higher versions.

Incompatible commands are not sent to the standby Replication Server. The configuration parameter `dist_stop_unsupported_cmd` suspends the DIST if there are incompatible commands. You can configure this parameter using the following syntax:

```
configure replication server
  set 'dist_stop_unsupported_cmd' to ['on' | 'off']
alter connection srv.db
  set 'dist_stop_unsupported_cmd' to ['on' | 'off']
alter logical connection lsrv.ldb
  set 'dist_stop_unsupported_cmd' to ['on' | 'off']
```

By default, `dist_stop_unsupported_cmd` is off. When the parameter is on, the DIST suspends itself if a command cannot be sent to some destinations. Resume DIST by skipping the entire transaction, or reset the parameter to off.

## Setting up an MSA system

You can set up MSA replication in many different ways. This section describes how to set up three representative MSA replication architectures:

- Simple, full-database replication
- Replication of specified tables and functions
- Replication to multiple replicate databases

You can easily add syntax to these examples to replicate DDL or system stored procedures. See “Replicating DDL and system procedures” on page 426.

## Replicating the database

In this simple scenario, you use database replication definitions and subscriptions to replicate the entire primary database to one or more replicate databases.

The basic steps are:

- 1 Mark the primary database for replication using `sp_reptostandby`. For example:

```
sp_reptostandby primary_db, 'all'
```

---

**Note** `sp_reptostandby` does not mark user stored procedures for replication. You must mark each user stored procedure individually using `sp_setreproc`.

---

- 2 Set the RepAgent parameter `send warm standby xacts` to true so that RepAgent sends system transactions and DDL to both standby and replicate databases. For example, at the primary data server, enter:

```
sp_config_rep_agent primary_db,  
  'send warm standby xacts', 'true'
```

- 3 Create a database replication definition using `create database replication definition` at the primary Replication Server. For example:

```
create database replication definition repdef_1  
  with primary at PDS.primary_db
```

See “create database replication definition” in the *Replication Server Reference Manual* for complete syntax and usage information.

- 4 Create a database subscription for each subscribing database. In this example, we are creating a database subscription using `create subscription` and the `no materialization` method. The primary and replicate databases have been synchronized prior to subscription. You can also use `create subscription if activities` at the primary database can be suspended.

For example, at the replicate Replication Server, enter:

```
create subscription sub_1  
  for database replication definition repdef_1  
  with primary at PDS.primary_db  
  with replicate at RDS.rdb  
  without materialization  
  subscribe to truncate table
```

When creating a database subscription, you can use the `no materialization` method (as shown in step 4) or the `bulk materialization` method to synchronize databases. The procedure you use depends on which materialization method you choose and whether primary table activities can be suspended.

See “Materialization” on page 421 for syntax and usage information for using the `bulk materialization` method.

## Replicating tables and functions

You can use MSA capabilities to replicate particular tables or functions. The basic steps are:

- 1 Mark tables, stored procedures, and database for replication and create the database replication definition.

In this example, we are replicating `table1` and `table2` only. You can identify particular tables in either of two ways:

- Mark the database for replication using `sp_reptostandby`. Create the database replication definition and identify specific tables for replication using `create replication definition`. You must also tell the RepAgent to send replicate data to replicate as well as standby databases.

Enter this information at the primary data server:

```
sp_reptostandby primary_db, 'all'  
sp_config_rep_agent primary_db,  
  'send warm standby xacts', 'true'
```

Enter this information at the primary Replication Server:

```
create database replication definition rep_1B  
with primary at PDS.pdb  
  replicate tables in (table1, table2)
```

- Alternatively, mark particular tables and stored procedures for replication using `sp_setreptable` and `sp_setrepproc`. Then, create the database replication definition. For example:

```
sp_setreptable table1, 'true'  
sp_setrptable table2, 'true'  
  
create database replication definition rep_1A  
with primary at PDS.pdb
```

---

**Note** You can replicate DDL changes only if you mark its database for replication using `sp_reptostandby`.

---



- 2 Create the database subscription. To subscribe without materialization, see “Replicating the database” on page 408. To subscribe using bulk materialization, see “Materialization” on page 421.

---

**Note** You can also use `sp_reptostandby` to mark the database and then create table replication definitions and subscriptions—without creating a database replication definition. This method eliminates the need to mark individual tables, yet allows you to select and replicate partial tables. The database connection parameter `rep_as_standby` must be on.

---

For considerations when dealing with encrypted columns, see “Replicating encrypted columns”.

## Using replicate databases as warm standby databases

You can use MSA to replicate DDL and other database objects to multiple replicate or warm standby databases. You can create database replication definitions and database subscriptions to logical connections. See Chapter 3, “Managing Warm Standby Applications,” in the *Replication Server Administration Guide Volume 2* for detailed information about setting up logical connections.

This section uses an example to describe the basic setup for a multiple warm standby architecture. In this example, you replicate from one primary database (`dsA.db`) to two replicate databases (`dsB.db` and `dsC.db`). There is a single Replication Server controlling replication, and only standby replication takes place to and from the primary database. Only `dsA` can replicate DDL and system stored procedures. If users are switched to `dsB.db` or `dsC.db`, DDL and system stored procedures are not replicated.

---

**Note** This example uses a different database replication definition for each subscribing site. You could also create a single database replication definition that handles the common set of replicated tables and functions, and then create table and function subscriptions for the tables and functions that are not common to both standby databases.

---

The basic steps are:

- 1 Suspend all database activities.
- 2 Mark `dsA.db`, `dsB.db`, and `dsC.db` for replication using `sp_reptostandby`.

- 3 At each data server, set send warm standby xacts to true for each RepAgent. For example:

```
sp_config_rep_agent db,  
  'send warm standby xacts', 'true'
```

- 4 At Replication Server, set dsi\_replication off for each connection. For example:

```
alter connection to dsB.db  
  set dsi_replication 'off'
```

---

**Note** Sybase recommends that you set dsi\_replication to off for warm standby connections as it prevents replicated data in the transaction log from being replicated again in the event of a switchover. dsi\_replication should be turned on (the default) for normal replication.

---

- 5 Create a database replication definition for each database, defining each as the primary. For example:

```
create database replication definition rep_2  
  with primary at dsA.db  
  replicate DDL  
  replicate system procedures  
  
create database replication definition rep_2  
  with primary at dsB.db  
  
create database replication definition rep_2  
  with primary at dsC.db
```

- 6 As each database can be a primary or a standby database, create or define subscriptions so that each database subscribes to every other database. You can use different materialization methods for each subscription. For example:

```
create subscription sub_2B  
  for database replication definition rep_2  
  with primary at dsB.db  
  with replicate at dsA.db  
  without materialization  
  subscribe to truncate table  
  
create subscription sub_2C  
  for database replication definition rep_2  
  with primary at dsC.db  
  with replicate at dsA.db  
  without materialization  
  subscribe to truncate table
```

```
define subscription sub_2A
  for database replication definition rep_2
  with primary at dsA.db
  with replicate at dsB.db
  subscribe to truncate table
  use dump marker

create subscription sub_2C
  for database replication definition rep_2
  with primary at dsC.db
  with replicate at dsB.db
  without materialization
  subscribe to truncate table

define subscription sub_2A
  for database replication definition rep_2
  with primary at dsA.db
  with replicate at dsC.db
  subscribe to truncate table
  use dump marker

create subscription sub_2B
  for database replication definition rep_2
  with primary at dsB.db
  with replicate at dsC.db
  without materialization
  subscribe to truncate table
```

- 7 Dump dsA.db.
- 8 With the dsB.db DSI suspended, load database to dsB.db.
- 9 Resume connection to dsB.db.
- 10 With the dsC.db DSI suspended, load database to dsC.db.
- 11 Resume connection to dsC.db.
- 12 Resume database activities.

## Switchover

In any standby situation, switchover involves disconnecting users from the active database and reconnecting them to the new active database. In this case, switchover must wait for the queues to empty so that no transactions are lost.

Refer to Chapter 3, “Managing Warm Standby Applications,” in the *Replication Server Administration Guide Volume 2* for more information about logical connections and switchover.

## Marking data for replication

You can mark databases, tables, and functions for replication using `sp_reptostandby`, `sp_setreptable`, and `sp_setrepproc`.

When the database is marked by `sp_reptostandby`:

- The RepAgent configuration parameter `send warm standby xacts` must be true.
- User-defined stored procedures are not replicated unless they are marked individually using `sp_setrepproc`.
- RepAgent sends DDL, system procedures, and transactions to the Replication Server. A database replication definition can filter them out at the Replication Server.
- And you use table replication definitions and table subscriptions, you can send table data to both replicate databases and warm standby databases by setting the database connection parameter `rep_as_standby` on.

When the database is not marked by `sp_reptostandby`, DDL is not replicated for the marked tables and functions.

Table 12-1 summarizes how data is replicated.

**Table 12-1: Data replication**

Data marked by	Table and function subscriptions only	Database subscription only	Table, function, and database subscriptions coexist
<code>sp_setreptable</code> and <code>sp_setrepproc</code>	<ul style="list-style-type: none"> <li>• Replicate marked data</li> <li>• Do not replicate DDL</li> </ul>	<ul style="list-style-type: none"> <li>• Replicate marked data</li> <li>• Do not replicate DDL</li> </ul>	<ul style="list-style-type: none"> <li>• Replicate marked data</li> <li>• Do not replicate DDL</li> </ul>
<code>sp_reptostandby</code>	<ul style="list-style-type: none"> <li>• Check <code>rep_as_standby</code></li> <li>• Do not replicate DDL</li> </ul>	<ul style="list-style-type: none"> <li>• Replicate all data</li> <li>• Replicate DDL (optional)</li> </ul>	<ul style="list-style-type: none"> <li>• Check <code>rep_as_standby</code></li> <li>• Replicate DDL (optional)</li> </ul>
<code>sp_setreptable</code> , <code>sp_setrepproc</code> , and <code>sp_reptostandby</code>	<ul style="list-style-type: none"> <li>• Check <code>rep_as_standby</code></li> <li>• Do not replicate DDL</li> </ul>	<ul style="list-style-type: none"> <li>• Replicate all data</li> <li>• Replicate DDL (optional)</li> </ul>	<ul style="list-style-type: none"> <li>• Check <code>rep_as_standby</code></li> <li>• Replicate DDL (optional)</li> </ul>

## Managing database replication definitions

A table, a function, a transaction, DDL, or a system stored procedure can be a replicated object. A database replication definition specifies filters for replicated objects—either including or excluding the same or entire replicated object from replication. For example:

```
create database replication definition rep_1C
with primary at PDS.pdb
  replicate tables in (table1, table2)
  not replicate functions in (fc_a)
  not replicate system procedures
  replicate transactions
  replicate DDL
```

In this example, we are replicating:

- table1 and table2
- All functions except fc\_a
- All transactions
- Supported DDL commands

We are not replicating:

- Any database tables except table1 and table2
- Function fc\_a
- Any system procedures

See “create database replication definition” in the *Replication Server Reference Manual* for complete syntax and usage information.

---

**Note** Database replication definitions do not support the options `send-standby-all-columns`, `send-standby-all-parameters`, and `send_standby_repdef_cols`. Where a database replication definition exists, Replication Server sends all columns or parameters.

---

## Altering database replication definitions

You can change a database replication definition using `alter database replication definition`. This command allows you to replace one filter at a time. For example:

```
alter database replication definition rep_1C
  with primary at PDS.pdb
  not replicate tables in (table2)
  with dsi_suspended
```

See the *Replication Server Reference Manual* for complete syntax and usage information.

When you execute `alter database replication definition`, Replication Server writes an `rs_marker` to the inbound queue. The command does not take effect until the marker reaches the Distributor (DIST), which will by then have rebuilt the Database Subscription Resolution Engine (DSRE) to incorporate the changes.

Altering a database replication definition with associated subscriptions may desynchronize replicate tables. To resynchronize, you can either:

- Quiesce Replication Server, drain the transaction log, and apply changes manually, or
- Use the `with_dsi_suspended` option, which causes the replicate Replication Server to suspend the replicate DSI when it reads the “alter database replication definition” marker.

❖ **Altering a database replication definition and resynchronizing replicate tables**

- 1 Execute `alter database replication definition` and include the `with_dsi_suspended` phrase.
- 2 Wait for the replicate DSI to suspend.
- 3 Use bulk materialization to resynchronize replicate tables.
- 4 Resume the connection.

## Dropping database replication definitions

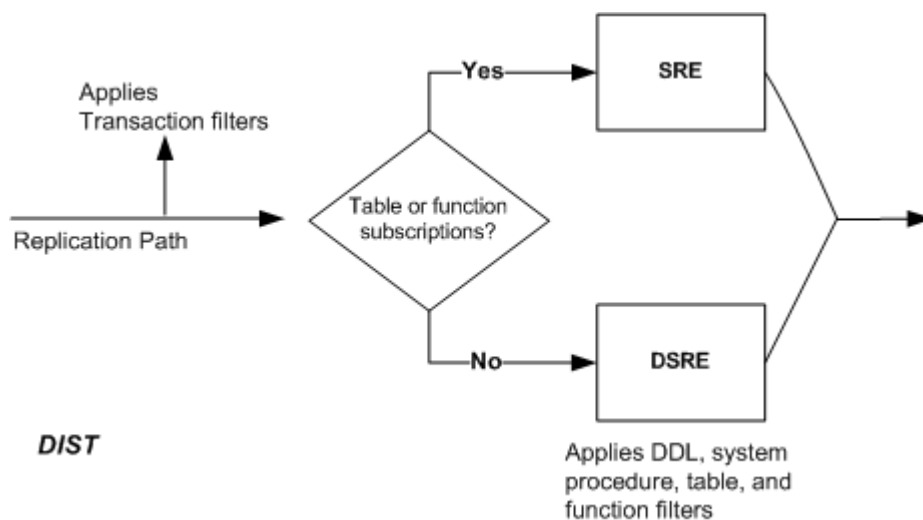
You can drop a database replication definition, but you must first have dropped all associated database subscriptions. See “drop database replication definition” in the *Replication Server Reference Manual* for syntax and usage information.

## Using database replication filters

The Subscription Resolution Engine (SRE) evaluates table and function subscription rows. The Database Subscription Resolution Engine (DSRE) evaluates database objects—except transactions. When a database replication definition causes a transaction to be sent to the replicate Replication Server, the DIST evaluates the transaction *before* other database objects are evaluated by the DSRE and *before* transaction rows are evaluated by the SRE. Thus, Replication Server filters out transactions even if they contain data that satisfies a database subscription or a table or function subscription.

If a database subscription and a table or function subscription coexist for the same table or function, the table or function subscription takes precedence. In this instance, the DIST does not pass the replicated table or function to the DSRE for evaluation; the DIST passes it to the SRE.

**Figure 12-1: Evaluation of database replication filters**



## Viewing information about database replication definitions

You can use `rs_helpdbrep` to view information about a specific database replication definition or all database replication definitions for a database or a data server.

For example, to view information about the `rep_1B` database replication definition, enter:

```
rs_helpdbrep rep_1B, PDS, pdb
```

For syntax and usage information about `rs_helpdbrep`, see the *Replication Server Reference Manual*.

## Using database, table, and function replication definitions concurrently

You do not need to add table and function replication definitions when you use a database replication definition. However, to transform the data, replicate minimal columns, not replicate dynamic SQL, or use primary keys to improve performance, you must do so.

Create and use table or function replication definitions that include the `send standby` clause to:

- Change the name of a replicated table or function
- Change the name of a replicated column
- Publish different column datatypes
- Replicate fewer columns or parameters
- Replicate minimal columns
- Not replicate dynamic SQL
- Use customized function strings

Create and use table or function replication definitions where the `send standby` clause is optional to:

- Declare different table column datatypes
- Improve performance using primary keys



- Set autocorrection for materialization or dematerialization
- Use table or function subscription to override database replication
- Exclude dynamic SQL in the standby database combined with `dynamic_sql` setting

---

**Note** As long as a database replication definition and a database subscription are in place, you can use table or function replication definitions without table or function subscriptions. You need to use table or function subscriptions only if you require the functionality they provide. See “Using database, table, and function subscriptions concurrently” on page 423.

---

If a database replication definition and table replication definitions exist at the primary, and a database subscription but no table subscriptions exist at the replicate, replication behavior depends in part on the presence or absence of the `send standby replication definition columns/parameters` clause in the table or function replication definition.

- If the `send standby` clause is present, the database subscription honors the table or function replication definition; the table replication definition’s primary key columns and `replicate minimal columns` settings are used to replicate into the replicate database. The database subscription always treats `send standby all columns` as `send standby replication definition columns`.
- If the table replication definition does not contain the `send standby` clause and other replication definitions exist for a given table, the database subscription replicates data using the internal table replication definition (the union of all such replication definitions). All columns are replicated, and data is converted to the declared columns or datatypes.

## Altering database replication definitions

Adding or dropping database replication definitions does not affect table or function subscriptions.

When you alter a database replication definition, you replace the database replication filter. Replication Server places a marker in the inbound queue and allows the `DIST` to process the command. The new filter is applied to transactions committed after the marker.

- If a table subscriptions exists, no action is required.

- If no table subscription exists, you must include the `dsi_suspended` clause in the alter database replication definition command, or manually materialize or dematerialize the table.

## Altering table and function replication definitions

If you create or delete a table or function replication definition—and a database subscription exists—the change takes place immediately for new data. Data already in the stable queues continues to reference preexisting conditions until all referring data has been applied.

Similarly, if you drop a table or function replication definition, Replication Server continues to reference that replication definition until all referring data in the stable queues has been applied.

However, if you alter a table or function replication definition when a database subscription exists, the change takes place immediately for new data *and* for data already in the stable queues.

Alter a table or function replication definition using one of these methods:

- Suspend primary table and function activities before using alter replication definition, or
- Create a new table or function replication definition and then drop the old one.

## Managing database subscriptions

When you create a database replication definition at the primary database, you must also create a database subscription at each subscribing database. You can use the no materialization method or the bulk materialization method. If you create a database subscription, you cannot use a `where` clause to set the criteria for subscribed data; all data is subscribed.

If you need to set criteria for particular tables or functions, you can add table or function subscriptions. See “Using database, table, and function subscriptions concurrently” on page 423 for more information.

When there is a database subscription, the DSI for that connection is always treated as if for regular replication. That is, the `dsi_replication` parameter is off, and the `dsi_keep_triggers` parameter is on.

When there is a database subscription—and table and function replication definitions—but there are no table or function subscriptions:

- If the table and function replication definitions contain the send standby clause, the database subscription honors the table or function replication definition.
- If the table and function replication definitions do *not* contain the send standby clause, all columns and parameters are replicated and the data is converted to the declared column and parameter datatypes.

## Materialization

Database subscription requires either the none method (no materialization) or the bulk method of materialization.

### Subscription without materialization

To create a database subscription when the primary and replicate databases have been synchronized prior to subscription, or activities at the primary database can be suspended, use create subscription with the without materialization clause. See the example in “Replicating the database” on page 408.

When you use the no materialization method, you can materialize the replicate databases using bcp, dump and load, mount and unmount, or other methods. Because Replication Server does not coordinate the initial database synchronization process, you will likely need to suspend database applications. Use this method if you are materializing the replicate database with a cross platform dump and load (XPDL).

### Subscription with bulk materialization

You can use dump and load or manual coordination methods to synchronize databases.

To create a database subscription using dump and load coordination, use define subscription with the use dump marker clause. Both the primary and replicate databases and Replication Servers, must have the same server user ID, password, and role settings.

```
define subscription sub_2
  for database replication definition repdef_1
```

```
with primary at PDS.primary_db
with replicate at RDS.rdb
subscribe to truncate table
use dump marker
```

After you define the subscription:

- 1 Dump PDS.pdb. The DSI connection to the replicate database is suspended when the dump marker reaches the replicate Replication Server. It is suspended so that no data will be replicated until you finished step 2. Replication Server activates and validates the subscription automatically when the dump marker is replicated.

---

**Warning!** Do not activate subscription or it will override the wait for dump marker at the Replication Server.

---

- 2 Load PDS.pdb to RDS.rdb.
- 3 Resume the DSI connection to the RDS.rdb.

## Altering database subscriptions

You cannot alter a database subscription directly. To alter a database subscription, delete the existing database subscription using `drop subscription`, and then create a new one.

## Dropping database subscriptions

You can delete a database subscription using `drop subscription`. You must include the `without purge` option so that Replication Server will not remove rows added by the subscription to the replicate. For example:

```
drop subscription sub_1a
for database replication definition rep_1
with primary at PDS.pdb
with replicate at RDS.rdb
without purge ...
```

Dropping a database subscription does not affect existing table or function subscriptions. Similarly, dropping a table or function subscription does not affect existing database subscriptions. However, in this case, the replicate tables may need to be rematerialized.

See the *Replication Server Reference Manual* for complete syntax and usage information.

## Viewing information about database subscriptions

You can use `rs_helpdbsub` to view information about a specific database subscription or all database replication definitions for a database or a data server.

For example, to view information about the `sub_2B` database subscription, enter:

```
rs_helpdbsub sub_2B, dsA, db
```

For syntax and usage information about `rs_helpdbsub`, see the *Replication Server Reference Manual*.

## Using database, table, and function subscriptions concurrently

If a database subscription and table or function subscriptions coexist, the table or function subscription overrides the database subscription. That is, Replication Server replicates the table or function according to the table or function subscription, not the database subscription.

Database subscriptions do not support the `where` clause or the `for new articles` clause. When using a database subscription, you need to create a table subscription only to:

- Use the `where` clause of a table subscription
- Replicate a table filtered out by the database replication definition

---

**Note** A database subscription supports the `subscribe to truncate table` clause, but not for those tables with a table subscription.

---

- Implement autocorrection on a table

## Creating and dropping subscriptions

When database and table or function subscriptions are used concurrently, take care when creating or dropping those subscriptions.

- If you create a database subscription that references a table with an existing table subscription, make sure you do not overwrite the replicated table when synchronizing databases:
  - a Back up the replicated table.
  - b Use dump and load to synchronize the replicate database.
  - c Copy the replicated table back into the replicate database.
- Drop a table or function subscription with suspension. After the replicate DSI is suspended, you can dematerialize or resynchronize the replicate table or function.
- When a database subscription exists that includes a table and you want to add a table subscription, make sure you define the table subscription using bulk materialization. Defining the table subscription does not stop database replication for the table. Database replication for a table stops when its table subscription is activated.
- When dropping a database subscription, you must manually purge all replicated tables that do not have table subscriptions. Replication Server does not dematerialize replicated tables.

## Replicating the master database in an MSA environment

You can replicate Adaptive Server logins from one master database to another. Master database replication is limited to DDL, and the system commands used to manage logins and roles. Master database replication does not replicate data from system tables, or replicate data or procedures from any other user tables in the master database.

Both the source Adaptive Server, and the target Adaptive Server must be the same hardware architecture type (32-bit versions and 64-bit versions are compatible) and the same operating system (different versions are also compatible).

For a list of supported DDL and system procedures that apply to the master database, see “Restrictions and requirements when using `sp_reptostandby`” and “Supported DDL commands and system procedures” in the *Replication Server Administration Guide Volume 2*.

Replication Server versions 12.0 and later support master database replication with warm standby, and with MSA in Replication Server 12.6 and later. The primary or active Adaptive Server must be version 15.0 ESD #2 and later.

See “Replicating the master database in a warm standby environment” in the *Replication Server Administration Guide Volume 2* for information about master database replication in a warm standby environment.

❖ **Setting up a master database replication in an MSA environment**

- 1 User `rs_init` to set up the primary and replicate master databases.
- 2 Use `bcp` or manually synchronize syslogins and suids at each master database. Do not use dump and load to materialize the replicate master database.
- 3 Mark the primary master database:

```
sp_reptostandby master, 'all'
```
- 4 Stop the RepAgent on the primary master database:

```
sp_stop_rep_agent master
```
- 5 Configure the replication primary master database to send warm standby transactions:

```
sp_config_rep_agent master, 'send warm standby
xacts', 'true'
```
- 6 Restart the RepAgent on the primary master database:

```
sp_start_rep_agent master
```
- 7 Create a database replication definition to replicate the system procedures:

```
create database replication definition master_dbrep
with primary at PDS.master
replicate system procedures
```
- 8 Create a database subscription for each subscribing master database; do not materialize the data:

```
create subscription master_dbsub1
for database replication definition master_dbrep
with primary at PDS.master
with replicate at RDS.master
```

without materialization

## Replicating DDL and system procedures

MSA lets you replicate DDL to nonstandby databases. See Chapter 3, “Managing Warm Standby Applications” in the *Replication Server Administration Guide Volume 2* for a list of DDL commands supported for replication.

To replicate DDL and system procedures:

- Mark the primary database using `sp_reptostandby`.
- Set the `RepAgent` parameter `send warm standby xacts` to `true`—even if there is no standby database.
- Create a database subscription.
- Make sure that both the primary and replicate data servers are the same version of Adaptive Server.

In addition, these constraints apply:

- *When replicating system procedures, and the primary and replicate databases have different names* – filter out the `sp_config_rep_agent` and the `sp_add_user` system procedures in the database replication definition as they use database names as parameters. For example:

```
create database replication definition myrepdef
  with primary at PDS.pdb
  not replicate system procedures in
  (sp_config_rep_agent, sp_add_user)
```

- *When replicating DDL* – the primary and replicate databases must have the same login names and passwords; the DSI uses the original server login name and password to log in to the replicate database.
- *When replicating DDL contained in user-defined transactions* – make sure that the Adaptive Server database option `ddl in tran` is set to `true`. Otherwise, the DSI will shut down when replicating DDL.

Heterogeneous data servers

In a heterogeneous environment, non-Sybase data servers can replicate DDL if the Replication Agent can capture and send DDL commands in Transact-SQL or ANSI SQL (preferred).



## Replicating user stored procedures

To replicate user stored procedures, mark each procedure individually using `sp_setrepproc`. Database replication definitions do not check for owner information for user stored procedures.

## Customizing function strings

You can customize only those function strings that are not in the `rs_default_function_class` function-string class.

For functions with replication-definition scope:

- The DSI uses `rs_default_function_class` for functions that do not use a table or function replication definition with the `send standby` clause.
- Otherwise, the DSI uses the function-string class associated with the connection.

For functions with function-string class scope, the DSI always uses the function-string class associated with the connection.



# Index

## A

- activate subscription command 371, 396, 400
  - for publications 396
  - publication subscription example 400
- Adaptive Server
  - described 28
  - login name for Replication Server access 195
- Add Server wizard 68
- adding
  - ID server domains 89
  - primary keys 301
  - Replication Servers to existing systems 88
  - searchable columns 303
- Advanced Security option 237
- alter applied function replication definition command 333
- alter connection command 128, 166, 172, 226, 227, 236, 321
  - changing maintenance user password 195
  - disabling password encryption 202
- alter database replication definition command 415
- alter function replication definition command 249, 299, 334, 346
- alter replication definition command 322
- alter request function replication definition command 334
- alter route command 92, 106
  - changing passwords 195
  - disabling password encryption 202
- alter user command
  - changing passwords 201
- application models
  - consolidated replicate 12
  - redistributed corporate rollup 14
  - replicated consolidated replicate 14
- applied functions
  - described 335
  - prerequisites for implementing 330
  - setting up 337

- article subscriptions, creating 397
- articles
  - adding to publication 314
  - definition 308
  - displaying information about 313
  - dropping 316
- asynchronous transaction replication 2
- atomic materialization
  - create subscription command for 374
  - described 353, 354
  - text and image columns 386
- autocorrection
  - bulk materialization 363
  - enabling for nonatomic materialization 356, 369
- automatic backup, ERSSD 92

## B

- background processing 59
- backup, ERSSD 101
- bars
  - displaying 59
  - hiding 59
- batch configuration parameter 174
- batch ltl configuration parameter 115
- batch\_begin configuration parameter 174
- bcp utility program 360
- bidirectional replication environment 407
  - creating 70
- bitmap subscriptions 388
- bulk materialization
  - autocorrection 363
  - define subscription command 377
  - described 354, 357
  - for replicated functions 345
  - methods 358
  - publication subscriptions 399
  - refreshing publication subscriptions 401
  - simulating atomic materialization 361

## Index

- simulating nonatomic materialization 363
  - stopping primary updates 359
  - taking a snapshot 359
  - buttons
    - toolbar 59
- ## C
- case, in RCL commands xx
  - certificate authority 237
  - certificates 237
  - certifications
    - component xvii
    - product xvii
  - changing
    - database connections 170
    - existing replication system 88, 90
    - ID Server login name and password 195
    - replication definitions 293
    - Replication Server login names for the RSSD RepAgent 194
    - routes 157
    - RSSD primary or maintenance user 193
    - searchable columns 307
    - user passwords 200
  - character sets, conversion 175
  - check publication command 309, 313
  - check subscription command 371, 379, 396
    - example 403
    - for articles 396
    - for publication subscriptions and articles 403
    - for publications 396
  - CipherSuites 237
  - class-level translations 319
    - existing connections 321
    - new connections 319
    - system-defined variables 322
    - with column-level translations 326
  - client application
    - described 31
  - Client/Server Interfaces (C/SI), client applications for 31
  - column-level translations 322
    - creating 323
    - multiple replication definitions 326
  - columns
    - changing in replicated tables 303, 307
    - deleting from primary or replicate table 306
    - IDENTITY 291
    - rs\_address datatype 388
    - specifying for replication definition 253
  - command\_retry command configuration parameter 174
  - commands
    - configure replication server 99
    - for encrypting passwords 197
    - sysadmin erssd 98, 100
  - Computed columns 288
    - deterministic expressions 288
    - materialized 288
    - nondeterministic expressions 288
    - rs\_set\_dml\_on\_computed function string 289
    - virtual 288
  - concurrency control, described 48
  - configuration file 87
    - Replication Server 87
    - rs\_subcmp program 392
  - configuration parameters
    - dynamic 95
    - ERSSD 99
    - for Replication Server 91
    - send\_standby 418
    - viewing 125
  - configuration star 140
  - configure replication server command 91, 92, 99, 218, 223, 229
  - Configure Replication wizard 67
  - configuring
    - environment with primary and multiple replicates 69
    - replication environment 67
    - standard warm standby environment 67
  - configuring a standard warm standby environment 68
  - connect database configuration parameter 115
  - connect dataserver configuration parameter 115
  - connect source permission 203
  - connecting to Replication Server, using network-based security 230, 231, 232
  - connections
    - creating 71
    - defined 40
    - network-based security for 226

- consistency
    - verifying for subscriptions 391
  - consolidated replicate application model 12
  - Contents tab 55
  - context menus 58
  - context-sensitive menus, shortcut to 58
  - controls
    - dialog box 58
  - create applied function replication definition command 333
  - create article command 309, 311
  - create connection command 166, 168
  - create database replication definition command 415
  - create function replication definition command 334, 338
  - create function string command 366
  - create object permission 203
  - create partition command 47
  - create publication 309, 310
  - create replication definition command 249, 251, 322, 323
  - create request function replication definition command 333
  - create route command 145
  - create subscription command
    - and nonatomic materialization 376
    - 370, 396, 398
    - and atomic materialization 375
    - example for publications 398
    - for publications 312, 396
  - Create Subscription Without Materialization option 69
  - create user command 200
    - adding Replication Server login name for RSSD RepAgent 194
  - creating
    - a replication environment object 65
    - connections 71
    - database connections 167
    - function replication definitions 338, 342
    - multiple ID Server domains 89
    - replicate tables 382
    - replication definitions 71, 242, 250
    - Replication Server login names 200
    - subscriptions 72, 198, 368
  - creating a bidirectional replication environment 70
  - current\_rssd\_version configuration parameter 91
- ## D
- data availability
    - fault tolerance 3
    - local access 3
  - Data Definition Language (DDL) 406
    - replicating 426
  - data limits filter mode configuration parameter 115
  - data server
    - and C/SI support 16
    - ID numbers 90
    - maintenance user login names 20
    - support for heterogeneous 16, 28
    - suspending access to 171
  - Database Administrator, role of 21
  - database connections
    - attributes 167
    - changing attributes of 170
    - creating 167
    - displaying 188
    - dropping 187
    - information for 168
    - managing 163, 183
    - monitoring 188
    - resuming 171
    - suspending 171
  - database logs
    - truncation 50
  - database replication definitions 415
  - database replication filters 417
  - database schema, replication definitions 299
  - Database Subscription Resolution Engine (DSRE) 417
  - database subscriptions
    - altering 422
    - dropping 422, 424
    - managing 420
    - materialization 421
    - with table and function subscriptions 423
  - databases
    - managed by Replication Server 188
    - preparing for replication 163
  - datatype classes

## Index

- rs\_db2\_dt\_class 322
- rs\_mssql\_dt\_class 322
- rs\_oracle\_dt\_class 323
- rs\_sqlserver\_dt\_class 322
- rs\_udb\_dt\_class 323
- datatype definitions 324
  - for DB2 datatypes 325
  - for Microsoft SQL Server datatypes 325
  - for Oracle datatypes 326
- datatypes
  - identity columns 291
  - rawobject and rawobject in row 269, 274
  - rs\_address 291, 388
  - timestamp columns 292
- db\_packet\_size configuration parameter 174
- DB2 databases, function-string class 17
- declared datatypes 323
- define subscription command 371, 396, 399
  - and bulk materialization 377
  - creating publication subscriptions 399
  - for publications 396
  - publications subscription example 400
  - using with replicated functions 340
- Defined Subscription for Bulk Materialization option 69
  - deleting an object 63
- Details list 57
- direct routes 140
- directory services 38
- disconnecting
  - from a replication environment 66, 67
- disk\_affinity configuration parameter 147, 174
- displaying
  - bars 59
  - database connections 188
  - databases with subscriptions 391
  - DSI thread status 189
  - icons 59
  - replication definitions 298
  - RSI thread status 161
  - subscription information 391
  - users of replication system 208
  - users' permissions 209
- displays
  - updating 60
- distributed data models
  - corporate rollup 9
  - custom design 9
  - distributed primary fragments 9
  - redistributed corporate rollup 9
- distributed database system and Replication Server 4
- distributed primary fragments, consolidated replicate application model 14
- distributor thread (DIST) 180
- drop article command 310, 316
- drop connection command 106, 187
- drop database replication definition command 416
- drop function replication definition 347
- drop function replication definition command 333
- drop publication command 310, 315, 316
- drop replication definition command 249, 304
- drop route command 107
- drop subscription command 105, 371, 396, 401
  - example 402
  - example of 402
  - for articles 397
  - for publications 396
  - function replication definitions 381
  - table replication definitions 381
- drop user, dropping login names 201
- drop\_repdef clause 316
- dropping
  - database connections 187
  - databases from the ID Server 187
  - function replication definitions 347
  - primary keys 301
  - replication definitions 304
  - Replication Server login names 201
  - Replication Servers from existing system 105, 109
  - routes 159
  - searchable columns from the searchable columns list 301, 307
  - subscriptions 198
- DSI threads
  - described 46
  - displaying 189
  - scheduler 46
- dsi\_alt\_writetext configuration parameter 174
- dsi\_charset\_convert configuration parameter 175
- dsi\_check\_lock\_wait configuration parameter 175
- dsi\_cmd\_batch\_size configuration parameter 175
- dsi\_cmd\_separator configuration parameter 175

- dsi\_commit\_check\_locks\_intrvl configuration parameter 175
  - dsi\_commit\_check\_locks\_max configuration parameter 176
  - dsi\_commit\_control configuration parameter 176
  - dsi\_exec\_request\_sproc configuration parameter 176
  - dsi\_fadeout\_time configuration parameter 176
  - dsi\_ignore\_underscore configuration parameter 176
  - dsi\_isolation\_level configuration parameter 176
  - dsi\_keep\_triggers configuration parameter 177
  - dsi\_large\_xact\_size configuration parameter 177
  - dsi\_max\_cmds\_to\_log configuration parameter 177
  - dsi\_max\_text\_to\_log configuration parameter 177
  - dsi\_num\_large\_xact\_threads configuration parameter 177
  - dsi\_num\_threads configuration parameter 177
  - dsi\_partitioning\_rule configuration parameter 177
  - dsi\_replication configuration parameter 178
  - dsi\_serialization\_method configuration parameter 178
  - dsi\_sqt\_max\_cache\_size configuration parameter 178
  - dsi\_text\_convert\_multiplier configuration parameters 179
  - dsi\_text\_max\_xacts\_in\_group configuration parameter 177
  - dsi\_xact\_group\_size configuration parameter 179
  - dump and load coordination 70
  - dump command 360
  - dump\_load configuration parameter 179
  - dynamic\_sql configuration parameter 179
  - dynamic\_sql\_cache\_management configuration parameter 180
  - dynamic\_sql\_cache\_size configuration parameter 179
- E**
- enabling RepAgent 113
  - encrypted passwords
    - sending 196
  - encryption
    - disabling for Replication Server 202
    - enabling for Replication Server 201
  - environment
    - three-tier 75
    - two-tier 64
  - error classes
    - Open Server gateway 17
  - errsd\_backup\_dir 99
  - errsd\_backup\_interval 99
  - errsd\_backup\_start\_time 99
  - errsd\_ra 99
  - ERSSD
    - automatic backup 92
    - created by rs\_init program 88
    - media failure, recovery from 103
    - recovery instructions 101
    - recovery procedures 102
    - routing 100
    - use isql to execute 84
  - ERSSD (Embedded Replication Server System Database) 97
  - ERSSD (Embedded Replication Server System Database), configuring 98
  - ERSSD backup directory path 98
  - ERSSD configuration parameters 99
  - ERSSD database file path 98
  - ERSSD transaction log file path 98
  - ERSSD transaction log mirror file path 98
  - ERSSD, backup directory files 101
  - ERSSD, configuration parameters in rs\_init table 91
  - ERSSD, files, moving 100
  - ERSSD, users 100
  - errsd\_backup\_start\_date 99
  - Event Log pane 59
  - event triggers
    - adding 76
  - examples
    - assigning domain ID numbers 90
    - atomic materialization 355
    - replication definition 251
    - routing 148
    - rs\_subcmp configuration file 392
  - exceptions log
    - transactions written to 49
  - exec\_cmds\_per\_timeslice configuration parameter 179
  - exec\_sqm\_write\_request\_limit configuration parameter 180
  - executing

## Index

- RCL commands 82
  - scripts with isql 85
- extended limits 121, 263, 268
  - more columns 264
  - wide columns 264
  - wide data 264
  - wide messages 265

## F

- fault tolerance, achieving 3
- features
  - background processing 59
  - Details list 57
  - Event Log pane 59
  - script editors 60
- files
  - interfaces 38
  - moving, ERSSD 100
  - Replication Server configuration 87
  - Replication Server run file 86
- folder icons 57
- for new articles clause 399
- formatting, RCL commands xix
- function replication definition
  - modifying 346
- function replication definitions
  - altering 346
  - commands for managing 333
  - dropping 347
  - subscribing to 345
- Function strings
  - rs\_set\_dml\_on\_computed 289
- function strings
  - changing replication definitions 299
  - customizing 427
  - defined 42
  - for Java columns 275
  - variables 42
- function-string classes 320
  - defined 43
  - for Adaptive Server databases 17
  - for DB2 databases 17
  - open architecture 17
  - rs\_msss\_function\_class 320

- rs\_oracle\_function\_class 320
- rs\_sqlserver\_function\_class 320
- rs\_udb\_function\_class 320

## G

- grant command 165, 202, 207

## H

- ha failover configuration parameter, RepAgent 115
- ha\_failover configuration parameter 94, 97
- HDS. See heterogeneous datatype support
- help
  - topic 55
- help contents 55
- heterogeneous data servers 426
- heterogeneous datatype support 317–327
  - class-level translations 319
  - column-level translations 322
  - data servers 317
  - function-string classes for 320
  - overview 317
  - procedure for 318
- hiding
  - bars 59
  - icons 59

## I

- icons
  - Adaptive Server xxi
  - client application xxi
  - displaying 59
  - folder 57
  - hiding 59
  - object 57
  - Replication Agent xxi
  - Replication Manager xxi
  - Replication Server xxi
- ID numbers
  - data servers 90
  - Replication Server 90



- ID Server 88
    - adding server domains 89
    - dropping a database from 187
    - guidelines 89
    - ID numbers 90
    - login name 27
    - login name and password 194
    - network-based security for 229
    - requirements 27
    - specifying domain ID numbers 89
  - id\_msg\_confidentiality parameter 229
  - id\_security\_mechanism parameter 229
  - id\_server configuration parameter 91
  - identifiers
    - format xx
    - function parameters xx
    - length xx
  - IDENTITY columns 291
  - image datatype
    - changing replication for 283
    - overview of replication 277
  - inbound queue
    - defined 44
  - inconsistencies
    - correcting 392
    - locating 392
    - occurring in tables 391
    - resulting from skip transaction clause 183
  - indirect routes 141
  - interfaces file 38
    - defined 38
    - requirements 38
  - isql interactive SQL utility 84, 196
    - executing RCL commands 82
    - executing scripts 85
    - to execute ERSSD 84
- J**
- Java datatypes 274
- K**
- keyboard shortcuts 58
- keytab file 218, 230
- L**
- large identifiers 122
  - large messages 268
  - LDAP server 39
    - Open Client/Server 40
  - libtcl.cfg file 213
  - list
    - Details 57
  - LOB datatypes
    - limitations 287
    - partial update 288
  - log
    - event 59
  - log transfer
    - resuming 128
    - suspending 126, 128
  - Log Transfer Language (LTL) 29
  - log truncation, Adaptive Server 50
  - logical connections 68
  - login names
    - for subscriptions 369
    - applied functions 199
    - applied stored procedures 199
    - creating for maintenance user 165
    - creating for Replication Server 200
    - data server 20
    - dependencies 192, 199
    - displaying maintenance user 166
    - dropping Replication Server 201
    - ID Server 27, 194
    - list of commands for managing 200
    - Replication Server 20
    - Replication Server for RSSD RepAgent use 194
    - for Replication Server use 195
    - request functions 199
    - request stored procedures 199
    - RSSD maintenance user 193
    - RSSD primary user 193
    - for subscriptions 198

**M**

- maintenance user 68, 69
  - changing passwords 195
  - described 195
  - displaying list of 166
  - granting database access 166
  - login names 20, 166
  - required permissions 165
  - RSSD 193
- managing
  - replicated tables 241
  - stable queues 45
- mapping security-system login 235
- marking data for replication 414
- master database
  - replication 424
- materialization 69
  - database subscriptions 421
  - MSA 421
- materialization method 68
- materialization methods
  - for function replication definitions 398
  - for publication subscriptions 398
- md\_sqm\_write\_request\_limit configuration parameter 180
- media failure, ERSSD, recovery 103
- memory\_max configuration parameter 182
- menus
  - context 58
- menus and toolbars 58
- minimal columns
  - specifying for replication 251, 258
- minimum\_rssd\_version configuration parameter 92
- mixed version
  - multisite availability 407
- mixed versions
  - replication system 18
- mnemonics 58
- monitoring
  - database connections 188
  - routes 161
- monitoring of status 60
- more columns 264
- mount command 359
- move primary command 106
  - routing requirements 139
- moving ERSSD files 100
- msg\_confidentiality parameter 222
- msg\_integrity configuration parameter 222
- msg\_origin\_check configuration parameter 222
- msg\_replay\_detection configuration parameter 222
- msg\_sequence\_check configuration parameter 222
- multiple replication definitions
  - column-level translations 326
- multisite availability (MSA) 33, 352, 405
  - advantages of 406
  - bidirectional replication environment 407
  - bulk materialization 409
  - concurrent replication definitions 418
  - data replication 414
  - database replication definitions 415
  - dropping database replication definitions 416
  - features of 406
  - function strings 427
  - heterogeneous data servers 426
  - marking data 414
  - master database replication 424
  - mixed version 407
  - replicating tables and functions separately 410
  - replicating the database 408
  - resynchronizing tables 416
  - set up 408
  - warm standby 411
- mutual\_auth 222
- MySybase xvii

**N**

- name space, replication definition 252
- naming replicated tables 305
- net password encryption configuration parameter, RepAgent 115
- network-based security 210–236
  - activating 218
  - altering 233
  - configuring services 219
  - credential 210
  - disabling 233
  - environment variables for 216
  - global settings 224
  - how it works 211

- logging in 230
- mapping login 235
- message protection 211
- parameters 222
- pathways 220, 221
- planning for 223
- potential security breach 236
- requirements 212
- requirements and restrictions 211
- restrictions 212
- setting up 213
- using multiple security mechanisms 236
- viewing information about 234
- new features
  - background processing 59
  - Details list 57
  - dynamic configuration 95
  - Event Log pane 59
  - manual data reconciliation 393
  - rs\_subcmp performance enhancement 393
  - schema comparison 393
  - script editors 60
- no materialization method
  - described 353
  - describing 357
  - requirements for using 357
- nonatomic materialization
  - autocorrection 369
  - described 353, 355
  - text and image columns 386
- num\_threads configuration parameter 182

## O

- object icons 57
- object properties 62
- object property sheets 63
- object tree 56
  - moving through 57
- objectid.dat file 213, 215
- objects
  - creating 65
  - creating in Sybase Central 62
  - deleting 63
  - selecting 57

- online help 54
- Open Client Client-Library 82
- Open Server gateway
  - creating for Replication Server 16
- oserver configuration parameter 92
- outbound queue, defined 45

## P

- parameters
  - RepAgent configuration 88, 115
- parameters, stored procedure
  - adding to replicated functions 346
- partial update
  - LOB datatypes 288
- partitions
  - guidelines for choosing 47
- password encryption
  - extended support 197
  - for maintenance user passwords 197
  - for route user passwords 197
  - for user passwords 197
  - for user passwords in configuration file 197
  - replication system 20
- password\_encryption configuration parameter 202
- passwords
  - alter user command 201
  - applied functions 199
  - applied stored procedures 199
  - changing 200
  - changing for maintenance user 195
  - changing for Replication Server in RSSD RepAgent 194
  - changing for RSSD primary user 193
  - dependencies 195, 199
  - enabling encryption 201
  - encrypted 202
  - ID Server 194, 195
  - for RepAgent use 194
  - for Replication Server use 195
  - request functions 199
  - request stored procedures 199
  - requirements for Replication Server 200
  - subscriptions 198
- performance

## Index

- replicating local data 3
  - routing 143
  - permissions
    - creating subscriptions 368
    - displaying for users 209
    - dropping subscriptions 380
    - for adding Replication Server and Adaptive Server Enterprise 65
    - granting 207
    - granting database access for maintenance user 166
    - maintenance user 165
    - managing for Replication Server 202, 210
    - revoking 208
    - sa, sa\_role, sso\_role 65
    - subscription requirements 369
    - summary of commands for 205
    - system for 20
  - personalized views
    - creating xvii
  - preparing
    - three-tier 75
  - prev\_min\_rssd\_version configuration parameter 92
  - prev\_rssd\_version configuration parameter 92
  - primary data 49
    - failure to update 49
  - primary data server
    - subscription requirements 369
  - primary databases
    - required permissions 167
    - subscription requirements 369
  - primary key
    - adding or dropping 301
    - defined 32, 250
    - requirement for unique 244
  - primary key clause 256
  - primary key columns
    - restrictions on updating 244
  - primary subscribe permission 203
  - primary tables
    - subscription requirements 368
  - primary user
    - RSSD 193
  - principal user 218
  - priority configuration parameter 116
  - processing
    - background 59
  - properties
    - object 62
  - property sheets 62, 63
  - publication subscriptions 394, 403
    - activating 400
    - bulk materialization method 399
    - creating 397
    - defining 399
    - definition 308
    - dropping 401
    - monitoring 402
    - refreshing 399, 401
    - restrictions 395
    - specifying materialization methods 398
    - status information 403
    - validating 400
  - publications 316
    - adding articles 315
    - altering 314
    - creating at the command line 310
    - definition 308
    - displaying information about 313
    - dropping 315
    - dropping replication definitions 315
    - for stored procedures 348
    - from Sybase Central 309
    - procedure for creating 308
    - RCL commands for 309
    - viewing information about 313
  - published datatype 323
- ## Q
- queue data
    - accessing 72
    - viewing 72
  - quiescing
    - procedure for Replication Server 104
    - replication system 104
- ## R
- RCL commands
    - activate subscription command 371

- alter applied function replication definition
    - command 333
  - alter connection command 166, 172
  - alter function replication definition command 334
  - alter function string command 348
  - alter replication definition command 249, 299
  - alter request function replication definition
    - command 334
  - check subscription command 371, 379
  - create applied function replication definition
    - command 333
  - create connection command 168
  - create function replication definition command 334, 338, 342
  - create replication definition command 249, 251
  - create request function replication definition
    - command 333
  - create route command 145
  - create subscription command 370, 375, 376
  - define subscription command 371, 377
  - drop connection command 187
  - drop function replication definition command 333, 347
  - drop replication definition command 249
  - drop route command 159
  - drop subscription command 371, 380
  - drop user command 201
  - executing command 82
  - grant command 207
  - resume connection command 183
  - resume route command 150
  - revoke command 208
  - set autocorrection command 369
  - shutdown command 87
  - suspend connection command 171
  - suspend log transfer command 126
  - suspend route 150
  - sysadmin dropdb command 188
  - sysadmin purge\_route\_at\_replicate command 160
  - table of permissions 205, 207
  - validate subscription command 371
- RCL, formatting commands xix
- recovery instructions, ERSSD 101
  - recovery procedures, ERSSD 102
  - recovery, from media failure, ERSSD 103
  - redistributed corporate rollup application model 14
  - Rep Agent user 68
  - rep\_as\_standby configuration parameter 180
  - RepAgent 111
    - configuration parameters 114, 115
    - described 29
    - disabling 119
    - enabling for databases 113
    - enabling on Adaptive Server 112
    - error messages 119
    - extended limits 121
    - network security for 120
    - parameters 88
    - for RSSD 194
    - secondary truncation point 50
    - setting up 112
    - starting 118
    - starting up 118
    - status information 124
    - stopping 118, 119
    - suspending 126
    - thread status 125
    - thread user status 126
    - truncation point 50
  - RepAgent Executor 179
  - RepAgent user 69
  - replicate databases
    - upgrading to primary databases 184
  - replicate Replication Server
    - subscription requirements 369
  - replicate tables
    - requirements for subscriptions 368
  - replicated consolidated replicate application model 14
  - replicated function
    - creating 348
    - modifying 348
  - replicated functions
    - adding parameters 346
    - adding searchable parameters 346
    - described 35, 334
    - dropping 345
    - modifying 345
    - subscribing to 339
  - replicated stored procedures
    - enabling for replication 344

## Index

- login and password dependencies 199
- replicated function delivery 335
- replicated tables
  - changing 305
  - changing searchable columns 307
  - commands for modification 249
  - dropping 305
  - enabling for replication 270
  - failed updates 49
  - procedures for changing 305
  - renaming primary and replicate copies 305
  - requirements 32
  - subscribing to 352
- replicating
  - encrypted columns 289
  - large objects to non-ASE servers 279
  - LOB datatypes 287
  - master database in an MSA 424
  - partitioned tables 373
  - timestamp columns 292
- Replicating computed columns 288
- replication
  - ASE shared-disk cluster 123
- Replication Agent
  - described 28
  - open architecture 16
  - requirements 30
- replication definitions
  - for distributed primary fragments 12
  - changing 293
  - commands for managing 249
  - creating 250
  - datatypes 253
  - defined 33
  - described 250
  - dropping 304
  - dropping from articles 316
  - dropping from publications 315
  - examples 251
  - extended limits 263
  - functions 335
  - name space 252
  - for distributed primary fragments 9
  - primary key 256
  - requirements for creating subscriptions 368
  - rs\_address datatype 388
  - searchable columns 257
  - text and image columns 261
  - text or image columns 278
  - using 250, 304
- replication environment
  - configuring 67
  - configuring with primary and multiple replicates 69
  - disconnecting from 66, 67
  - setting up 64
- replication environment object
  - creating 65
- Replication Server
  - adding to an existing system 88
  - advantages of 3
  - configuration file 87, 225
  - configuring rs\_config system table 90
  - connections 38
  - described 26
  - distributed data models 9
  - dropping from existing system 105, 109
  - executable program 86
  - general description 1
  - and heterogeneous data servers 16
  - ID numbers 90
  - introduction 1
  - list of databases managed by 188
  - login name for Adaptive Server use 195
  - login name for RSSD use 195
  - managing 79, 104
  - managing login names 199
  - managing objects 70
  - objects 70
  - permissions 202, 208
  - primary copy model 6
  - quiescing 104
  - role in a distributed database system 4
  - run file for 86
  - security 210
  - shutting down 87
  - starting 86
  - subscription requirements 369
  - system data flow 7
  - technical overview 25
  - transaction handling 43
- Replication Server programs

- repserver 86
- rs\_init 88
- rs\_subcmp 392
- Replication Server System Database (RSSD)
  - described 29
  - login names 193
  - maintaining 90
  - managing 95
  - RepAgent for 194
  - requirements 30
  - rs\_helpdb stored procedure 189
  - rs\_maintusers 209
  - rs\_users 209
  - system tables 29
  - users 193
- replication system
  - components 25
  - creating multiple domains 89
  - domains 89
  - open architecture 16
  - quiescing 104
  - roles and responsibilities 21, 22
  - security 191
  - setting up 79
- Replication System Administrator
  - role of xiii, 21
- repserver command 86
- request functions
  - defined 35
  - described 336
  - login names and passwords 199
  - permissions needed at primary 167
  - prerequisites for implementing 330
  - setting up 340
- request stored procedures
  - login names and passwords 199
  - primary copy model 6
- restrictions
  - on replicated data 244
- resume connection command 183, 227, 364
- resume log transfer command 126
- resume route command 150
- resuming
  - log transfer 127
  - RepAgent 127
  - routes 150
- resynchronizing replicate tables 416
- retry timeout configuration parameter 116
- revoke command 202, 208
- RMS
  - adding servers 76
  - connecting to 75
  - monitoring replication environment 75
  - three-tier management solution 64
  - viewing objects 76
- RMS (Replication Monitoring Services) 32
- roll-up
  - consolidated replicate application model 12
  - consolidated replicate as primary application model 14
- route version
  - between Replication Servers 160
- routes
  - changing 151, 156
  - creating 144
  - creating login names 146
  - defined 40
  - determining 138
  - direct 140
  - dropping 158
  - indirect 141
  - managing 137, 158
  - monitoring creation of 161
  - network-based security for 227
  - purging 108
  - requirements 138
  - resuming 150
  - subscriptions 143
  - suspending 150
  - unsupported 144
  - upgrading 160
- routing
  - ERSSD 100
  - examples 148
  - overlapping subscriptions 143
  - schemes 140
- row migration
  - text and image columns 387
- rs name configuration parameter 116
- rs password configuration parameter 116
- rs username configuration parameter 116
- rs\_address datatype 291

## Index

- rs\_config system table 93
  - rs\_databases system table 188
  - rs\_helpdb stored procedure 105, 189
  - rs\_helppub stored procedure 310, 313, 403
  - rs\_helpprepdb stored procedure 391
  - rs\_helpproute stored procedure 162
  - rs\_helpuser stored procedure 209
  - rs\_idnames system table
    - dropping database from 188
  - rs\_init
    - creates ERSSD 88
  - rs\_init program 91
  - rs\_lastcommit system table 183
    - permissions 166
  - rs\_maintusers system table 209
  - rs\_marker stored procedure 167
  - rs\_set\_dml\_on\_computed function string 289
  - rs\_setproxy function string 232
  - rs\_subcmp program 392
  - rs\_subscriptions system table 400
  - rs\_update\_lastcommit stored procedure 167
  - rs\_users system table 209
  - RSI threads
    - described 46
    - displaying 161
  - rsi\_batch\_size configuration parameter 147
  - rsi\_fadeout\_time configuration parameter 147
  - rsi\_packet\_size configuration parameter 147
  - rsi\_sync\_interval configuration parameter 147
  - rsi\_xact\_with\_large\_msg configuration parameter 147
  - RSSD 65
    - network-based security for 225
  - rssd\_error\_class configuration parameter 92
  - run file, Replication Server 86
- S**
- sa permission xiii, 65, 203, 205
  - sa\_role permission 65
  - save\_interval configuration parameter
    - for database connection 180
    - for route 147
  - scan batch size configuration parameter 116
  - scan timeout configuration parameter 116
  - schema cache growth factor configuration parameter 116
  - script editors 60
  - scripts
    - executing in isql 85
    - replication definition examples 382
  - searchable columns
    - adding searchable columns 303
    - dropping from the searchable columns list 301, 307
  - searchable columns clause 251
  - searchable parameters
    - adding to replicated functions 346
  - secondary truncation point
    - and disabling RepAgent 119
    - described 50, 51
  - secure sockets layer 237
  - security
    - network-based 210
    - RepAgent 120
    - Replication Server 191, 210
    - replication system 20
  - security mechanisms
    - CyberSafe Kerberos 212
    - DCE 214
    - Transarc DCE 212
  - security services, configuring 219
  - security, network-based 199–236
  - security\_mechanism 223
  - select command 360
  - select with holdlock clause 398
  - send buffer\_size configuration parameter 116
  - send maint xacts to replicate configuration parameter 116
  - send structured opids configuration parameter 117
  - send warm standby xacts configuration parameter 117
  - send\_emc\_pw configuration parameter 92
  - send\_enc\_password configuration parameter 196
  - send\_standby configuration parameter 418
  - set autocorrection command 369
  - set proxy command 217, 232
  - setting up network-based security 213
  - short ltl keywords configuration parameter 117
  - shortcut menus 58
  - shutdown RCL command 87
  - shutting down



- Replication Server 87
- skip ltl errors configuration parameter 117
- skip transaction clause 183
- skip unsupported features configuration parameter 117
- sp\_role system procedure 165
- sp\_setreplicate system procedure
  - marking rs\_marker for replication 186
- sp\_setrepproc system procedure 337
  - marking stored procedures for replication 344
  - using for applied function 337
  - using for request function 342
- sp\_stop\_rep\_agent command 106, 119
- SSL 237
  - certificate authority 237
  - enabling on Replication Server 239
  - on Replication Server 238
  - requirements for 238
  - setting up 239
  - trusted roots file 237
- SSL handshake 238
- SSL security 237
- sso\_role permission 65
- stable queues
  - atomic materialization 355
  - described 44
  - disk files 48
  - management 45
  - requirements 47
  - for routes 144
- star configuration
  - described 140
- starting
  - Replication Server 86
  - Sybase Central 53
- startup delay configuration parameter 117
- status
  - monitoring 60
- status bar 59
- stopping Sybase Central 54
- stored procedures
  - marking for replication using sp\_setrepproc 344
  - publications 349
  - rs\_helpdb 189
  - rs\_helprep 298
  - rs\_helprepdb 391
  - rs\_helproute 161
  - rs\_helpsub 391
  - rs\_helpuser 209
  - rs\_update\_lastcommit 167
- style conventions xviii
- sub\_sqm\_write\_request\_limit configuration parameter 180
- subscribe to truncate table clause 397
- subscribing
  - to function replication definitions 345
  - to replicated tables 352
- subscriptions
  - primary fragments 12
- subscription dematerialization
  - methods 380
  - phases 367
  - processing 365
  - with purge 365
- subscription materialization 180
  - defined 33
  - methods 368
  - phases 366
  - text and image columns 385
- subscription materialization queue
  - defined 45
- subscription migration
  - rs\_address columns 390
- subscription resolution engine (SRE) 417
- subscriptions
  - adding to a publication subscriptions 399
  - bitmap 388
  - commands for managing 370
  - defined 352
  - displaying 391
  - dropping 198, 380, 424
  - login names and password dependencies 198
  - overlapping 143
  - permissions for creating 203
  - preparations for creating 368
  - primary fragments 9
  - removing rows manually 381
  - requirements 368
  - user permission requirements 204
  - verifying consistency 391
- support
  - bidirectional replication 407

## Index

- extended password encryption 197
    - for ASE shared-disk cluster 123
    - longer identifiers 122
  - suspend connection command 171, 227
  - suspend log transfer command 126
  - suspend route command 150
  - suspending
    - database connections 171
    - routes 150
  - Sybase Central 83
    - creating an object in 62
    - online help 54
    - shortcut menus 58
    - stopping 54
    - toolbar 59
  - sysadmin dropdb command 107, 109, 188
  - sysadmin droprs command 107
  - sysadmin erssid, command 98, 100
  - sysadmin purge\_route\_at\_replicate command 160
  - system procedures
    - replicating 426
    - sp\_reptostandby 408
    - sp\_setreplicate 278
    - sp\_setrepproc 344
    - sp\_setreptable 270
  - system tables
    - described 29
    - rs\_config 91
    - rs\_databases 188
    - rs\_idnames 187
    - rs\_lastcommit 183
- ## T
- tables
    - creating for replication 242
    - marking for replication 280
    - materialization options 69
    - procedure for replicating 246
    - subscription requirements 369
  - tabs
    - help contents 55
  - text datatype
    - changing replication for 283
    - overview of replication 277
  - threads
    - DSI scheduler 46
    - RSI 46
  - three-tier environment
    - preparing 75
  - three-tier management solution 64
    - RMS 64
  - timestamp datatypes 292
  - toolbar 59
    - buttons 59
    - hiding 59
  - toolbar button help 55
  - tooltips 55
  - topic help 55
  - transactions
    - handling suspended 171
    - handling with Replication Server 43
    - modifying data in multiple data servers 49
    - skip transaction clause 183
  - triggers
    - event 76
  - truncate table command
    - enabling replication of 399
    - Transact-SQL 397
  - truncation of Adaptive Server database logs 50
  - trusted roots file 237
  - two-tier environment
    - preparing 64
  - two-tier management solution 64
- ## U
- unified\_login 223
  - upgrading routes 160
  - use\_batch\_markers configuration parameter 181
  - use\_index 281
  - user
    - maintenance 68
    - RepAgent 68
  - user stored procedures
    - replicating 427
  - users
    - maintenance 69
    - RepAgent 69
  - users, displaying replication system 208

users, ERSSD 100

## V

validate publication command 309, 311, 313  
 validate subscription command 371, 396, 400  
 variables  
   function strings 42  
 verify password clause 201  
 verifying translations 327  
 version number for route 160  
 versions, replication system 18  
 viewing object properties 62  
 visual monitoring of status 60

## W

warm standby applications  
   MSA advantages 407  
   switchover with MSA 413  
   using MSA 411  
 warm standby environment  
   configuring 67  
 where clause  
   altering 314  
   for create subscription command 371  
   operators 312  
   syntax 312  
   used in articles 312  
 wide columns 264  
 wide data 264  
 wide messages 265  
 with nowait clause 171  
 with primary table named clause 252  
 with purge clause 402  
 with replicate table named clause 252  
 without holdlock clause 376  
 without purge clause 402  
 wizards  
   Add Server 68  
   Configure Replication 67  
 writetext 279

