Administration Guide

Replication Agent™ 15.7.1 SP100

Linux, Microsoft Windows, and UNIX

DOCUMENT ID: DC38261-01-1571100-01
LAST REVISED: May 2013
Copyright © 2013 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.
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 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.
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.
Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates 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

Conventions

These style and syntax conventions are used in Sybase® documentation.

Style conventions

Key	Definition
`monospaced (fixed-width)`	• SQL and program code • Commands to be entered exactly as shown • File names • Directory names
`italic monospaced`	In SQL or program code snippets, placeholders for user-specified values (see example below).
italic	• File and variable names • Cross-references to other topics or documents • In text, placeholders for user-specified values (see example below) • Glossary terms in text
bold sans serif	• Command, function, stored procedure, utility, class, and method names • Glossary entries (in the Glossary) • Menu option paths • In numbered task or procedure steps, user-interface (UI) elements that you click, such as buttons, check boxes, icons, and so on

If necessary, an explanation for a placeholder (system- or setup-specific values) follows in text. For example:

Run:

```
installation directory\start.bat
```

where *installation directory* is where the application is installed.

Syntax conventions

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

Case-sensitivity

- All command syntax and command examples are shown in lowercase. However, replication command names are not case-sensitive. For example, **RA_CONFIG**, **Ra_Config**, and **ra_config** are equivalent.
- Names of configuration parameters are case-sensitive. For example, **Scan_Sleep_Max** is not the same as **scan_sleep_max**, and the former would be interpreted as an invalid parameter name.
- Database object names are not case-sensitive in replication commands. However, to use a mixed-case object name in a replication command (to match a mixed-case object name in the primary database), delimit the object name with double quote characters. For example: **pdb_get_tables** "*TableName*"
- Identifiers and character data may be case-sensitive, depending on the sort order that is in effect.
  - If you are using a case-sensitive sort order, such as “binary,” you must enter identifiers and character data with the correct combination of uppercase and lowercase letters.
  - If you are using a sort order that is not case-sensitive, such as “nocase,” you can enter identifiers and character data with any combination of uppercase or lowercase letters.

Terminology

Replication Agent™ is a generic term used to describe the Replication Agents for Adaptive Server® Enterprise, Oracle, Microsoft SQL Server, and IBM DB2 for Linux, Unix and Windows. The specific names are:

- RepAgent – Replication Agent thread for Adaptive Server Enterprise
- Replication Agent for Oracle

- Replication Agent for Microsoft SQL Server
- Replication Agent for IBM DB2 UDB

About Replication Agent

Replication Agent™ extends the capabilities of Replication Server® by allowing non-Sybase (heterogeneous) data servers to act as primary data servers in a replication system based on Sybase replication technology.

Basic Replication System Concepts

Use transaction replication to maintain data in separate databases called replicate databases. Replicate databases contain accurate, current copies or subsets of data from a primary database.

When a table in the primary database is marked for replication, transactions that change the data in that table are captured for replication. The primary database processes the transaction, and a copy of the transaction (including all its operations) is stored in the transaction log.

In the case of a stored procedure marked for replication, when the stored procedure is invoked in the primary database, all parameter values provided with the procedure invocation are captured and recorded in the transaction log. When a marked stored procedure generates a transaction that affects data in marked tables in the primary database, the transaction generated by the stored procedure is ignored, so only the procedure invocation is replicated.

Transaction Replication

The events captured for replication through a Sybase replication system are referred to as transactions, even if they do not directly correspond to an actual transaction in the primary database. For example, if a transaction affects both marked tables and unmarked tables, only the operations that affect the marked tables are captured for replication. Operations on unmarked tables are ignored.

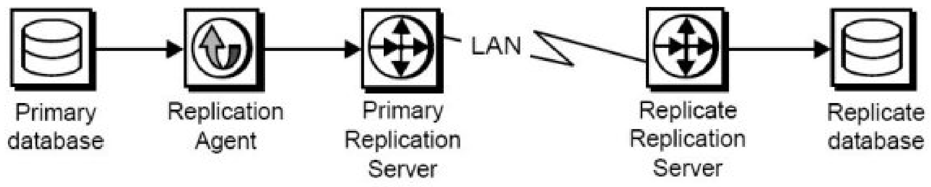
All data-change operations captured for replication exist within a transaction context, that is, only committed transaction operations are replicated; transactions that are rolled back are not replicated.

Even if the data-change events replicated through a Sybase replication system are really operations, those operations are grouped in an atomic collection, and they represent the results of a committed transaction in the primary database.

Replication System Components

The basic components of a replication system are the primary database, the replicate database, and Replication Server.

Figure 1: Typical Sybase Replication System



Primary Databases

A primary database is the source of transactions that modify data in the replicate databases. Transactions are replicated by table or by procedure.

Tables marked for replication in a primary database are called primary tables. A primary table must be marked for replication so that the Replication Agent can identify and replicate the transactions that affect the data in that table. By default, Large-object (LOB) columns within a primary table are marked for replication.

To replicate invocations of a stored procedure, the procedure must be marked for replication so that the Replication Agent can identify and replicate invocations of that procedure in the primary database.

Replication Agents

A Replication Agent is the Sybase replication system component that captures the replicated transactions in a primary database and sends those transactions to a Replication Server for distribution to replicate databases.

Replication Agent reads a transaction log in the primary database and generates Log Transfer Language (LTL) output. LTL is the language that Replication Server uses to process and distribute replicated transactions throughout a replication system.

To provide more sophisticated replication features and generate more efficient LTL, you can configure Replication Agent to use information stored in the Replication Server System Database (RSSD) of the primary Replication Server.

Replication Agent retrieves the information it needs for transaction replication from the native transaction log maintained by the primary data server.

Replication Agent uses the log-based solution for primary databases in these primary database types:

- Oracle
- Microsoft SQL Server
- IBM DB2 for Linux, UNIX, and Windows

Note: Procedure and DDL replication is not available for IBM DB2.

Replication Servers

The Replication Server that receives replicated transactions from a primary database (that is, directly from a Replication Agent) is called the primary Replication Server. The Replication Server that sends replicated transactions to a replicate database is called the replicate Replication Server.

Note: In a simple replication system, a single Replication Server can act as both the primary Replication Server and the replicate Replication Server.

After it receives LTL from a Replication Agent, the primary Replication Server sends the replicated transaction to a replicate database, either directly or through a replicate Replication Server. The replicate Replication Server converts the replicated transaction from the LTL it receives to the native language of the replicate database, and then it sends the replicated transaction to the replicate data server for processing. When the replicated transaction is processed successfully by the replicate database, the replicate database is synchronized with the primary database.

Each Replication Server holds transaction operations in a stable queue and delivers them as soon as possible to other Replication Servers or replicate databases. By doing this, Replication Server guarantees that every transaction successfully received from a Replication Agent is guaranteed to be delivered to appropriately subscribing replicate databases.

Each Replication Server uses a database called the Replication Server System Database (RSSD) to store replication system data and metadata. Replication Agent can use some of the information stored in the RSSD to provide advanced replication features.

Replication Agent Functionality

Replication Agent supports transaction replication from a primary database through Replication Server.

Note: See the *Replication Server Options Release Bulletin* for information about the specific versions of Oracle, Microsoft SQL Server, and IBM DB2 that Replication Agent supports.

Replication Agent runs as a standalone application, independent of the primary data server, the primary Replication Server, and any other replication system components.

Replication Agents can reside on the same host machine as the primary database or any other replication system component, or they can reside on a machine separate from any other replication system components that has network access to the primary database. Replication

Agents must execute on a machine that has access to the transaction logs for the primary database. For Replication Agent for UDB, this is accomplished using the IBM DB2 client.

Replication Agent is compatible with Replication Manager (RM). Replication Agent instances can be configured, managed, and monitored by RM. In addition, you can completely configure, manage, and monitor a Replication Agent instance using any Open Client™ application that is capable of communicating with the Sybase Tabular Data Stream™ (TDS) protocol (such as **isql**).

Replication Agent Instances

You must create an instance of Replication Agent for each primary database from which you want to replicate transactions. Each Replication Agent instance is an independent application with its own configuration and log files, administration port, and connections to the primary database and the primary Replication Server.

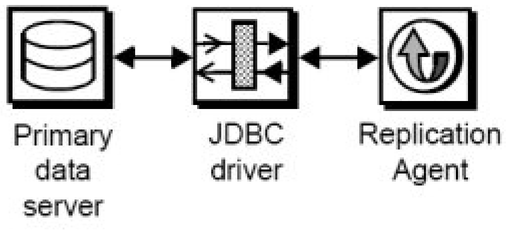
Replication Agent instances created for a specific primary database type are referred to in this book as:

- Oracle Database Server – Replication Agent for Oracle
- Microsoft SQL Server – Replication Agent for Microsoft SQL Server
- IBM DB2 for Linux, UNIX, and Windows – Replication Agent for UDB

Replication Agent Communications

Replication Agent uses the Java Database Connectivity (JDBC™) protocol for all communications. However, some supported databases require the Open Database Connectivity (ODBC) protocol. When connecting to a primary database, Replication Agent connects to either the JDBC driver or the JDBC/ODBC bridge provided by the database vendor.

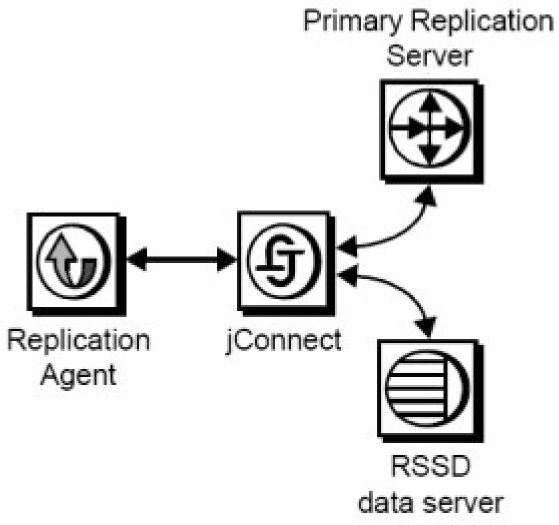
Figure 2: Replication Agent Primary Database Communication Using a JDBC driver



Replication Agent uses the Sybase JDBC driver (jConnect™ for JDBC) to communicate with all Open Client and Open Server™ applications, such as Adaptive Server Enterprise and Replication Server. Each Replication Agent instance uses a single instance of jConnect for JDBC.

While replicating transactions, the Replication Agent maintains connections with both the primary database and the primary Replication Server, and it may occasionally connect to the RSSD of the primary Replication Server to retrieve replication definition data.

Figure 3: Replication Agent Communication with Replication Server and its RSSD



Replication Agent Components

Replication Agent consists of a set of components that work together to perform all the operations required to propagate transactions from a primary database for replication.

These are the main Replication Agent components:

- Log Reader – reads the transaction log in the primary database to retrieve transactions for replication.
- Log Transfer Interface (LTI) – generates Log Transfer Language (LTL) and sends it to the primary Replication Server.
- Log Administrator – administers the Replication Agent transaction log and manages transaction log objects.
- Log Transfer Manager (LTM) – manages all the other components and coordinates their operations and interactions.

In this process:

1. The Log Reader component retrieves transaction data from the transaction log in the primary database.
2. The Log Reader generates change set data and passes the change sets to the LTI.

3. The LTI component processes the change set data from the Log Reader and generates the LTL to send to the primary Replication Server.

   The LTI component also receives messages from the primary Replication Server.

Although the LTM component is not involved in the flow of data from the primary database to the primary Replication Server, it coordinates the activities of the other Replication Agent components and processes any errors generated by those components.

Administration Port

Replication Agent provides an administrative user interface through its administration port.

The administration port allows an Open Client application to log in to a Replication Agent instance as if the Replication Agent were an Open Server application. After it is logged in, the administrative client can issue commands to control, administer, and monitor the Replication Agent instance.

The administration port communicates with the client through the Sybase JDBC driver (jConnect for JDBC).

The administration port passes commands from the administrative client to the Replication Agent components. The administration port also processes the messages from Replication Agent components, and passes those messages out to the client.

Java Requirement

Replication Agent includes a Java Runtime Environment (JRE), so the computer that acts as the Replication Agent host machine must meet JRE requirements.

For more information on installing and setting up a JRE, see:

- *Replication Agent Installation Guide*
- *Replication Server Options Release Bulletin*

Multiple Replication Agents

Replication Agent for Oracle supports the use of multiple Replication Agent instances with Replication Server Multi-Path™ Replication.

Multiple Replication Agent instances allow transactions to be replicated in parallel along multiple independent paths, increasing replication throughput and performance and reducing resource contention. See the *Replication Agent Primary Database Guide*.

Setup and Configuration

Set up Replication Agent after installation, verify that your replication system is ready to replicate data, and start replication.

Note: The procedures described here assume you have already installed the Replication Agent software and Replication Server software, as described in the *Replication Agent Installation Guide* and the Replication Server installation and configuration guides for your platform.

Replication Agent Instance Creation

After you install the Replication Agent software, you must create one instance of the Replication Agent for each primary database that you want to replicate transactions from.

Each Replication Agent instance is an independent process, with its own instance directories to contain its configuration file, system log files, and script files. In addition, each Replication Agent instance creates some tables and stored procedures in the primary database. Replication Agents for Oracle, IBM DB2 UDB, and Microsoft SQL Server also create objects in the Replication Agent System Database (RASD). Each Replication Agent instance manages its own connections to the primary data server, primary Replication Server, and RSSD.

When you create a Replication Agent instance, you must specify:

- A unique instance (server) name
- A unique client socket port number for its administration port
- The type of primary database the instance supports
- A Replication Agent administrator login name for the Replication Agent instance
- A password for the new administrator login

You can create and run more than one Replication Agent instance on a single host machine, but each instance must have a unique name and a unique port number.

See also

- *ra_admin* on page 16
- *Creating a Replication Agent Instance Using the Command Line* on page 29
- *Creating an Instance with a Resource File* on page 23

Replication Agent Instance Directories

Replication Agent instance directories are created under the Replication Agent base directory when you create a Replication Agent instance.

The Replication Agent base directory (`RAX-15_5`) and the installation directory (`sybase`) are created when you install the Replication Agent software.

Note: A single installation (on a single host machine) can support multiple Replication Agent instances. Each instance directory resides under the Replication Agent base directory created when you install the software.

The permissions on all files and directories in the `RAX-15_5` installation directory are set to 600 (read/write for user, no permissions for group and other) or 700 (read/write/execute for user, no permissions for group and other) respectively, to improve security.

Table 1. Permissions on Replication Instance Files and Directories

File/Directory	Permission
`$SYBASE/RAX-15_5/bin/ra_admin.sh`	700
`$SYBASE/RAX-15_5/<instance_name>/<instance_name>.cfg` where *<instance_name>* is the name of your Replication Agent instance.	600
`$SYBASE/RAX-15_5/<instance_name>/log`	700
`$SYBASE/RAX-15_5/<instance_name>/RUN_<instance_name>.sh`	700

Replication Agent Utilities

The utilities provided with Replication Agent are **agt_service**, **ra**, and **ra_admin**.

- **agt_service** – runs a Replication Agent instance as a Windows service.
- **ra** – starts a Replication Agent instance, or returns the Replication Agent software version number.
- **ra_admin** – creates, copies, verifies, and deletes Replication Agent instances, or lists all verifiable, installed Replication Agent instances on a machine.

Replication Agent utilities are supplied as batch files for Windows platforms and as shell scripts for UNIX platforms. The utility files reside in the `bin` subdirectory, under the Replication Agent base directory.

Note: On Windows platforms, when you execute a `run` script, you can omit the extension: **ra -i my_ra**. However, on UNIX, you must always include the extension: **ra.sh -i my_ra**.

You can use the **-help** option with either **ra_admin** or **ra** to obtain information about that utility.

See also

- *The Command Line Interface* on page 14

Preparing to Use the Utilities

Perform these tasks before using the Replication Agent utilities.

Prerequisites

- Log in to the operating system on the Replication Agent host machine with a user login that has **execute** permission in the Replication Agent installation directory and all subdirectories (for example, the "sybase" user)
- Set the SYBASE environment before you invoke any utilities

The SYBASE environment script is supplied as a batch file for Microsoft Windows platforms (`SYBASE.bat`) and as a shell script for UNIX platforms (`SYBASE.sh` or `SYBASE.csh`).

Task

1. Log in to the operating system on the Replication Agent host machine with a user login that has the appropriate permissions.
2. Open an operating system command window.
3. At the operating system prompt, navigate to the Replication Agent installation directory:
   - On Microsoft Windows:
     ```
     cd c:\sybase
     ```
     where `c:\sybase` is the path to the Replication Agent installation directory.
   - On UNIX:
     ```
     cd /opt/sybase
     ```
     where `/opt/sybase` is the path to the Replication Agent installation directory.
4. In the Replication Agent installation directory, invoke the *SYBASE* environment script:
   - On Microsoft Windows:
     ```
     SYBASE
     ```
   - On UNIX, for Bourne and Korn shells:
     ```
     . SYBASE.sh
     ```
   - On UNIX, for C-shell:
     ```
     source SYBASE.csh
     ```

Note: On UNIX platforms, you can insert the source command line in the `.login` file for the Replication Agent administrator (or "sybase" user), so that the SYBASE environment is set automatically when you log in to the Replication Agent host machine.

The Command Line Interface

Use the command line interface to administer Replication Agent from the Windows or UNIX command line.

agt_service

Runs Replication Agent as a Windows Service.

Syntax

```
agt_service[.cmd]
[-install inst_name [-startup state] [-charset charset] [-maxmem
mem] ] |
[-remove inst_name ] |
[-start inst_name ] |
[-stop inst_name ] |
[-h]
```

Parameters

- **-install *inst_name* –** Specifies a Replication Agent instance for which to create a Windows service, where *inst_name* is the name of an existing Replication Agent instance.
- **-startup *state* –** (Optional) Specifies a start-up state for the Replication Agent instance.

  Valid **-startup** *state* values are:

  - **-admin** – (Default) starts the Replication Agent instance in Admin state.
  - **-replicate** – starts the Replication Agent instance in Replicating state.
- **-charset *charset* –** (Optional) Java character that the JVM uses. This option corresponds to the RA_JAVA_DFLT_CHARSET environment variable.
- **-maxmem *mem* –** (Optional) maximum amount of memory that this Replication Agent instance uses. This option corresponds to the RA_JAVA_MAX_MEM environment variable.
- **-remove *inst_name* –** Specifies a Replication Agent instance for which to remove a Windows service, where *inst_name* is the name of an existing Replication Agent instance.
- **-start *inst_name* –** Specifies a Replication Agent instance for which to start a Windows service, where *inst_name* is the name of an existing Replication Agent instance.
- **-stop *inst_name* –** Specifies a Replication Agent instance for which to stop a Windows service, where *inst_name* is the name of an existing Replication Agent instance.
- **-h *inst_name* –** Returns command usage information.

Examples

- **Example 1 –** Creates a Windows service for a Replication Agent instance named “my_ra” in Replicating state:

```
agt_service -install my_ra -startup replicate
```

- **Example 2 –** Removes a Windows service for a Replication Agent instance named "my_ra":

  ```
  agt_service -remove my_ra
  ```

- **Example 3 –** Starts a Windows service for a Replication Agent instance named "my_ra":

  ```
  agt_service -start my_ra
  ```

- **Example 4 –** Stops a Windows service for a Replication Agent instance named "my_ra":

  ```
  agt_service -stop my_ra
  ```

Usage

Use **agt_service**:

- Create a Windows Service to run a specified Replication Agent instance
- Remove a Windows Service instance running a Replication Agent instance
- Start a Windows service, thereby starting the corresponding Replication Agent instance
- Stop a Windows service, thereby stopping the corresponding Replication Agent instance

To run **agt_service**, invoke it as a command at the operating system prompt.

See also

- *Replication Agent Instance Shutdown* on page 69
- *Replication Agent Start-Up* on page 33

ra

Starts Replication Agent.

Syntax

```
ra [-help | -i inst_name [-state] | -v]
```

Parameters

- **-help –** Returns command usage information.

  Note: You can also invoke **ra** with no option specified to return command usage information.

- **-i *inst_name* –** Specifies a Replication Agent instance to start, where *inst_name* is the name of an existing Replication Agent instance.
- **- *state* –** Keyword that specifies a start-up state for the Replication Agent instance.

  Valid **-** *state* values are:

  - **-admin** – (Default) starts the Replication Agent instance in Admin state.
  - **-replicate** – starts the Replication Agent instance in Replicating state.
- **-v –** Returns the Replication Agent software version number.

Examples

- **Example –** Starts a Replication Agent instance named “my_ra” in Replicating state:

```
ra -i my_ra -replicate
```

Usage

The Replication Agent **ra** utility:

- Starts a specified Replication Agent instance
- Returns the Replication Agent software version string

To run **ra**, invoke it as a command at the operating system prompt.

See also

- *Replication Agent Start-Up* on page 33
- *ra_admin* on page 16

Start-Up Errors

A Replication Agent instance may encounter errors on start-up.

- On Microsoft Windows platforms, start-up errors appear in the operating system command window.
- On UNIX platforms, start-up errors appear in the operating system command window and recorded in the Replication Agent system log.

See also

- *Troubleshooting* on page 117

ra_admin

Administers Replication Agent instances.

Syntax

```
ra_admin [option [create options]] [inst_name]
```

Parameters

- **-b –** Returns the complete path of the Replication Agent installation directory.
- **-c *inst_name* –** Creates a new Replication Agent instance using the specified name (*inst_name*).

The *inst_name* string must be a valid server name, and unique on the host machine.

When you use the **-c** option, one of these options is required:

- **-p** and **-t**, or
  -p , **-t**, **-uid**, and **-pwd**.
- **-p** and **-f**.

When you use the **-f** option to copy an existing Replication Agent configuration, you need not specify the **-t** option. The primary database type specified for the existing Replication Agent instance is copied to the configuration of the new Replication Agent instance when you specify the **-f** option.

- **-h –** Returns command usage information.
- **-p *port_num* –** Specifies a client socket port number for the administration port of the Replication Agent instance.

  The *port_num* must be a valid port number and unique on the Replication Agent host machine. Replication Agent also requires that a second port, *port_num+1*, must be available for the RASD.
- **-reset *inst_name* -uid *username* -pwd *new_pwd* –** Resets the Replication Agent administrator password to a new value. where:

  - *inst_name* is the name of the Replication Agent instance.
  - *username* is the Replication Agent administrator user name.
  - *new_pwd* is a new password that complies with the password security requirements.
- **-t *database* –** Identifies the type of data server that the primary database resides in.

  The *database* string must be one of:

  - **oracle** – Oracle database server
  - **mssql** – Microsoft SQL Server
  - **ibmudb** – IBM DB2 for Linux, UNIX, and Windows

Note: The *database* value is not case-sensitive.

When you use **-c**, you can specify the configuration of the new Replication Agent instance to be based on the configuration file for an existing Replication Agent instance. To do this, use the **-f** option.

- **-f *old_inst* –** Copies the configuration of an existing Replication Agent instance for a new Replication Agent instance.

  The *old_inst* string is the name of the existing Replication Agent instance whose configuration you want to copy for the new Replication Agent instance.

  When you use the **-f** option to copy an existing Replication Agent configuration, you need not specify the **-t** option. The primary database type specified for the existing Replication Agent instance is copied to the configuration of the new Replication Agent instance when you specify the **-f** option.

Note: When you use the **-f** option, some configuration parameters are set to default values.

- **-d *inst_name* –** Deletes a specified Replication Agent instance.

  The *inst_name* string must be the name of an existing Replication Agent instance.

  When you invoke **ra_admin** with the **-d** option, the utility deletes all of the subdirectories associated with the specified instance from the Replication Agent installation directory.

  Note: On Windows platforms, if any application is accessing a file or directory associated with a Replication Agent instance when you delete the instance, the open file or directory is not deleted. An error message informs you of the file or directory not deleted.

  To finish deleting a Replication Agent instance after a file or directory access conflict on a Microsoft Windows platform, you must:

  - Verify that the file or directory is not open in any application
  - Manually delete the file or directory
- **-l (lowercase L) –** Lists all verifiable Replication Agent instances.
- **-v *inst_name* –** Verifies the complete directory structure for a specified Replication Agent instance. The *inst_name* string must be the name of an existing Replication Agent instance.
- **-vr *res_file* –** Validates the specified resource file (*res_file*), without creating a Replication Agent instance or making any change in the environment.
- **-r *res_file* –** Creates a Replication Agent instance, based on the contents of the specified resource file (*res_file*).
- **-u *upgrade_option* –** Upgrades a Replication Agent instance based on the specified upgrade option (*upgrade_option*):

  - *source_installation_dir* – upgrade all instances of Replication Agent in the specified source installation directory from the current product installation directory.
  - *source_instance_dir* – upgrade one instance of Replication Agent in the specified source instance directory from the current product installation directory. The instance is first copied to and then updated within the current product installation directory.
  - **all** – upgrade all instances of Replication Agent within the current product installation directory. The configuration files are backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back.

    The **all** option requires relatively less space because upgrades are performed directly on instances within the current product installation directory, not to copies. However, reversing an upgrade is more difficult for the same reason.
  - *instance_name* **|instance=** *instance_name* – upgrade only the specified instance in the current product installation directory. The upgrade is performed directly on the specified instance, not to a copy. The configuration file is backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back.
- **-uid *ra_username* –** Creates a new administrator login name you want to use for this Replication Agent instance.

- **-pwd *ra_password* –** Sets the password for the new administrator login.
- **-i *instance_name* -k –** Clears the **ra_random** configuration property value. The Replication Agent generates a new **ra_random** value and a new **instance-encryption-key** entry after a restart.

Usage

Note: Set the SYBASE environment before you invoke the Replication Agent **ra_admin** utility.

The Replication Agent **ra_admin** utility:

- Creates, copies, deletes, and verifies Replication Agent instances.
- Lists all valid Replication Agent instances on the Replication Agent host machine.
- Returns the path of the Replication Agent installation directory.
- Creates an output file in the `$SYBASE/RAX-15_5/admin_logs` directory. The format of the file name is `adminmmddyyyy_hhmmss.log`, where *mmddyyy* and *hhmmss* are the current date and time.
- Upgrades Replication Agent instances.

To reset the Replication Agent administrator login password with the **reset** keyword, you must stop the Replication Agent instance, and it may be necessary to kill the process.

To run **ra_admin**, invoke it as a command at the operating system prompt.

Note: You can also invoke **ra_admin** with no option specified to return command usage information.

See also

- *Replication Agent Instance Creation* on page 11
- *Creating a Resource File* on page 21
- *Editing a Resource File* on page 22
- *Validating a Resource File* on page 22
- *Creating an Instance with a Resource File* on page 23
- *Preparing to Use the Utilities* on page 13

Creating a Replication Agent Instance

Create a Replication Agent instance after the Replication Agent software is installed.

At any time after the Replication Agent software is installed, invoke **ra_admin** with the **-c** option:

```
ra_admin -c new_inst -p port_num {-t database|-f old_inst} [-uid
ra_username] [-pwd ra_password]
```

where:

- *new_inst* is the name of the new Replication Agent instance you are creating.

- *port_num* is the client socket port number for the administration port of the new Replication Agent instance. Make sure that a second port, *port_num+1*, is available for the RASD.
- *database* is the type of data server that contains the primary database.
- *port_num* is the client socket port number for the administration port of the new Replication Agent instance. Make sure that a second port, *port_num+1*, is available for the RASD.
- *old_inst* is the name of an existing Replication Agent instance whose configuration you want to duplicate for the new Replication Agent instance.
- *ra_username* is the Replication Agent administrator user name. If you do not provide this option on the command line, you will be prompted for a value.
- *ra_password* is the Replication Agent administrator password. If you do not provide this option on the command line, you will be prompted for a value.

Creating a Replication Agent Instance Using Resource Files

Create a Replication Agent instance with a resource file.
Invoke **ra_admin** with the command line parameters that support creating and validating a Replication Agent instance using a resource file:

```
ra_admin {-vr res_file | -r res_file}
```

where:

- **-vr** *res_file*

  validates the specified resource file (*res_file*), without creating a Replication Agent instance or making any change in the environment.
- **-r** *res_file*

  creates a Replication Agent instance, based on the contents of the specified resource file (*res_file*).

A resource file is an ASCII text file that contains configuration information for the Replication Agent instance to be created by **ra_admin**.

The **ra_admin** parameters in the resource file allow you to specify these options, in addition to creating a Replication Agent instance:

- Create the instance user login in the primary data server, and grant all required permissions.
- Start the new instance after it is created.
- Initialize the new instance after it starts.

Note: When you validate a resource file with **ra_admin -vr**, no other action is taken, and no Replication Agent instance is created.

When you create a Replication Agent instance with a resource file:

- The **asa_password** configuration parameter value must not contain single quotes, double quotes, or a semicolon.

- The **pds_username** and **pds_password** configuration parameter values must not contain single or double quotes if the **create_pds_username** parameter is set to yes.

Creating a Resource File

Create a resource file from one of the provided templates.

Prerequisites

Locate your resource file templates, `mssql.rs` (for Microsoft SQL Server), `oracle.rs` (for Oracle), and `ibmudb.rs` (for IBM DB2 for Linux, UNIX, and Windows) in the `init` subdirectory of the Replication Agent installation directory. For example:

```
C:\sybase\RAX-15_5\init\oracle.rs
```

or

```
C:\sybase\RAX-15_5\init\mssql.rs
```

or

```
C:\sybase\RAX-15_5\init\ibmudb.rs
```

The resource file template contains comments that describe each configuration parameter and its value.

Note: Sybase recommends that you validate each resource file before you create a Replication Agent instance using that resource file.

Task

1. Copy the resource file template to another file that you edit to create the new resource file. For example:

   ```
   cp oracle.rs pubs2.rs
   ```

   where `pubs2.rs` is the name of the new resource file you want to create.

   If you have an existing resource file, you can copy that file to create a new resource file, instead of copying the template.
2. Use your preferred text editor to edit the resource file copy that you created.

After you create a new resource file, you should validate it.

See also

- *Validating a Resource File* on page 22

Editing a Resource File

The **ra_admin** resource file is an ASCII text file that you can edit using any standard text editor.

- Make sure that configuration parameters for both the Replication Agent and **ra_admin** use this format:

  ```
  param=value
  ```

  where:

  - *param* is the name of the configuration parameter.
  - *value* is the value of the configuration parameter.

  Note: Spaces are not allowed before or after the = symbol, or within the *value* string.
- Make sure that each `param=value` statement occurs on a separate line.
- If a default value exists for a configuration parameter, use:

  ```
  param=USE_DEFAULT
  ```

  where *param* is the name of the configuration parameter.
- Make sure that these **ra_admin** configuration parameters have a value of yes or no:

  - **create_pds_username**
  - **start_instance**
  - **initialize_instance**

  The yes value is case-insensitive. Any string other than `{y|Y}{e|E}{s|S}` is interpreted as no.

  Note: Blank lines and lines that begin with the # symbol are ignored in the resource file.

Validating a Resource File

Use **ra_admin** to validate your resource file.

When you invoke **ra_admin** with the **-vr** option, the utility validates the specified resource file and returns information about the validation process.

ra_admin verifies:

- Uniqueness of the Replication Agent administration port number and instance name
- Access to the primary data server, Replication Server, and RSSD
- The host name, port number, database name, user login, and password on each server
- The Replication Server database connection for the primary database
- That the **pds_username** user has all the required permissions at the primary database

If any validation fails, **ra_admin** returns an error message and information about the failure.

You can repeat the validation process as many times as necessary. No entities are changed or created as a result of this process.

Note: Sybase recommends that you validate a new resource file before you create a Replication Agent instance using the new resource file.

1. Invoke **ra_admin**, specifying the **-vr** option and the name of the resource file:

   ```
   ra_admin -vr res_file
   ```

   where *res_file* is the name of the resource file you want to validate.

   For example, if the resource file is named `pubs2.rs`, enter this at the command prompt:

   ```
   ra_admin -vr pubs2.rs
   ```

   Validation results are returned as either:
   - `Response-file processing completed`.

     or
   - `Response-file processing completed with errors.`

   If the validation is successful, skip step 2, and use the resource file to create a Replication Agent instance.

   If the validation encounters errors, continue to step 2.
2. Use this procedure to correct validation errors:
   a) Review the error messages to determine the cause of the failure.
   b) Edit the resource file to correct the appropriate values.
   c) Invoke **ra_admin -vr** again, specifying the name of the resource file.

   Repeat this step until the resource file is successfully validated.

See also

- *Creating an Instance with a Resource File* on page 23

Creating an Instance with a Resource File

Create a Replication Agent instance and specify a resource file at the same time.

When you invoke **ra_admin** with the **-r** option, the utility first validates the specified resource file except:

- If the Replication Agent for Microsoft SQL Server or Replication Agent for Oracle primary database user login does not exist in the primary data server, the utility creates it, if specified in the resource file (`create_pds_username=yes`). If the user login does exist in the primary data server but does not have all the required privileges, set **create** to yes, to have the utility grant all required permissions.

  If the Replication Agent for Microsoft SQL Server or Replication Agent for Oracle primary database user login does exist in the primary data server, has all the required privileges, and the resource file specifies that it should be created, the utility returns an error message and does not create the instance.

- If the resource file specifies that the new Replication Agent instance should be initialized (`initialize_instance=yes`), then:
  - The Replication Agent primary database user login must either exist in the primary data server, or be created by **ra_admin** (`create_pds_username=yes`).
  - The resource file must specify that the Replication Agent instance should be started (`start_instance=yes`).

  Otherwise, the utility returns an error message and does not create the instance.

After validating the resource file successfully, **ra_admin**:

- Creates and configures a Replication Agent instance, based on the contents of the specified resource file.
- Creates or grants all required privileges for the instance user, if specified in the resource file.
- Starts the new Replication Agent instance, if specified in the resource file.
- Initializes the new Replication Agent instance, if specified in the resource file.

The utility also returns information about the instance created and the result.

If instance creation fails, **ra_admin** returns an error message and information about the failure.

Note: Sybase recommends that you validate a new resource file before you create a Replication Agent instance using the new resource file.

Invoke **ra_admin**, specifying the **-r** option and the name of the resource file:

```
ra_admin -r res_file
```

where *res_file* is the name of the resource file.

For example, if the resource file is named `pubs2.rs`, enter this at the command prompt:

```
ra_admin -r pubs2.rs
```

Results are returned as either:

- `Response-file processing completed`.

  or
- `Response-file processing completed with errors.`

If the instance creation is successful, you can begin using the new Replication Agent instance.

If the instance creation fails, you may have to:

- Delete all files and subdirectories in the instance directory, and delete the instance directory from the Replication Agent installation directory.
- Edit the resource file to correct the appropriate values.

Note: If the instance creation fails, use the error-recovery procedure before you attempt to create the instance again.

See also

- *Validating a Resource File* on page 22

Recovering from Instance Creation Errors

You may need to recover from errors resulting from an instance creation.

1. If the resource file does not specify that the instance user login be created in the primary data server, skip this step and continue with step 2.

   If the resource file specifies that the instance user login be created in the primary data server (that is, `create_pds_username=yes`), then:

   a) Check the primary database to determine if the instance user was added.
   b) Check that the **pds_sa_username** has sufficient privileges to create the instance login at the primary database.
   c) Edit the resource file to specify that the instance user login should not be created in the primary data server (`create_pds_username=no`).

   Note: If the Replication Agent primary database user login is successfully created before the instance creation fails, you must either:

   - Edit the resource file to set the value of the **create_pds_username** parameter to no, or,
   - Log in to the primary data server and drop the instance login.

2. Check the Replication Agent base directory on the Replication Agent host to determine if a new instance directory was created. The Replication Agent base directory is:

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

   where *%SYBASE%* is the Replication Agent installation directory.

   If you do not find a new instance directory in the Replication Agent base directory, skip to step 4.

   If you find a new instance directory in the Replication Agent base directory, continue with step 3.

3. To delete the new instance directory, you have two options:

   - Use **ra_admin** to delete the instance:

     ```
     ra_admin -d inst_name
     ```

     where *inst_name* is the name of the instance you want to delete, or
   - Use operating system commands to delete all of the files and subdirectories in the new instance directory, and then delete the new instance directory.

4. Review the error messages to find the cause of the instance creation failure, and if necessary, edit the resource file to correct the appropriate values.

   After editing the resource file, validate it:

   ```
   ra_admin -vr res_file
   ```

where *res_file* is the name of the resource file.

After you complete the recovery procedure, retry creating the Replication Agent instance.

See also

- *Validating a Resource File* on page 22

Copying a Replication Agent Configuration

You can copy a Replication Agent configuration with `ra_admin`.

When you create a new Replication Agent instance, copy the configuration of an existing instance by invoking **ra_admin** with the **-c** option and **-f** option:

```
ra_admin -c new_inst -p port_num -f old_inst
```

where:

- *new_inst* is the name of the new Replication Agent instance you are creating.
- *port_num* is the client socket port number for the administration port of the new Replication Agent instance. Make sure that a second port, *port_num+1,* is available for the RASD.
- *old_inst* is the name of an existing Replication Agent instance whose configuration you want to duplicate for the new Replication Agent instance.

See also

- *Replication Agent Instance Creation* on page 11

Copying an Existing Replication Agent Instance Configuration to a New Instance

Create a new Replication Agent instance based on the configuration of an existing instance.

1. On the Replication Agent host machine, open an operating system command window.
2. Navigate to the Replication Agent `bin` directory:
   - On Windows platforms:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```
     where *%SYBASE%* is the path to the Replication Agent installation directory.
   - On UNIX platforms:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
     where *$SYBASE* is the path to the Replication Agent installation directory.
3. In the Replication Agent `bin` directory, invoke **ra_admin** to create a new Replication Agent instance whose configuration is based on the configuration of an existing instance:
   ```
   ra_admin -c new_inst -p port_num -f old_inst
   ```
   where:
   - *new_inst* is the name of the new Replication Agent instance.

- *port_num* is the client socket port number for the administration port of the new instance.
- *old_inst* is an existing Replication Agent instance whose configuration you want to copy for the new instance.

After you invoke **ra_admin**, the operating system prompt returns when the new Replication Agent instance is created.

4. Verify that the Replication Agent instance was created properly using one of these methods:

   - Invoke **ra_admin** with the **-v** option, and specify the name of the Replication Agent instance:

     ```
     ra_admin -v new_inst
     ```

     where *new_inst* is the name of the Replication Agent instance.

     When you verify a Replication Agent instance with the **-v** option, the utility verifies the instance by checking for an instance directory with the specified instance name under the Replication Agent base directory, and checking all of the subdirectories under the Replication Agent instance directory.
   - Invoke **ra_admin** with the **-l** option:

     ```
     ra_admin -l
     ```

     The **-l** option lists all verifiable Replication Agent instances, which should include the new one you just created.
   - As an alternative to using **ra_admin**, you can use operating system commands to verify that the Replication Agent instance directories were created correctly.

Note: When you create a new Replication Agent instance and copy the configuration of an existing instance, some configuration parameters are set to default values, and they are not copied from the existing configuration.

The values of these configuration parameters are not copied from an existing configuration:

- **admin_port**
- **log_directory**
- **pds_database_name**
- **pds_datasource_name**
- **pds_host_name**
- **pds_password**
- **pds_port_number**
- **pds_retry_count**
- **pds_retry_timeout**
- **pds_server_name**
- **pds_username**
- **rs_source_db**

- **rs_source_ds**
- **function_password**
- **rs_password**
- **rssd_password**
- **ssl_identity_password**
- **ra_admin_owner_password**
- **ddl_password**
- **rasd_backup_dir**
- **rasd_database**
- **rasd_trace_log_dir**
- **rasd_tran_log**
- **asa_port**
- **asm_password**
- **rman_password**

See the *Replication Agent Reference Manual* for more information about Replication Agent configuration parameters.

After you create a Replication Agent instance, you can use **ra** to start the instance so that you administer and configure it.

Note: A new user login name and password are required when creating a Replication Agent instance. You have the option to change the password.

See also

- *Preparing to Use the Utilities* on page 13
- *Creating or Changing the Replication Agent Administrator Login* on page 41

Resetting the Replication Agent Administrator Password

Reset the Replication Agent administrator password to a new value.

1. Stop the Replication Agent instance, and it may be necessary to kill the process.
2. Log in to the Replication Agent administration port.
3. Reset the password for administrator:

   ```
   ra_admin –reset inst_name [,-uid username, -pwd new_pwd]
   ```

   where:
   - *inst_name* is the name of the Replication Agent instance.
   - *username* is the Replication Agent administrator user name.
   - *new_pwd* is a new password that complies with the password security requirements.

   Note: The parameter **-uid** and **-pwd** are optional on the command-line. You will be prompted to provide them if they are not included in the command.

4. Restart the Replication Agent instance.
5. Check the Replication Agent log file to verify that the Replication Agent instance has started successfully:
   - (UNIX or Linux)
     ```
     $SYBASE/RAX-15_x/<instance_name>/log/<instance_name>.log
     ```
   - (Windows)
     ```
     %SYBASE%\RAX-15_x\<instance_name>\log\<instance_name>.log
     ```

Creating a Replication Agent Instance Using the Command Line

Create a Replication Agent instance from the Windows or UNIX command line.

Prerequisites

Set the SYBASE environment before you invoke **ra_admin**.

Task

1. On the Replication Agent host machine, open an operating system command window.
2. Navigate to the Replication Agent `bin` directory:
   - On Windows platforms:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```
     where *%SYBASE%* is the path to the Replication Agent installation directory.
   - On UNIX platforms:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
     where *$SYBASE* is the path to the Replication Agent installation directory.
3. In the Replication Agent `bin` directory, invoke **ra_admin** to create a new Replication Agent instance:
   ```
   ra_admin -c new_inst -p port_num -t database [-uid ra_username]
   [-pwd ra_password]
   ```
   where:
   - *new_inst* is the name of the new Replication Agent instance.
   - *port_num* is the client socket port number for the administration port of the new instance. Make sure that a second port, *port_num+1*, is available for the RASD.
   - *database* identifies the type of data server that the primary database resides in:
     - **oracle** – Oracle database server
     - **mssql** – Microsoft SQL Server (valid only on Microsoft Windows platforms)
     - **ibmudb** – IBM DB2 for Linux, UNIX, and Windows
   - *ra_username*
     is the administrator login name you want to use for this Replication Agent instance. If you do not provide this option on the command line, you will be prompted for a value.

- *ra_password* is the password for the administrator login. If you do not provide this option on the command line, you will be prompted for a value.

   After you invoke **ra_admin**, the operating system prompt returns when the new Replication Agent instance is created.

4. Verify that the Replication Agent instance was created properly using one of these methods:

   - Invoke **ra_admin** with the **-v** option, and specify the name of the Replication Agent instance:

     ```
     ra_admin -v new_inst
     ```

     where *new_inst* is the name of the Replication Agent instance.

     When you verify a Replication Agent instance with the **-v** option, the utility verifies the instance by checking for an instance directory with the specified instance name under the Replication Agent base directory, and checking all of the subdirectories under the Replication Agent instance directory.
   - Invoke **ra_admin** with the **-l** option:

     ```
     ra_admin -l
     ```

     The **-l** option lists all verifiable Replication Agent instances, which should include the new one you just created.
   - As an alternative to using **ra_admin**, you can use operating system commands to verify that the Replication Agent instance directories were created correctly.

After you create a Replication Agent instance, you can use **ra** to start the instance so that you can administer and configure it.

See also

- *Replication Agent Start-Up* on page 33
- *Preparing to Use the Utilities* on page 13
- *Creating or Changing the Replication Agent Administrator Login* on page 41

Deleting a Replication Agent Instance

Delete a Replication Agent instance with **ra_admin**.

Prerequisites

- Shut down the Replication Agent instance, if it is running.
- If the Replication Agent software is installed on a Microsoft Windows platform, verify that none of the files in the instance subdirectories are open, and that no application or window is accessing the instance subdirectories.

Note: Set the SYBASE environment before you invoke the **ra_admin** utility.

Note: If you delete a Replication Agent instance, Replication Agent does not unmark any primary database objects marked for replication, nor does it delete its transaction log objects.

Before you shut down and delete a Replication Agent instance, unmark primary database objects and deinitialize the Replication Agent so that it removes the objects it created in the primary database.

Task

1. On the Replication Agent host machine, open an operating system command window.
2. Navigate to the Replication Agent `bin` directory:
   - On Windows platforms:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```
     where *%SYBASE%* is the path to the Replication Agent installation directory.
   - On UNIX platforms:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
     where *$SYBASE* is the path to the Replication Agent installation directory.
3. In the Replication Agent `bin` directory, invoke **ra_admin** with the **-d** option to delete a Replication Agent instance:
   ```
   ra_admin -d inst_name
   ```
   where *inst_name* is the name of the Replication Agent instance you want to delete.

   After you invoke **ra_admin** with the **-d** option, you see:
   ```
   Are you sure you want to delete the Replication Agent instance
   inst_name? [y/n]
   ```
4. Enter **y** to delete the Replication Agent instance.

   After the instance is deleted, the operating system prompt returns.

   If the instance is running when you invoke **ra_admin** with the **-d** option, the utility returns:
   ```
   Cannot delete Replication Agent instance 'inst_name' because it is
   currently running.
   ```
   To shut down a Replication Agent instance, log in to its administrative port, and use the **shutdown** command.
5. Verify that the Replication Agent instance was deleted properly using one of these methods:
   - Invoke **ra_admin** with the **-v** option, and specify the name of the Replication Agent instance:
     ```
     ra_admin -v new_inst
     ```
     where *new_inst* is the name of the Replication Agent instance.
     When you verify a Replication Agent instance with the **-v** option, the utility verifies the instance by checking for an instance directory with the specified instance name under

the Replication Agent base directory, and checking all of the subdirectories under the Replication Agent instance directory.

- Invoke **ra_admin** with the **-l** option:

  ```
  ra_admin -l
  ```

  The **-l** option lists all verifiable Replication Agent instances, which should include the new one you just deleted.
- As an alternative to using **ra_admin**, you can use operating system commands to verify that the Replication Agent instance directories were deleted correctly.

Note: On Microsoft Windows platforms, if any application is accessing a file or directory associated with a Replication Agent instance when you delete the instance, the open file or directory is not deleted. An error message informs you of the file or directory not deleted.

To finish deleting a Replication Agent instance after a file or directory access conflict occurs on a Microsoft Windows platform, you must:

- Verify that the file or directory is not open in any application
- Manually delete the file or directory

6. Verify that the Replication Agent instance was deleted properly using one of these methods:

See also

- *Replication Agent Instance Shutdown* on page 69
- *Preparing to Use the Utilities* on page 13

Replication Agent Instance Upgrades

You can upgrade a Replication Agent instance from the command line interface

Upgrades for Instances at the Same Release Level

Use **ra_admin -u** *source_installation_dir* to upgrade instances of Replication Agent earlier than version 15.5 in the specified directory from the current installation directory.

For instances of Replication Agent of version 15.5 or later, use **ra_admin -u all** or **ra_admin -u** *instance_name* **|instance=** *instance_name* to upgrade an instance or instances.

See also

- *ra* on page 15

Replication Agent Instance Downgrades

Use the downgrade API to downgrade instances of Replication Agent.

Use the **ra_downgrade**, **ra_migrate**, and **resume purge** commands to downgrade an instance of Replication Agent. See the sections on upgrading and downgrading Replication Agent in the *Replication Agent Primary Database Guide* for instructions on using these commands.

Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands are deprecated as of Replication Agent 15.7.1 but may be necessary to downgrade to earlier versions.

Replication Agent Start-Up

To start a Replication Agent instance, you must log in to the Replication Agent host machine with a user name that has **execute** permission in the Replication Agent installation directory and all subdirectories (for example, the “sybase” user).

Note: On Windows Vista, you must run the command window as an Administrator. To do so, click **Start > All Programs > Accessories**, right-click **Command Prompt**, then select **Run As Administrator**.

You can start a Replication Agent instance in any of these ways:

- Invoke the `RUN` script for the instance that you want to start.
- Invoke **ra**, and specify the instance that you want to start.
- If Replication Agent is installed on a Windows system, start a Windows service for the instance that you want to start.

The `RUN` script and the **ra** utilities are batch files on Microsoft Windows and shell scripts on UNIX.

Preparing for Start-Up

Before you can start a Replication Agent instance and connect to the primary data server, you must set all required variables.

1. Add the location of the JDBC driver for the primary database to the CLASSPATH environment variable.

   See the *Replication Agent Primary Database Guide* for more information about installing and setting up the JDBC driver for the primary database and setting up Replication Agent connectivity.
2. If the default character set on the host machine on which your Replication Agent is running is different from the one on your primary database, you need to set the RA_JAVA_DFLT_CHARSET environment variable so it is the same as that of the primary database.

Next

See the *Replication Agent Primary Database Guide* for more information about connectivity requirements specific to your primary database.

Character Settings

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

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

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

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

Using character set conversions slows performance.

Note: Sybase recommends that you use the same character set on all servers and platforms in a Replication Agent system.

Environment Character Set Configuration

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

To support overriding the default character set, all of the executable scripts (or batch files) in the Replication Agent `/bin` directory refer to an environment variable named RA_JAVA_DFLT_CHARSET. You can set this environment variable to use the character set you want. The character set you specify must be the character set configured on the primary database. For a list of valid Java 7 character sets, see the supported encodings on the internationalization page at *http://docs.oracle.com/javase/7/docs/technotes/guides/intl/encoding.doc.html*

All Replication Agent instance `RUN` scripts also reference the RA_JAVA_DFLT_CHARSET environment variable.

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

You can override the system default character set by either:

- Set the value of a system variable named RA_JAVA_DFLT_CHARSET in your environment and use **ra** to start the Replication Agent instance, or,
- Set the value of the RA_JAVA_DFLT_CHARSET variable in the Replication Agent instance `RUN` script and use the `RUN` script to start the Replication Agent instance.

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

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

Overriding the System Default Character Set for All Replication Agent Instances

Override the default character set for all instances of Replication Agent.

1. Enter a character set value in the `ra` script:
   - For Windows, edit the `%SYBASE%\RAX-15_5\bin\ra.bat` file.
   - For UNIX, edit the `$SYBASE/RAX-15_5/bin/ra.sh` file:

     ```
     RA_JAVA_DFLT_CHARSET=charset
     ```

     where *charset* is the Java-supported encoding.

     For example, `ISO8859_1` or `Cp1252` for ISO-1 (also known as Latin-1), and `ISO8859_8` or `Cp1255` for Hebrew.

     Note: In UNIX, spaces are not allowed on either side of the equals sign.
2. Uncomment these lines of code:
   - For Windows:

     ```
     set RA_JAVA_DFLT_CHARSET=charset
     ```
   - For UNIX:

     ```
     RA_JAVA_DFLT_CHARSET=charset
     export RA_JAVA_DFLT_CHARSET
     ```

Overriding the System Default Character Set for a Specified Replication Agent Instance

Override the default character set for only one specified instance of Replication Agent. Enter a character set value in the `RUN` script:

- For Windows, edit the `%SYBASE%\RAX-15_5\<instance>\RUN_<instance>.bat` script.
- For UNIX, edit the `$SYBASE/RAX-15_5/<instance>/RUN_<instance>.sh` batch file.

Here, *charset* is the Java-supported encoding.

For example, `ISO8859_1` or `Cp1252` for ISO-1 (also known as Latin-1), and `ISO8859_8` or `Cp1255` for Hebrew is supported.

Note: In UNIX, spaces are not allowed on either side of the equals sign.

Starting Replication Agent Using the ra Utility

When you start the Replication Agent with **ra**, you can specify the instance start-up state. If you do not specify a start-up state when you invoke **ra**, the Replication Agent instance starts in its default Admin state.

Prerequisites

Set the SYBASE environment before you invoke **ra**.

Task

1. On the Replication Agent host machine, open an operating system command window.
2. Navigate to the Replication Agent `bin` directory:
   - On Windows platforms:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```
     where *%SYBASE%* is the path to the Replication Agent installation directory.
   - On UNIX platforms:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
     where *$SYBASE* is the path to the Replication Agent installation directory.
3. In the Replication Agent `bin` directory, invoke **ra** to start the Replication Agent instance:
   ```
   ra -iinst_name
   ```
   or
   ```
   ra -iinst_name -state
   ```
   where:
   - *inst_name* is the server name of the Replication Agent instance.
   - *state* is the optional keyword for the start-up state:
     - **admin** – starts the Replication Agent instance in Admin state.
     - **replicate** – starts the Replication Agent instance in Replicating state.

   Note: If you do not specify the state option, Replication Agent starts in Admin state.

   For example, to start the Replication Agent instance named “my_ra” in Replicating state, enter:
   ```
   ra -i my_ra -replicate
   ```

After you start the Replication Agent instance, you must open another operating system command window to log in to its administration port.

See also

- *ra* on page 15

- *Preparing to Use the Utilities* on page 13

Starting a Replication Agent Instance with the RUN Script

You can use the `RUN` script to start a Replication Agent instance.

Prerequisites

Note: You do not need to set the SYBASE environment variable before you invoke the `RUN` script, because the `RUN` script sets the SYBASE environment variable before it starts the Replication Agent instance.

Task

1. On the Replication Agent host machine, open an operating system command window.
2. At the operating system prompt, navigate to the Replication Agent instance directory, and enter:
   - On Windows:
     ```
     cd %SYBASE%\RAX-15_5\inst_name
     ```
     where:
     - *%SYBASE%* is the path to the Replication Agent installation directory.
     - *inst_name* is the name of the Replication Agent instance.
   - On UNIX:
     ```
     cd $SYBASE/RAX-15_5/inst_name
     ```
     where:
     - *$SYBASE* is the path to the Replication Agent installation directory.
     - *inst_name* is the name of the Replication Agent instance.
3. In the Replication Agent instance directory, invoke the `RUN` script to start the Replication Agent instance:
   ```
   RUN_inst_name
   ```
   where *inst_name* is the server name of the Replication Agent instance.

   For example, to start the Replication Agent instance named "my_ra," enter:
   ```
   RUN_my_ra
   ```
   The `RUN` script is created automatically when the Replication Agent instance is created. The `RUN` script invokes **ra** with the appropriate parameter values to start the Replication Agent instance. You can edit the `RUN` script to specify the start-up state.

   Note: The UNIX version of the `RUN` script does not have the `.sh` extension.

After you start the Replication Agent instance, you must open another operating system command window to log in to its administration port.

Starting Replication Agent Using agt_service

You can use **agt_service** to start a Replication Agent instance.

Prerequisites

Set the SYBASE environment before you invoke **agt_service**.

Task

If Replication Agent is installed on a Windows system, you can use **agt_service** to run Replication Agent as a Windows service. The Replication Agent instance starts in the start-up state that you specified when you created the Windows service. If you did not specify a start-up state, the Replication Agent instance starts in its default Admin state.

1. On the Replication Agent host machine, open an operating system command window.
2. Navigate to the Replication Agent `bin` directory:

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

   where *%SYBASE%* is the path to the Replication Agent installation directory.
3. In the Replication Agent `bin` directory, invoke **agt_service** to start a Windows service for the Replication Agent instance:

   ```
   agt_service -start inst_name
   ```

   where *inst_name* is the name of the Replication Agent instance assigned to the Windows service when it was created. This is also the name of the Windows service. For example, to start the Replication Agent instance named “my_ra” with the Windows service name of “my_ra,” enter:

   ```
   agt_service -start my_ra
   ```

See also

- *agt_service* on page 14
- *Preparing to Use the Utilities* on page 13

Starting Replication Agent from the Windows Services Tool

You can use the Windows Services administration tool to start a Replication Agent instance both automatically and manually.

1. Log in to Windows using an account with Windows administrator privileges.
2. Select **Start** > **Settings** > **Control Panel** > **Administrative Tools** > **Services**.
3. Scroll through the list of available services until you find the listings for your Sybase servers.

   Server names use this:

"Sybase Replication Agent *inst_name*"

where *inst_name* is the name of the Replication Agent instance.

4. Right-click the service name, and choose **Start**.
5. Click **Close**.

You can verify the status of the instance either from Sybase Central™ or by looking at the Status column of the Windows Services tool.

The Replication Agent Administration Port

When you create a Replication Agent instance, you specify a client socket port number for its administration port. Client applications use this port to connect to the Replication Agent.

The administration port allows Open Client (or Open Client-compatible) applications to log in and execute Replication Agent commands. You can use any Sybase Open Client interface utility (such as **isql** or SQL Advantage®) to connect to the Replication Agent administration port.

Note: Client applications are not provided with the Replication Agent software. **isql** is provided with the Replication Server software, and **isql** and SQL Advantage are provided with the Adaptive Server Enterprise software.

Interfaces File Entries

In general, Open Client applications (such as **isql**) require an interfaces file to identify available servers, host machines, and client ports. On Windows, the interfaces file is named `sql.ini`; on UNIX, the interfaces file is named `interfaces`.

To connect Open Client applications to the Replication Agent administration port as they would connect to any other Open Server application, create a server entry for the Replication Agent in the interfaces file on the Open Client application host machine.

A server entry for a Replication Agent administration port in an interfaces file appears:

```
[inst_name]
 query=protocol,host_name,port_num
```

where:

- *inst_name* is the name of the Replication Agent instance.
- *protocol* is the network protocol used for the connection.
- *host_name* is the name of the Replication Agent host machine.
- *port_num* is the client socket port number for the Replication Agent instance.

For example, to specify an interfaces file entry for a Replication Agent instance named "my_ra," using the Windows socket protocol, on a host named "my_host," with client socket port number 10002, add these lines to the interfaces file:

```
[my_ra]
query=NLWNSCK,my_host,10002
```

Some systems require the interfaces file to be in the TLI form. If your system does, you must use a utility (such as **sybtli** or **dsedit**) that edits the interfaces file and saves the result in a form compatible with TLI.

After you create an entry for the Replication Agent instance in the interfaces file, you can connect to the administration port using any Open Client application that uses that interfaces file.

Logging in to a Replication Agent Instance Using isql

You can use the **isql** interactive SQL utility to log in to the Replication Agent administration port.

Prerequisites

Before you can log in to the Replication Agent administration port with an Open Client application (such as **isql**), create a server entry for the Replication Agent instance in the interfaces file.

Task

1. Open an operating system command window.
2. Set your environment variables by sourcing the `SYBASE.csh` file (UNIX or Linux) or by executing the `SYBASE.bat` file (Windows).
3. At the operating system prompt, enter:

   ```
   isql -Uusername -Ppassword -Sinst_name
   ```

   where:
   - *username* is the Replication Agent administrator login.
   - *password* is the corresponding password.
   - *inst_name* is the name of the Replication Agent instance.

   For example, to log in to a new Replication Agent instance named “my_ra,” enter:

   ```
   isql -Usa -Ppassword -Smy_ra
   ```

Once you have successfully logged in to the administration port, you can use Replication Agent commands to administer the Replication Agent instance.

See also

- *Interfaces File Entries* on page 39

Creating or Changing the Replication Agent Administrator Login

Each Replication Agent instance has only one administrator login. The default administrator login is created when the Replication Agent instance is created.

You can use **ra_set_login** to create (or change) the administrator login for a Replication Agent instance.

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

   When you log in to the Replication Agent instance for the first time, use the default administrator login.
2. After you log in, enter:

   ```
   ra_set_login admin_user,admin_pw
   ```

   where:
   - *admin_user* is the new administrator login name you want to use for this Replication Agent instance.
   - *admin_pw* is the password for the new administrator login.

The new login name replaces the current administrator login. The next time you log in to the Replication Agent instance, you must use the new administrator login name and password.

Replication Agent Connectivity Setup

You must set up connectivity between the Replication Agent instance and the primary data server, Replication Server, and the RSSD.

Primary databases require you to perform specific setup tasks before you can set up connectivity between the Replication Agent and a primary database. See the *Replication Agent Primary Database Guide* to verify that the required setup tasks have been performed for your primary database.

Note: The term "RSSD" in this document refers to both RSSD and ERSSD; any difference is noted.

Setting up connectivity for the Replication Agent requires:

- Creating a user login name, with the appropriate authority in the primary data server and the primary database, for the Replication Agent
- Creating a user login name, with **connect source** and **create object** permission in the Replication Server, for the Replication Agent
- Creating a user login name, with the appropriate authority in the RSSD data server and the RSSD, for the Replication Agent
- Setting values for the Replication Agent connection configuration parameters

To record the values of connection configuration parameters for each Replication Agent instance, use the "Installation and Setup Worksheet" in the *Replication Agent Installation Guide*.

Creating a Primary Database User Login for Replication Agent

Replication Agent requires client access to the primary database to get information about database schema, read and manage Replication Agent objects in the primary database, and get information about database log devices (Replication Agent for Oracle and Replication Agent for Microsoft SQL Server).

To set up a user login name in the primary data server and the primary database for the Replication Agent instance:

Note: You must have a system administrator user role in the primary data server to perform this procedure.

1. Log in to the primary data server with a system administrator user role.
2. Add the Replication Agent login name to the primary data server, and if necessary, to the primary database.
   - Refer to the *Replication Agent Primary Database Guide* for information about the permissions and authorities required in each type of primary data server and primary database.
   - Refer to the documentation provided by your primary data server vendor for information about the specific commands you need to execute to create the Replication Agent login name in the primary data server (and, if necessary, in the primary database).

After you set up the Replication Agent user login in the primary data server, verify that the new user login name is valid (it can log in to the primary data server and access the primary database).

Creating a Replication Server User Login for Replication Agent

Replication Agent requires client access to the primary Replication Server to send replicated transactions.

Prerequisites

You must have **sa** permission in the Replication Server to perform this procedure.

Task

To set up a Replication Server user login name for the Replication Agent instance:

1. Log in to the Replication Server with a login that has "**sa**" permission.
2. In the Replication Server, create the Replication Agent user login name:

```
create user ra_rs_user set password ra_rs_pwd
```

where:

- *ra_rs_user* is the Replication Agent user login name.
- *ra_rs_pwd* is the password for the user login name.

3. Grant **connect source** permission to the Replication Agent login name:

```
grant connect source to ra_rs_user
```

where *ra_rs_user* is the Replication Agent user login name.

After you set up the Replication Agent user login in the primary Replication Server, verify that the new user login name is valid (it can log in to the Replication Server).

The RSSD User Login Name

Replication Agent requires client access to the ERSSD or RSSD to obtain information about replication definitions.

Setting Up the ERSSD User Login for Replication Agent

Set up a user login name for the Replication Agent instance in an ERSSD managed by SQL Anywhere®.

Prerequisites

You must have the primary user role in the ERSSD ("sa" permission in the Replication Server) to perform this procedure.

Task

By default, the Replication Server uses an embedded RSSD.

1. Log in to the ERSSD as the primary user.
2. Add the Replication Agent login name to the ERSSD:

```
grant connect to ra_rssd_user
identified by ra_rssd_pwd
```

where:

- *ra_rssd_user* is the Replication Agent user login name.
- *ra_rssd_pwd* is the password for the user login name.

3. Give the Replication Agent user permission to read the Replication Server system tables:

```
grant membership in group rs_systabgroup
to ra_rssd_user
```

where *ra_rssd_user* is the Replication Agent user login name.

After you set up the Replication Agent user login in the ERSSD, verify that the new user login name is valid (it can log in to the ERSSD and access the Replication Server system tables).

Setting Up the RSSD User Login for Replication Agent

Set up a user login name for the Replication Agent instance in an RSSD managed by Adaptive Server Enterprise.

Prerequisites

You must have a system administrator user role in the Adaptive Server Enterprise that manages the RSSD to perform this procedure.

Task

You can configure Replication Server to use an external Adaptive Server Enterprise database to host the RSSD information. By default, the Replication Server uses an embedded RSSD. If your environment requires that an Adaptive Server Enterprise must be used to host the RSSD, these instructions apply.

1. Log in to the Adaptive Server Enterprise that manages the RSSD.
2. Add the Replication Agent login name to the RSSD data server:

   ```
   use master
   sp_addlogin ra_rssd_user, ra_rssd_pwd, rssd_db
   ```

   where:
   - *ra_rssd_user* is the Replication Agent user login name.
   - *ra_rssd_pwd* is the password for the user login name.
   - *rssd_db* is the database name of the RSSD.
3. Add the Replication Agent user login name to the RSSD, and add the login name to the *rs_systabgroup* group:

   ```
   use rssd_db
   sp_adduser ra_rssd_user
   sp_changegroup rs_systabgroup, ra_rssd_user
   ```

   where:
   - *rssd_db* is the database name of the RSSD.
   - *ra_rssd_user* is the Replication Agent user login name.

After you set up the Replication Agent user login in the RSSD, verify that the new user login name is valid (it can log in to the RSSD data server and access the RSSD).

Connection Configuration Parameter Settings

When Replication Agent connects to another replication system component, it uses values stored in its configuration parameters to define a minimal set of connection properties.

These properties are:

- Server host name
- Port number
- User login name
- User login password

Note: The complete set of connection parameters is different for each database. For the complete set of connection parameters that each database requires, see the *Replication Agent Primary Database Guide*.

For its connection to the Replication Server, Replication Agent relies on the values of two additional configuration parameters (**rs_source_db** and **rs_source_ds**) to identify the Replication Server primary database connection in the LTL **connect source** command.

The Replication Agent instance must be in Admin state to set up connection parameters. In Admin state, the instance has no connections established to other replication system components, but it is available to execute administrative commands.

Note: The values of the **rs_source_db** and **rs_source_ds** parameters must exactly match the database and data server names specified in the **create connection** command for the Replication Server primary database connection. The values are case-sensitive.

See the *Replication Agent Reference Manual* for more information about the **rs_source_db** and **rs_source_ds** parameters.

To record the values of connection configuration parameters for each Replication Agent instance, use the “Installation and Setup Worksheet” in the *Replication Agent Installation Guide*.

Note: The Replication Agent instance must be running before you can set its connection configuration parameter values.

See also

- *Replication Agent States* on page 65
- *Replication Agent Start-Up* on page 33

Setting Up Connection Parameters for the Primary Database

Configure connectivity to the primary database.

1. Log in to the Replication Agent administration port, and verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

   If the instance is not in the Admin state, change it to the Admin state:

   ```
   suspend
   ```

2. Specify the primary data server host name:

   ```
   ra_config pds_hostname, pds_host
   ```

   where *pds_host* is the network name of the primary data server host machine.

If Your Primary Data Server Is:	Specify:
Oracle	The primary data server host name: `ra_config pds_hostname, pds_host` where *pds_host* is the network name of the primary data server host machine.
Microsoft SQL Server	The primary data server name: `ra_config pds_server_name, server` where *server* is the name of the primary data server.
IBM DB2 UDB	The primary data server host name: `ra_config pds_hostname, pds_host` where *pds_host* is the network name of the primary data server host machine. The data source name (DSN) specified for the connectivity driver used on the primary database connection.: `ra_config pds_datasource_name, pds_dsn` where *pds_dsn* is the DSN.

3. Specify the primary data server port number:

```
ra_config pds_port_number, NNN
```

where *NNN* is the number of the network port where the primary data server listens for connections.

4. Specify the primary database name:

```
ra_config pds_database_name, pdb
```

where *pdb* is the database name of the primary database.

5. Specify the primary data server user login name for the Replication Agent instance:

```
ra_config pds_username, ra_pds_user
```

where *ra_pds_user* is the user login name that the Replication Agent uses to log in to the primary data server.

6. Specify the password for the Replication Agent user login:

```
ra_config pds_password, ra_pds_pwd
```

where *ra_pds_pwd* is the password for the user login name that the Replication Agent uses to log in to the primary data server.

After you set up connection configuration parameters for the primary database, you can use the Replication Agent **test_connection PDS** command to test connectivity between the Replication Agent and the primary database.

See also

- *Network Connectivity Test* on page 49

Setting Up Connection Parameters for the Replication Server

Configure connectivity to Replication Server.

1. Log in to the Replication Agent administration port, and verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

   If the instance is not in the Admin state, change it to the Admin state:

   ```
   suspend
   ```
2. Specify the Replication Server host name:

   ```
   ra_config rs_hostname, rs_host
   ```

   where *rs_host* is the network name of the Replication Server host machine.
3. Specify the Replication Server port number:

   ```
   ra_config rs_port_number, NNN
   ```

   where *NNN* is the number of the network port where Replication Server listens for connections.
4. If you are using Replication Server 15.0 or earlier, specify the Replication Server character set:

   ```
   ra_config rs_charset, charset
   ```

   where *charset* matches the **RS_charset** value in the Replication Server `configuration (.cfg)` file. The location of the Replication Server configuration file is `$SYBASE/REP-15_0/install/<instance>.cfg`, where *<instance>* is the Replication Server instance.

   Note: For Replication Server 15.0.1 and later, Replication Agent uses the value of the Replication Server **RS_charset** parameter instead of the Replication Agent **rs_charset** parameter.
5. Specify the Replication Server user login name for the Replication Agent instance:

   ```
   ra_config rs_username, ra_rs_user
   ```

   where *ra_rs_user* is the user login name that the Replication Agent uses to log in to the primary Replication Server.
6. Specify the user login password for the Replication Agent instance:

   ```
   ra_config rs_password, ra_rs_pwd
   ```

where *ra_rs_pwd* is the password for the user login name that the Replication Agent uses to log in to the primary Replication Server.

7. Specify the primary data server name for the Replication Server primary database connection:

   ```
   ra_config rs_source_ds, pds
   ```

   where *pds* is the primary data server name that the Replication Agent uses in the LTL **connect source** command.
8. Specify the primary database name for the Replication Server primary database connection:

   ```
   ra_config rs_source_db, pdb
   ```

   where *pdb* is the primary database name that the Replication Agent uses in the LTL **connect source** command.

Setting Up Connection Parameters for the ERSSD (or RSSD)

Configure connectivity to the ERSSD or RSSD.

1. Log in to the Replication Agent administration port, and verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

   If the instance is not in the Admin state, change it to the Admin state:

   ```
   suspend
   ```
2. Specify the ERSSD host name:

   ```
   ra_config rssd_hostname, rssd_host
   ```

   where *rssd_host* is the network name of the ERSSD host machine.
3. Specify the ERSSD port number:

   ```
   ra_config rssd_port_number, NNN
   ```

   where *NNN* is the number of the network port where the ERSSD server listens for connections.
4. Specify the ERSSD database name:

   ```
   ra_config rssd_database_name, rssd_db
   ```

   where *rssd_db* is the database name of the ERSSD.
5. Specify the ERSSD user login name for the Replication Agent instance:

   ```
   ra_config rssd_username, ra_rssd_user
   ```

   where *ra_rssd_user* is the user login name that the Replication Agent uses to log in to the ERSSD.
6. Specify the user login password for the Replication Agent instance:

   ```
   ra_config rssd_password, ra_rssd_pwd
   ```

where *ra_rssd_pwd* is the password for the user login name that the Replication Agent uses to log in to the RSSD.

After you set up connection configuration parameters for the primary Replication Server and RSSD, you can use the Replication Agent **test_connection RS** command to test connectivity between the Replication Agent and the Replication Server and RSSD.

Network Connectivity Test

Replication Agent provides a simple means of testing network connections.

Note: You must be in Admin state to test network connectivity.

The **test_connection** command sends a connection request and confirms the network connection to these servers:

- Primary data server
- Primary Replication Server
- RSSD server (if so configured)

Note: If the value of the **use_rssd** configuration parameter is true, the **test_connection** command tests Replication Agent connectivity to the RSSD when it tests connectivity to the Replication Server. If the value of the **use_rssd** configuration parameter is false, the **test_connection** command does not test Replication Agent connectivity to the RSSD.

The **test_connection** command returns a failure message if:

- The connection specifications (server name, port number, user login, and so forth) recorded in the Replication Agent configuration parameters are not correct.
- The Replication Agent cannot establish a connection to a server because of a network failure.
- The Replication Agent cannot establish a connection to a server because the server is down.

The **test_connection** command does not validate Replication Agent user permissions in the primary database. It verifies only that the user and password specified in the **pds_username** and **pds_password** parameters can log in to the primary data server.

The **test_connection** command does verify that the Replication Agent user login (specified in the **rs_username** and **rs_password** parameters) has **connect source** permission in the primary Replication Server.

See also

- *Replication Agent Connectivity Setup* on page 41

Verifying Replication Agent Connections

Verify Replication Agent connectivity.

1. Log in to the Replication Agent instance with the administrator login.
2. Test Replication Agent network connections:

   ```
   test_connection
   ```

   This command tests all of the connections from the Replication Agent instance you logged in to.

   Note: You can test a specific connection (either the primary data server or the primary Replication Server) by specifying the connection you want to test.

If the **test_connection** command returns a failure, check the Replication Agent system log to determine the cause of the failure. You may also need to check the system log of the server associated with the connection to determine the cause of the failure.

See the *Replication Agent Reference Manual* for information about the **test_connection** command.

Replication Agent Initialization

Replication Agent uses the native transaction log maintained by the primary database to obtain transactions. To support replication, Replication Agent creates some objects in the primary database.

Note: Before you initialize a Replication Agent that has a Replication Agent System Database (RASD), take measures to prevent any database activity during initialization, especially to prevent any DDL operations from changing database objects or schema.

See also

- *Initializing a Replication Agent Instance* on page 76

Object Name Prefix

Before you create the Replication Agent objects, you can specify the object name prefix string to be used to name the objects. You can set this prefix string to avoid conflicts with the names of existing database objects in your primary database.

Replication Agent uses the prefix string to find its objects in the primary database. In Replication Agent for Microsoft SQL Server and Replication Agent for UDB, this prefix string is defined by the value of the **ra_admin_instance_prefix** parameter. In Replication Agent for Oracle, the prefix string is defined by the value of the **ra_admin_prefix** parameter.

Use the **ra_config** command to change the value **ra_admin_instance_prefix** or **ra_admin_prefix**. If you change the value of **ra_admin_instance_prefix** or **ra_admin_prefix** after you initialize Replication Agent, Replication Agent cannot find the objects that use the old prefix.

Initializing Replication Agent

Initialize a Replication Agent instance.

Prerequisites

See the *Replication Agent Primary Database Guide* to verify that the required setup tasks have been performed for your primary database.

Task

1. Log in to the Replication Agent administration port.
2. To define a prefix that uniquely identifies the Replication Agent transaction log you are creating, use:

   ```
   ra_config ra_admin_instance_prefix, string
   ```

   where *string* is a character string of one to three characters that is used as a prefix for all names of the Replication Agent objects created in the primary database.

   Note: The default value of the **ra_admin_instance_prefix** parameter is **ra_**. Unless this string poses a conflict with existing database object names in your primary database, use the default value.
3. Initialize the Replication Agent:

   ```
   ra_admin init
   ```

   When you invoke the **ra_admin** command with the **init** option, Replication Agent:
   - Checks the primary database for compatible settings.
   - Generates a SQL script that is run in the primary database. This script creates the Replication Agent objects in the primary database.

   For Replication Agents that use an RASD, the RASD is initialized with information from the primary database.

   Note: Replication Agent must be initialized before any objects can be marked for replication in the primary database.
4. To verify that the Replication Agent was initialized and that its objects were created in the primary database, enter:

   ```
   ra_admin
   ```

   When you invoke the **ra_admin** command with no options, Replication Agent returns a list of the objects in the primary database, if initialization completed successfully. If no information is returned, Replication Agent has not been initialized, and none of its objects exist in the primary database.

When the Replication Agent is initialized and both primary database and Replication Server connections are defined correctly, you can put the Replication Agent instance in Replicating state.

See also

- *Replication Agent Start-Up* on page 33

Primary Database Object Marking

Individual tables to be replicated must be marked. Tables can be marked explicitly with the **pdb_setreptable** command or automatically during **ra_admin init** processing when the **pdb_automark_tables** configuration parameter is set to true.

Note: The **pdb_automark_tables** parameter is not available for Replication Agent for UDB.

Tables, stored procedures, and sequences (Oracle only) must be marked for replication and have replication enabled for the object (table, stored procedure, or sequence). LOB columns must have replication enabled, and the table that contains the LOB column must be marked for replication and have replication enabled for that table.

There are four types of objects that can be marked for replication in a primary database:

- Tables
- (Oracle and Microsoft SQL Server only) Stored procedures
- Large-object (LOB) columns
- (Oracle and Microsoft SQL Server only) DDL
- (Oracle only) Sequences

If you drop a table in the primary database that is marked for replication and run **pdb_setreptable**, the command returns an incorrect mark status indicating that the table as marked. The **pdb_setreptable** *owner.tablename* command returns a correct error message indicating that the table does not exist in the primary database. If you create the same table again and run **pdb_setreptable**, the command still returns an incorrect mark status. The **pdb_setreptable** *owner.tablename* command returns a correct status indicating that the table is unmarked. These inconsistencies can also occur when you drop a stored procedure in the primary database and run the **pdb_setrepproc** command or drop a sequence in the primary database and run the **pdb_setrepseq** command. To avoid these inconsistencies, run the **ra_truncatearticles** command after you drop any table, stored procedure, or sequence in the primary database.

Marking a Table in the Primary Database

For transactions against a table to be replicated, the primary table in the primary database must be marked for replication, and replication must be enabled for that table.

Prerequisites

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if the table is marked in the primary database:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the primary database table that you want to mark for replication.
   - If **pdb_setreptable** returns information that the specified table is marked and replication is enabled, you need not continue this procedure.
   - If **pdb_setreptable** returns information that the specified table is marked but replication is disabled, skip step 3 and go to step 4 to enable replication for the table.
   - If **pdb_setreptable** returns information that the specified table is not marked, continue this procedure to mark the table for replication.
3. Mark the table for replication and specify the name to use for replication:

   - To mark the table for replication using a replication definition with the same table name, use:

     ```
     pdb_setreptable pdb_table, mark
     ```
   - To mark the table for replication using a replication definition with a different table name, use:

     ```
     pdb_setreptable pdb_table, rep_table, mark
     ```

     where:
     - *pdb_table* is the name of the table in the primary database that you want to mark for replication.
     - *rep_table* is the name of the table in the **with primary table named rep_table** clause in the replication definition for this table.
   - When you mark a table for replication, if the Replication Server replication definition for the table is to be owner qualified, you must specify that the log transfer language (LTL) sent by Replication Agent should also be owner qualified to match the replication definition. To do this, use the **owner** keyword after the **mark** keyword:

     ```
     pdb_setreptable pdb_table, mark, owner
     ```

     where *pdb_table* is the name of the table in the primary database that you want to mark for replication.

If **pdb_dflt_object_repl** is set to true (the default), the table marked for replication with **pdb_setreptable** is ready for replication after you invoke **pdb_setreptable** successfully, and you can skip step 4 in this procedure.

If **pdb_dflt_object_repl** is false, you must enable replication for the table before replication can take place.

4. Enable replication for the table:

```
pdb_setreptable pdb_table, enable
```

To mark or enable all user tables at once, use:

```
pdb_setreptable all, {mark|enable}
```

where **mark** or **enable** are the keywords identifying the action to take against all user tables in the database.

After the table is marked and replication is enabled for the table, you can begin replicating transactions that affect data in that table.

Marking a Stored Procedure in the Primary Database

To replicate invocations of a stored procedure in the primary database, the stored procedure must be marked for replication, and replication must be enabled for that stored procedure.

Prerequisites

For Oracle, DDL replication must be disabled before marking (or unmarking) a stored procedure.

Note: Procedure replication is not available for UDB.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the stored procedure:

```
pdb_setrepproc pdb_proc
```

where *pdb_proc* is the name of the stored procedure in the primary database that you want to mark for replication.

- If **pdb_setrepproc** returns information that the specified stored procedure is marked and replication is enabled, you need not continue this procedure.
- If **pdb_setrepproc** returns information that the specified stored procedure is marked but replication is disabled, skip step 3 and continue this procedure from step 4 to enable replication for the stored procedure.
- If **pdb_setrepproc** returns information that the specified stored procedure is not marked, continue this procedure to mark the stored procedure for replication.

3. Mark the stored procedure for replication:
    - When the function replication definition has the same name as the procedure in the primary database, use:

      ```
      pdb_setrepproc pdb_proc, mark
      ```
    - When the name of the function replication definition is different from the procedure in the primary database, use:

      ```
      pdb_setrepproc pdb_proc, rep_proc, mark
      ```

      where *rep_proc* is the name of the stored procedure in the **with all procedures named rep_proc** clause in the Replication Server function replication definition for this stored procedure.

    If **pdb_dflt_object_repl** is true (the default), the stored procedure marked for replication with **pdb_setrepproc** is ready for replication after you invoke **pdb_setrepproc** successfully, and you can skip step 4 in this procedure.

    If **pdb_dflt_object_repl** is false, you must enable replication for the stored procedure so replication can take place.
4. Enable replication for the marked stored procedure:

    pdb_setrepproc *pdb_proc*, enable

After the stored procedure is marked and replication is enabled for the stored procedure, you can begin replicating invocations of that stored procedure.

Enabling Replication for a LOB Column in the Primary Database

For transactions that affect a LOB column to be replicated, the table that contains the LOB column must be marked for replication and have replication enabled.

Prerequisites

If **pdb_dflt_column_repl** is true (the default), all LOB columns in a table have replication enabled automatically when you mark the table (by invoking **pdb_setreptable**). If **pdb_dflt_column_repl** is false, you must enable replication separately for each LOB column before replication can take place.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the LOB column:

    ```
    pdb_setrepcol pdb_table, pdb_col
    ```

    where:
    - where *pdb_table* is the marked table that contains the LOB column.
    - where *pdb_col* is the LOB column in the primary database.

Note: For Replication Agent for UDB, if **pdb_setrepcol** is invoked with a table containing a "DATE" column, the primary key in the primary table cannot include the "DATE" column.

If **pdb_setrepcol** returns information that the LOB column has replication enabled, you need not continue this procedure.

If **pdb_setrepcol** returns information that the LOB column does not have replication enabled, continue this procedure to enable replication for the column.

3. Enable replication for the LOB column:

   pdb_setrepcol *pdb_table*, *pdb_col*, enable

After replication is enabled for the LOB column, you can begin replicating transactions that affect data in that column.

DDL Replication

To replicate DDL, use **pdb_setrepddl** to set filtering rules accordingly.

You must also set the **ddl_username** and the **ddl_password**. See the *Replication Agent Reference Manual* for details on using the **pdb_setrepddl** command.

Note: DDL replication is not available for UDB.

To replicate DDL:

- Replication Agent requires a unique user name to be supplied that has authority to execute all DDL commands at the standby database.
- Replication Server must have a database-level replication definition with **replicate DDL** set in the definition.

For details about configuration property *ddl_username* and for database-level replication definition, refer to the *Replication Agent Reference Manual*.

Secure Socket Layer

The SSL protocol runs above TCP/IP and below application protocols such as remote method invocation (RMI) or Tabular Data Stream™ (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.

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.

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.

Setting Up SSL Services

Set up SSL services on Replication Agent.

Prerequisites

Review the SSL Plus user documentation and documentation for any third-party SSL security software you are using.

Include Trusted CA Certificates

Modify the Open Client/Server™ trusted roots file to include trusted CA certificates.

The list of known and trusted CAs is maintained in the trusted roots file:

- UNIX – `$SYBASE/RAX-15_5/config/trusted.txt`
- Windows – `%SYBASE%\RAX-15_5\config\trusted.txt`

The system administrator adds and deletes CAs using a standard ASCII-text editor. The trusted roots file is similar in format to a certificate file. The file path is configured through **ssl_certificates_filename** parameter.

By default, Replication Agent recognizes these third-party CAs:

- Thawte
- Entrust
- Baltimore
- VeriSign
- RSA

Creating an Identity File

Create the identity file that contains the concatenation of a certificate and its corresponding encrypted private key, and that is understood by the SSL Plus software.

Prerequisites

Verify that you have the system administrator role before performing this task.

Task

The name and default location of the identity file is the following, where *instance_name* is the name of the Replication Agent instance as specified at start-up:

- UNIX - `$SYBASE/RAX-15_5/`*`instance_name`*`/certificates/`*`instance_name`*`.p12`.
- Windows - `%SYBASE%\RAX-15_5\`*`instance_name`*`\certificates\`*`instance_name`*`.p12`.

1. To place the identity file in a different location, specify the alternate location in the **ssl_identity_filename** entry in the configuration file.
2. To make a successful connection, the common name in the certificate must match the Replication Agent instance name in the directory service.

   Client certificates are not supported.

Configuring SSL from a Client to Replication Agent Server

Enable or disable SSL using the **ra_config** with the **use_ssl** option.

1. Enable **ra_config**, enter:

   ```
   ra_config use_ssl, true
   ```

   Set **use_ssl** to false to disable SSL. By default, SSL is not enabled on Replication Agent. When **use_ssl** is set to false, Replication Agent does not accept SSL connections.

   use_ssl is a static option. You must restart Replication Agent after you change its value.
2. Update the `interfaces` file (`sql.ini`) to include ssl.

   For example:

   ```
   [RAO_SERVER1]
   master=NLWNSCK,localhost,13010,ssl
   query=NLWNSCK,localhost,13010,ssl
   ```
3. Add the CA root certificate that issues the Replication Agent certificate to:
   - UNIX – `$SYBASE/config/trusted.txt`
   - Windows – `%SYBASE%/ini/trusted.txt`

   The **isql** utility reads the `trusted.txt` for server authentication when establishing SSL connection.

Configuring SSL from a Replication Agent to Oracle

Enable or disable SSL from a Replication Agent (as a client) using the **ra_config** with the **pds_use_ssl** option.

1. Enable **pds_use_ssl**, enter:

   ```
   ra_config pds_use_ssl, true
   ```
2. Set **pds_port_number** to Oracle SSL port.

   ```
   ra_config pds_port_number, 2484
   ```
3. (Optional) Set **pds_ssl_sc_dn** to verify if the subject distinguised name (DN) in the server certificate matches this configuration parameter.

Configuring Oracle Primary Database for SSL Connection

Configure Oracle to accept SSL connections from Replication Agent.

1. Configure `listener.ora` to use TCPS protocol.

```
LISTENER = (ADDRESS_LIST= (ADDRESS=(PROTOCOL=tcps)
(HOST=servername)(PORT=2484)))
```

2. Create an Oracle wallet and configure a certificate and its associated private key into the wallet.

   See the Oracle documentations for details.

3. Provide the wallet location in `sqlnet.ora` and `listener.ora`.

   ```
   WALLET_LOCATION=(SOURCE=(METHOD=FILE)
   (METHOD_DATA=(DIRECTORY=/server/wallet/path/)))
   ```

4. Configure SSL_CLIENT_AUTHENTICATION in `sqlnet.ora` and `listener.ora` to turn on or off the client authentication.

   ```
   SSL_CLIENT_AUTHENTICATION=FALSE or
   SSL_CLIENT_AUTHENTICATION=TRUE
   ```

5. If you turn on the client authentication, add the CA root certificate of the Replication Agent certificate to the Oracle wallet.

Configuring SSL Connection Between Replication Agent and UDB

Configure UDB to accept SSL connections from Replication Agent.

1. Create a key database file and set up a self-signed CA certificate for UDB by using the IBM GSK8 library available in the IBM SDK.

   Note: For more information on using the IBM GSK8 library, refer to the IBM documentation.

   This example demonstrates how to create a key database file and set up a self-signed CA certificate for testing purpose:

   a. Delete the existing key database file and create a new key database file called `keydb.kdb` and a stash file called `keydb.sth` by running:

   ```
   gsk8capicmd -keydb -delete -db "c:\cert\keydb.kdb" -pw
   "secret123"
   gsk8capicmd -keydb -create -db "c:\cert\keydb.kdb" -pw
   "secret123" -stash
   ```

   b. Create a self-signed certificate with a label of Selfsigned by running:

   ```
   gsk8capicmd.exe -cert -create -db "c:\cert\keydb.kdb" -pw
   "secret123" -label "selfsigned" -dn "CN=sybase.com"
   ```

   c. Extract the certificate to a file called `keydb.arm` by running:

   ```
   gsk8capicmd.exe -cert -extract -db "c:\cert\keydb.kdb" -pw
   "secret123" -label "selfsigned" -target "c:\cert\keydb.arm" -
   format ascii -fips
   ```

2. Ensure that Replication Agent uses SSL to connect to UDB by running:

   ```
   ra_config pds_use_ssl, true
   ```

3. Edit the `trusted.txt` file and append the contents of the `keydb.arm` file to the `trusted.txt` file.

   By default, the `trusted.txt` file is available in the `$SYBASE/RAX-15_5/config` directory.

   If the `trusted.txt` does not exist, either:

   - Create a new `trusted.txt` file in the `$SYBASE/RAX-15_5/config` directory.
   - Specify a different location of the `trusted.txt` file by using the **ssl_certificates_filename** parameter.
4. Start UDB and restart Replication Agent for UDB.
5. Test the Replication Agent connection for the primary data server by running:

   ```
   test_connection pds
   ```

Configuring SSL Connection Between Replication Agent and Microsoft SQL Server

Configure Microsoft SQL Server to accept SSL connections from Replication Agent.

1. Set up a self-signed CA certificate for Microsoft SQL Server by using the Certificate Creation Tool (`makecert.exe`) or import a genuine CA-signed certificate.

   Note: For more information on using the Certificate Creation Tool, refer to the Microsoft documentation.

   This example demonstrates how to set up a self-signed CA certificate for testing purpose in the Windows local computer certificate store:

   a. Log in to Windows as Administrator.

   b. Open a command prompt and change to the SDK `bin` directory.

   c. Run the following command at the command prompt:

   ```
   makecert -r -pe -n "CN=sybase.com" -b 01/01/2012 -e
   01/01/2020 -eku 1.3.6.1.5.5.7.3.1 -ss my -sr localMachine
   -sky exchange -sp "Microsoft RSA SChannel Cryptographic
   Provider" -sy 12
   ```
2. Ensure that Replication Agent uses SSL to connect to Microsoft SQL Server by running:

   ```
   ra_config pds_use_ssl, true
   ```
3. Configure the common name (CN) of the server (as specified in the distinguished name (DN) of the server certificate) by running:

   ```
   ra_config pds_ssl_sc_cn, “common_name”
   ```

   where, *common_name* is the domain name.
4. Edit the `trusted.txt` file and append the contents of the CA root certificate file to the `trusted.txt` file.

   By default, the `trusted.txt` file is available in the `$SYBASE/RAX-15_5/config` directory.

If the `trusted.txt` does not exist, either:

- Create a new `trusted.txt` file in the `$SYBASE/RAX-15_5/config` directory.
- Specify a different location of the `trusted.txt` file by using the **ssl_certificates_filename** parameter.

5. Start Microsoft SQL Server and restart Replication Agent for Microsoft SQL Server.
6. Test the Replication Agent connection for the primary data server by running:

```
test_connection pds
```

Configuring SSL from a Replication Agent to Replication Server

Enable or disable SSL from a Replication Agent (as a client) using the **ra_config** with the **rs_use_ssl** option.

1. Enable **rs_use_ssl**, enter:

```
ra_config rs_use_ssl, true
```

2. Configure Replication Server to accept SSL connections from Replication Agent.

   See *Enabling or Disabling SSL on Replication Server* in the *Replication Server Configuration Guide* for your platform.
3. (Optional) Set **rs_ssl_sc_dn** to verify if the subject distinguished name (DN) in the Replication Server certificate matches this configuration parameter.

Replication Agent Configuration Requirements

Observe these requirements when configuring Replication Agent.

Primary Database

Configure the primary database:

- Add the Replication Agent user login name to the primary database, and grant the user appropriate permission to be able to perform tasks necessary to support replication.
- Add the Maintenance User login name (as specified in the Replication Server **create connection** command) to the primary database.
- (Oracle only): Enable supplemental logging.

Replication Agent

Configure the Replication Agent instance:

- Make sure that the connection configuration is set correctly for network communications with the primary database, Replication Server, and RSSD.
- (Microsoft SQL Server) Initialize each primary database server. Also, configure the database to allow a remote TCP/IP connection and to allow a remote DAC connection.
- Use the **ra_admin init** command to initialize Replication Agent. This command validates that the primary database is prepared for replication, sets up Replication Agent system

objects in the primary database, and initializes the RASD (Oracle and Microsoft SQL Server only).

Replication Server

Configure the Replication Server:

- Use the Replication Agent user login name, with **connect source** and **create object** permission granted.
- Identify or create the Replication Agent user login name for the RSSD.
- Define the database replication definition and subscription for the primary and standby database.
- Apply the heterogeneous datatype support scripts at the RSSD.
- (Oracle) If Replication Server is version 15.0 or earlier, apply the scripts distributed with Replication Agent to correctly define the Oracle error class.

RSSD

If the Replication Server version is 15.2 or later and you have already created a connection using the Replication Server **create connection** command with the **using profile** clause, your RSSD objects have already been loaded by the connection profile, and you can skip these instructions. If your Replication Server is version 15.0 or earlier, correctly define the Oracle error class in both Replication Server and the RSSD:

- At Replication Server, execute the `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_1_rs.sql` script.
- At the RSSD database, execute the `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_2_rssd.sql` script.
- At Replication Server, execute the `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_3_rs.sql` script.

Multiple Replication Agents

In a Replication Server Multi-Path Replication™ scenario, multiple Replication Agent for Oracle instances are grouped together and share common system resources to coordinate in a Multi-Path Replication solution:

- Every Replication Agent for Oracle instance in a Replication Agent group must have the same value set for the **ra_admin_owner** and **ra_admin_prefix** parameters.
- Each Replication Agent for Oracle instance in a Replication Agent group must have a unique value set for the **ra_admin_instance_prefix** parameter. This parameter is used to distinguish Replication Agent for Oracle instances within a group.

To avoid DDL synchronization errors with multiple Replication Agents, use the **all**, **marked**, and **unmarked** keywords with the **pdb_setrepddl** command.

See the *Replication Agent Reference Manual* and *Replication Agent Primary Database Guide* for details about these parameters and command keywords.

Supplemental Logging

When you create a Replication Agent instance, you can choose to enable supplemental logging of primary key (PK) and unique index (UI) columns at the table level.

Supplemental logging at database level is not essential for Replication Agent while replicating. Instead, you can enable supplemental logging of PK and UI columns at table level for the Oracle system tables and user tables that need to be replicated. After you enable supplemental logging of PK and UI columns at table level, Oracle writes the columns of PK and UI to the redo log file, whenever the table is updated. This reduces the size of redo log file compared to supplemental logging of PK and UI at database level. For a table whose supplemental logging is not enabled, Oracle writes columns of PK and UI to the redo log file, only when any column of PK or UI is updated.

Turning on Supplemental Logging at Table Level and Turning off Supplemental Logging at Database Level

Turn off supplemental logging at database level and turn on supplemental logging at table level.

Note: You can switch between supplemental logging at database level and table level at any point in time. However, you must not turn off supplemental logging at database level before you turn on supplemental logging at table level because it takes some amount of time for switching. During this period, Oracle does not write the PK and UI columns to the redo log file, until the supplemental logging at table level is turned on. This can cause Replication Agent to go to the Replication Down state.

1. Run:

   ```
   ra_admin supplemental_logging_level, table
   ```

   The command generates an SQL script at this location:

   ```
   $SYBASE/RAX-15_5/<instance_name>/script/prepare/
   prepare_pdb_for_table_level.sql
   ```

   Note: You must have a database administrator role to perform step 2.

2. Run the SQL script:

   ```
   $SYBASE/RAX-15_5/<instance_name>/script/prepare/
   prepare_pdb_for_table_level.sql
   ```

   The script turns on table level primary key and unique supplemental logging of the required Oracle system tables, already marked user tables, and Replication Agent system tables.

Turning on Supplemental Logging at Database Level and Turning off Supplemental Logging at Table Level

Turn off supplemental logging at table level and turn on supplemental logging at database level..

1. Run:

```
ra_admin supplemental_logging_level, database
```

The command generates an SQL script at this location:

```
$SYBASE/RAX-15_5/<instance_name>/script/prepare/
prepare_pdb_for_db_level.sql
```

Note: You must have a database administrator role to perform step 2.

2. Run the SQL script:

```
$SYBASE/RAX-15_5/<instance_name>/script/prepare/
prepare_pdb_for_db_level.sql
```

The script turns on database level primary key and unique supplemental logging.

Administration

Review the administrative tasks and procedures for the Replication Agent.

For information about installing the Replication Agent software, see the *Replication Agent Installation Guide*.

Note: Although example procedures in this section show **isql** as the Open Client application used to log in to the Replication Agent administration port, you can use any Open Client (or Open Client-compatible) application to do so.

See also

- *Setup and Configuration* on page 11

Determining the Status of a Replication Agent Instance

The Replication Agent status consists of the current state and activity of the Replication Agent instance.

To determine the current status of Replication Agent:

1. Log in to the Replication Agent instance with the administrator login.
2. Get the current status of the Replication Agent instance:

   ```
   ra_status
   ```

   This command returns the current state of the Replication Agent instance and any current activity, for example:

   ```
   State  Action
   ------ ----------------------------
   ADMIN  Waiting for operator command
   (1 row affected)
   ```

Replication Agent States

Replication Agent can be in any of several states:

These states are:

- Admin – the instance has no connections established to other replication system components but is available to execute administrative commands, such as changing configuration parameters and maintaining the transaction log or the RASD. No replication processing occurs when the Replication Agent instance is in Admin state.

- Replicating – the instance is performing its normal replication processing: scanning the transaction log, processing log records and change-set data, and sending LTL commands to the primary Replication Server. Some administrative commands are not allowed.
- Replicating (Resynchronization) – the instance has been restarted and is resynchronizing the primary and replicate databases.
- Replication Down – replication has stopped due to an error.

The state of a Replication Agent instance can be changed by either:

- An external event that occurs while the Replication Agent is processing replicated transactions (for example, a network error on the Replication Server connection), or
- Operator intervention (for example, invoking a command that changes the Replication Agent state).

From the moment a state-changing event occurs until the Replication Agent instance is actually in the new state, the instance is said to be "in transition." During state transition, some administrative commands are ignored.

Admin State

A Replication Agent instance goes to Admin state when:

- The instance is started in its default state.
- The instance is started with the **ra** utility **-admin** option.
- The Replication Agent **quiesce** or **suspend** command is invoked when Replication Agent is in Replicating state.

In Admin state, the Replication Agent instance is running but has no connection established to the primary Replication Server (or RSSD, if so configured) or the primary database.

You can perform most administrative tasks while the Replication Agent instance is in Admin state, including changing the value of any Replication Agent configuration parameter.

Note: In Admin state, the instance can open a connection to the primary database, if necessary, to process a command that requests results from the primary database.

Replicating State

A Replication Agent instance goes to Replicating state when:

- The instance is started with the **ra** utility **-replicate** option.
- The Replication Agent **resume** command is invoked when Replication Agent is in Admin state.

Note: The Replication Agent instance goes to the Replicating state only if a connection for the primary database has been created in the primary Replication Server. See the *Replication Agent Primary Database Guide*.

In Replicating state, the Replication Agent instance maintains a connection to the primary database and to the primary Replication Server (and RSSD, if so configured), and its Log Reader component scans the transaction log for transactions to replicate.

If the Replication Agent instance has finished processing all of the records in the transaction log, its state continues to appear as Replicating. When the Replication Agent instance reaches the end of the log:

- The Log Reader component log-scanning process "sleeps" according to the values of the **scan_sleep_increment** and **scan_sleep_max** configuration parameters.
- After the Log Transfer Interface (LTI) component finishes processing all of the change sets it received from the Log Reader and sending all of the LTL to the Replication Server, no replication throughput occurs until new replicated transactions appear in the log and the Log Reader scans them.
- The Replication Agent instance remains in Replicating state, unless some other event causes it to go to Admin or Replication Down state.

Replicating (Resynchronization) State

Replication Agent goes into the Replicating (Resynchronization) state after the **resume resync** command is invoked and completes successfully. This state indicates the Replication Agent is resynchronizing the primary and replicate databases. When Replication Agent has completed resynchronizing the primary and replicate databases, it returns to the Replicating state. For information on resynchronization, see the *Replication Server Heterogeneous Replication Guide* and the *Replication Server Administration Guide*. For information on the **resume** command, see the *Replication Agent Reference Manual*.

Replication Down State

A Replication Agent instance goes to the Replication Down state when:

- An unrecoverable error occurs when the instance is in Replicating state.
- A network failure or communication error causes a connection to the primary database or the primary Replication Server to be dropped.

When Replication Agent drops a connection, before it goes to Replication Down state, it first attempts to reestablish the connection using the values recorded in its configuration parameters for that connection. If it cannot reconnect, the Replication Agent instance goes to Replication Down state.

After the error has been resolved, Replication Agent may return to the Replicating state.

Note: Replication Agent behavior in the Replication Down state is the same as behavior in the Admin state, the only difference between the two states being that the Replication Down state is reached through a Replication Agent error.

Replication Agent State Changes

The state of a Replication Agent instance indicates its current operational condition, and determines which administrative tasks you can perform.

Generally, there are only two reasons to change the state of a Replication Agent instance:

- To perform certain administrative or maintenance procedures (change the state from Replicating to Admin)
- To restore normal replication processing (change the state from Admin to Replicating), either after an administrative or maintenance procedure, or after recovery from an error

Replicating State to Admin State

To change the state of the Replication Agent instance from Replicating to Admin, you can use either the **quiesce** or **suspend** command.

See the *Replication Agent Reference Manual* for more detailed information about the **quiesce** and **suspend** commands.

Admin State to Replicating State

To change the state of the Replication Agent instance from Admin to Replicating, you can use the resume command.

See the *Replication Agent Reference Manual* for more detailed information about the **resume** command.

See also

- *Stopping Replication in Replication Agent* on page 71
- *Starting Replication* on page 70

Replication Agent Statistics

The Replication Agent records information about the performance of its internal components whenever it is in Replicating state. You can use this information to tune Replication Agent performance or troubleshoot problems.

Information about Replication Agent performance can be obtained using the **ra_statistics** command. You can also use **ra_statistics** to reset the statistics counters.

Note: Each time the Replication Agent instance goes to Replicating state, statistics counters are reset automatically.

For more information about the **ra_statistics** command and Replication Agent performance statistics, see the *Replication Agent Reference Manual*.

Replication Agent Instance Shutdown

Each Replication Agent instance can be started and shut down independently of all other components in a replication system, and independently of other Replication Agent instances.

Shutting down the Replication Agent instance terminates its process on the host machine.

Note: You can stop all replication processing in the Replication Agent without shutting down the instance.

To shut down a Replication Agent instance, you must log in to the administration port and invoke the **shutdown** command. The **shutdown** command gives you two options:

- Normal shutdown – first quiesces the Replication Agent instance, and then shuts down the instance, terminating its process.
- Immediate shutdown – shuts down the Replication Agent instance and terminates its process immediately, without first quiescing. To use this method, use the **immediate** keyword when you invoke the **shutdown** command.

Note: If the Replication Agent instance is in state transition, it ignores the **shutdown** command with no option (normal shutdown). It does not ignore **shutdown immediate** when it is in any state, including transition from one state to another.

When a Replication Agent instance is shut down normally, it:

- Stops reading the transaction log
- Drops its connection to the primary database
- Finishes processing any transactions it already has in its internal queues
- Drops its connection to the Replication Server after successfully sending LTL for any transactions in its internal queues
- Terminates its process

See also

- *Replication Agent Start-Up* on page 33
- *Stopping Replication in Replication Agent* on page 71

Shutting Down a Replication Agent Instance

Shut down Replication Agent.

1. Log in to the Replication Agent instance with the administrator login.
2. Invoke the **shutdown** command:
   - Shut down the Replication Agent instance normally:

     ```
     shutdown
     ```

- Force an immediate shutdown, regardless of the state of the Replication Agent instance:

  ```
  shutdown immediate
  ```

  This command shuts down and terminates the Replication Agent instance immediately, without first quiescing.

For more detailed information about the **shutdown** command, see the *Replication Agent Reference Manual*.

Starting Replication

Start replication for an instance of Replication Agent.

Prerequisites

Before you attempt to replicate transactions from the primary database, you must complete all of the procedures in "Replication Agent Connectivity Setup." Also, make sure that the Replication Agent instance is running.

Task

When the Replication Agent instance is in Replicating state, it maintains connections to the primary database and the primary Replication Server (and RSSD, if so configured), and its Log Reader component scans the transaction log for transactions to replicate.

1. Log in to the Replication Agent administration port, and verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```
2. Start replication for the Replication Agent instance:

   ```
   resume
   ```

   After you invoke the **resume** command, the Replication Agent instance should go from Admin state to Replicating state.
3. Use the **ra_status** command to verify that the Replication Agent instance is in Replicating state.

Note: The Replication Agent instance goes to the Replicating state only if a connection for the primary database has been created in the primary Replication Server. For more information on creating the primary database connection in Replication Server, see the *Replication Agent Primary Database Guide*.

When the Replication Agent instance is in Replicating state, it is scanning the transaction log for transactions to be replicated and sending LTL to the primary Replication Server.

See the *Replication Agent Reference Manual* for more detailed information about the **resume** command and how Replication Agent starts replication processing.

See also

- *Replication Agent Connectivity Setup* on page 41
- *Troubleshooting* on page 117
- *Replication Agent Start-Up* on page 33

Stopping Replication in Replication Agent

Learn how to stop replication for a Replication Agent instance.

When you stop replication in the Replication Agent:

- The internal Log Reader and Log Transfer Interface components stop their normal replication processing.
- Any open connections to the primary database are released, and the connection to the Replication Server is dropped.
- The Replication Agent instance goes from Replicating state to Admin state.

When the Replication Agent instance is in Admin state, it is running and available to execute administrative commands, but it does not maintain connections to the primary database and the primary Replication Server (and RSSD, if so configured), and it does not process replicated transactions.

Some administrative tasks require the Replication Agent instance to be in Admin state. In a normally operating replication system, you must stop replication in the Replication Agent to perform those tasks.

There are two ways to stop replication in the Replication Agent:

- Quiesce the Replication Agent instance to stop replication gracefully.
- Suspend the Replication Agent instance to stop replication immediately.

Quiescing a Replication Agent Instance

Quiescing the Replication Agent instance stops its replication processing gracefully, ensuring that all transactions from the log have been read and sent to the Replication Server.

To quiesce Replication Agent:

- (Oracle and Microsoft SQL Server) The Log Reader component continues reading operations from the transaction log until there are no operations left; that is, until the Log Reader reaches the end of the log. The Log Reader continues to send change-set data to the Log Transfer Interface component until it finishes processing the last operation it scanned from the log.
- (UDB) The Log Reader component stops reading operations from the transaction log when the current scan is complete. It continues to send change-set data to the Log Transfer Interface component until it finishes processing the last operation it scanned from the log.

- The Log Transfer Interface component stops sending LTL commands to the Replication Server as soon as it finishes processing the last change set it received from the Log Reader.
- When the Log Transfer Interface component is finished processing its input queue and sending the resulting LTL, the Replication Agent instance releases all of its connections to the primary database (if any are open), and drops its connection to the Replication Server (and RSSD, if connected).
- The Replication Agent instance goes from Replicating state to Admin state.

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

   ```
   quiesce
   ```

   After you invoke **quiesce**, the Replication Agent instance should go from Replicating state to Admin state.
3. Verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

Note: If the internal queues are full and the primary database is still recording new activity to the log files when you invoke the **quiesce** command, the quiesce processing may take a while to complete, and there may be a delay before the Replication Agent instance completes the transition to Admin state.

For more detailed information about the **quiesce** command and its processing, see the *Replication Agent Reference Manual*.

Suspending a Replication Agent Instance

Suspending the Replication Agent instance stops its replication processing immediately.

To suspend Replication Agent:

- The Log Reader component stops scanning the transaction log immediately, and the Log Transfer Interface component stops sending LTL commands to the Replication Server immediately.
- All data in the Replication Agent internal queues (input and output queues of the Log Reader and Log Transfer Interface components) is flushed without further processing.
- The Replication Agent instance releases all of its connections to the primary database (if any are open), and drops its connection to the Replication Server (and RSSD, if connected).
- The Replication Agent instance goes from Replicating state to Admin state.

1. Log in to the Replication Agent instance with the administrator login.
2. Suspend replication for the Replication Agent instance:

   ```
   suspend
   ```

After you invoke **suspend**, the Replication Agent instance should go from Replicating state to Admin state.

3. Verify that the Replication Agent instance is in Admin state:

```
ra_status
```

For more detailed information about the **suspend** command, see the *Replication Agent Reference Manual*.

Moving Replication System Components

You can move a Replication Agent instance and primary database logs from one host machine to another.

You may need to move:

- A Replication Agent instance to the same platform
- A Replication Agent instance to a different platform
- Oracle logs

Moving a Replication Agent Instance to the Same Platform

Move a Replication Agent instance to another host machine that uses the same operating system and hardware.

In this example:

- The Replication Agent instance `inst1` is moved from `host1` to `host2`.
- The Replication Agent instance is installed in:
  - (Linux or UNIX) `/opt/sybase/RAX-15_`*`n`*
  - (Windows) `C:\sybase\RAX-15_`*`n`*

  where *n* is 0, 1, 2, or 5.

1. Install Replication Agent on `host2`. Sybase recommends that you use the same installation directory structure as on `host1`.

   Note: The version string of the Replication Agent that you install on `host2` must be exactly the same as the version string of the Replication Agent on `host1` .

2. On `host2`, install one of the following as appropriate for your primary database:
   - JDBC driver (Oracle or Microsoft SQL Server)
   - IBM DB2 client

   Update the CLASSPATH environment variable.
3. If you are moving an instance of Replication Agent for UDB (any version), Replication Agent for Oracle 15.7.1, or Replication Agent for Microsoft SQL Server 15.0, skip this step.

- (Linux or UNIX) Mount the drive containing the online and archive logs for the primary Oracle database.
- (Windows) For Replication Agent for Oracle version 15.6 and earlier, or Replication Agent for Microsoft SQL Server version 15.1 and later, map a network drive to the location containing the primary database transaction and archive logs.

4. Shut down `inst1` on `host1`.
5. Copy the `inst1` instance directory to `host2` from `host1`:
   - (Linux or UNIX) `$SYBASE/RAX-15_`*`n`*
   - (Windows) `%SYBASE%\RAX-15_`*`n`*
6. Update any `interfaces` (Linux or UNIX) or `sql.ini` (Windows) files that refer to the `inst1` instance. Change the host name of the entries to `host2` from `host1`.

   If you use Sybase Control Center (SCC) to monitor the Replication Agent instance, update the host name for the `inst1` instance in SCC.
7. If you chose to install Replication Agent on `host2` in the same installation directory as on `host1`, skip this step.

   If the installation directory on `host2` is different from that used on `host1`, edit these properties in the `$SYBASE/RAX-15_`*`n`*`/inst1/inst1.cfg` (Linux or UNIX) or `%SYBASE%\RAX-15_`*`n`*`\inst1\inst1.cfg` (Windows) configuration file:
   - **asa_install_dir**
   - **lob_cache_directory**
   - **log_directory**
   - **rasd_backup_dir**
   - **rasd_database**
   - **rasd_trace_log_dir**
   - **rasd_tran_log**
   - **rasd_tran_log_mirror**

   Note: Not all Replication Agent versions or types use all of these properties.

   For example, to edit **log_directory** on Linux or UNIX, change:

   ```
   log_directory=<host1_install_dir>/RAX-15_n/inst1/log
   ```

   to:

   ```
   log_directory=<host2_install_dir>/RAX-15_n/inst1/log
   ```

   where *<host1_install_dir>* and *<host2_install_dir>* are the installation paths for Replication Agent on `host1` and `host2`.

   On Windows, use double backslashes in path names. For example, change:

   ```
   log_directory=C\:\\sybase\\RAX-15_5\\inst1\\log
   ```

   to:

   ```
    log_directory=C\:\\software\\syb\\RAX-15_5\\inst1\\log
   ```

8. On `host2`, start `inst1` in the Admin state.
9. If you are moving an instance of Replication Agent for UDB (any version), Replication Agent for Oracle 15.7.1, or Replication Agent for Microsoft SQL Server 15.0 skip this step.

   For Replication Agent for Oracle version 15.6 and earlier, or Replication Agent for Microsoft SQL Server version 15.1 and later, adjust the device paths.
10. Resume `inst1` instance on `host2`.

The instance continues replicating from where it left off before it was moved.

Replication Agent Management

Review the administration and maintenance tasks for Replication Agent for Oracle, Replication Agent for Microsoft SQL Server, and Replication Agent for UDB.

The administration and maintenance tasks for Replication Agent include:

- Initializing Replication Agent
- Deinitializing Replication Agent
- Truncating the transaction log
- Backing up Replication Agent objects in the primary database

Replication for Oracle

The Replication Agent for Oracle uses the Oracle LogMiner to capture replicated transactions. The objects it creates in the primary database facilitate stored procedure replication. These database objects require no routine maintenance.

Depending on the configuration, Replication Agent may also access archived transactions logs (default) or may process only online transaction logs. For information about redo log and archive log files, see the *Replication Agent Primary Database Guide*.

Replication Agent for Microsoft SQL Server

The Replication Agent for Microsoft SQL Server uses the native Microsoft SQL Server log to capture replicated transactions. The objects it creates in the primary database facilitate replication. These database objects require no routine maintenance.

See the *Replication Agent Primary Database Guide* for information about how to automatically truncate the primary database transaction log.

Replication Agent for UDB

The Replication Agent for UDB uses the native DB2 log to capture replicated transactions. The Replication Agent for UDB creates objects in the primary database to store its system data, but those database objects require no routine maintenance.

Depending on the configuration, Replication Agent may process only online transaction logs (default) or may also access archived transactions logs. For information about online transaction log files, archive transaction log files, and log truncation, see the *Replication Agent Primary Database Guide*.

Initializing a Replication Agent Instance

Initialize an instance of Replication Agent, and create the objects it needs in the primary database.

Prerequisites

The Replication Agent instance must be running, and connectivity to the primary database must be established.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine whether objects associated with this Replication Agent instance already exist in the primary database:

   ```
   ra_admin
   ```

   If no Replication Agent objects exist, the **ra_admin** command returns no information. Continue this procedure to initialize Replication Agent. This procedure also creates objects in the primary database that support replication.

   Note: The **ra_admin** command looks for Replication Agent for Microsoft SQL Server and Replication Agent for UDB objects based on the value of the **ra_admin_instance_prefix** configuration parameter and for Replication Agent for Oracle objects based on the value of the **ra_admin_prefix** parameter. If the value of the **ra_admin_instance_prefix** or **ra_admin_prefix** parameter changes after a transaction log was created, the **ra_admin** command cannot find the previously created objects.

   If Replication Agent objects exist in the primary database, the **ra_admin** command returns a list of objects.

   If objects exist for the Replication Agent instance, you do not need to complete this procedure.
3. To use a particular string for the database object name prefix of the transaction log components, use the **ra_config** command to set the value of the **ra_admin_instance_prefix** (Microsoft SQL Server and UDB) or **ra_admin_prefix** (Oracle) parameter:

   ```
   ra_config ra_admin_instance_prefix, XXX
   ```

   where *XXX* is a one- to three-character string that is to be the new value of the **ra_admin_instance_prefix** or **ra_admin_prefix** parameter, and the prefix string used in the database object names when the objects are created. The default value is **ra_**.

   Note: The value of the **ra_admin_prefix_chars** parameter specifies the nonalphabetic characters that are allowed in the prefix string (the value of the **ra_admin_instance_prefix**

or **ra_admin_prefix** parameter). The primary data server may restrict the characters that can be used in database object names. See the *Replication Agent Primary Database Guide* for information about which characters are available for which database.

You can also use **ra_config** to determine the current value of the **ra_admin_instance_prefix** (Microsoft SQL Server and UDB) or **ra_admin_prefix** (Oracle) parameter:

```
ra_config ra_admin_instance_prefix
```

When you invoke **ra_config** and specify a configuration parameter with no value, it returns the current value of that parameter.

4. If your Replication Agent has an RASD, you may have to quiesce the primary database.
   - (Replication Agent for Oracle) You can avoid having to quiesce the primary database if your replication system is configured for database resynchronization. For information on configuring database resynchronization, see the *Replication Server Administration Guide*.
   - (Replication Agent for Microsoft SQL Server) You must quiesce the primary database or otherwise prevent any DDL operations that can change the database objects or schema. Log in to the primary data server with a user login that has appropriate permissions or authority, and quiesce the primary database (or execute the commands necessary to prevent any DDL operations that change the database objects or schema).
   - (Replication Agent for UDB) No action is required because there is no RASD.
5. Initialize the Replication Agent instance, and create its objects in the primary database.

```
ra_admin init
```

Note: When you invoke **ra_admin** with the **init** keyword, the command returns an error message if the Replication Agent objects (using the prefix string currently specified in the **ra_admin_instance_prefix** or **ra_admin_prefix** parameter) already exist in the primary database.

When you invoke the **ra_admin** command with the **init** option, the Replication Agent:
- Checks the primary database for compatible settings.
- Generates a SQL script that is run in the primary database. This script creates the Replication Agent objects.

For Replication Agents that use an RASD, the RASD is initialized with information from the primary database.

Note: You can configure the Replication Agent to generate the script—but not execute it—by setting the value of the **pdb_auto_run_scripts** parameter to false before you invoke the **ra_admin** command. To complete the transaction log creation, you must set **pdb_auto_run_scripts** to true and re-run the **ra_admin init** command. This script is for informational purposes only. Executing it manually in the primary database does not initialize the Replication Agent instance.

If the initialization was successful, the script is stored in a file named `partinit.sql` in the `scripts/xlog/installed` directory.

If the initialization was not successful, the primary database is not changed, and the script is stored in a file named `partinit.sql` in the `scripts/xlog` directory. Check the Replication Agent error log and, if necessary, the primary database error log to determine why the initialization was not successful.

See also

- *Replication Agent Connectivity Setup* on page 41
- *Replication Agent Start-Up* on page 33
- *Troubleshooting* on page 117

Deinitializing a Replication Agent Instance

Deinitialize an instance of Replication Agent, and remove its objects from the primary database.

Prerequisites

The Replication Agent instance must be running in Admin state to remove its objects from the primary database and to deinitialize Replication Agent.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Verify that the Replication Agent objects exist in the primary database:

   ```
   ra_admin
   ```

   If the Replication Agent objects do not exist in the primary database, the **ra_admin** command returns no information about any objects. If no objects exist, you do not need to complete this procedure.

   Note: The **ra_admin** command looks for Replication Agent objects based on the current value of the **ra_admin_instance_prefix** configuration parameter. If the value of the **ra_admin_instance_prefix** parameter changed after the Replication Agent instance was initialized, the **ra_admin** command cannot find the Replication Agent objects that were previously created.

   If objects exist for this Replication Agent instance, the **ra_admin** command returns a list of the names of the objects. Continue this procedure to remove the objects from the primary database.
3. Disable replication for all marked tables in the primary database:

   ```
   pdb_setreptable all, disable
   ```

When you invoke the **pdb_setreptable** command with the **all** and **disable** keywords, Replication Agent disables replication for all marked tables in the primary database.

4. Disable replication for all marked procedures in the primary database:

   ```
   pdb_setrepproc all, disable
   ```
5. Unmark all marked tables in the primary database:

   ```
   pdb_setreptable all, unmark
   ```

   When you invoke the **pdb_setreptable** command with the **all** and **unmark** keywords, Replication Agent removes replication marking from all marked tables in the primary database.
6. Unmark all marked procedures in the primary database:

   ```
   pdb_setrepproc all, unmark
   ```

   When you invoke the **pdb_setrepproc** command with the **all** and **unmark** keywords, Replication Agent removes replication marking from all marked procedures in the primary database. Normally, if any objects in the primary database are marked for replication, you cannot remove the Replication Agent transaction log.
7. Remove the Replication Agent objects:

   ```
   ra_admin deinit
   ```

   When you invoke the **ra_admin** command with the **deinit** keyword, the command returns an error message if no Replication Agent objects exist in the primary database.

   After you invoke the **ra_admin** command with the **deinit** keyword, Replication Agent generates a script that removes the objects from the primary database and deinitializes Replication Agent.

   Note: You can configure Replication Agent to simply build the script, but not execute it, by setting the value of the **pdb_auto_run_scripts** parameter to false before invoking the **ra_admin** command. To complete the removal of the Replication Agent objects, you must set **pdb_auto_run_scripts** to true and re-run the **ra_admin init** command. This script is for informational purposes only. Executing it manually in the primary database does not deinitialize the Replication Agent instance.

If the deinitialization was successful, the script is stored in a file named `partdeinit.sql` file in the `RAX-15_5\inst_name\scripts\xlog\installed` directory.

If the deinitialization was not successful, the script is stored in a file named `partdeinit.sql` in the `RAX-15_5\inst_name\scripts\xlog` directory.

See also

- *Replication Agent Start-Up* on page 33

Forcing Deinitialization

Force the removal of Replication Agent objects that remain after a script execution failure.

Prerequisites

If errors cause a script execution failure, refer to your primary database error logs and the Replication Agent system log to evaluate the errors and determine if any corrective action is necessary.

Task

When you invoke the **ra_admin** command with the **deinit** keyword, Replication Agent creates the `partdeinit.sql` script. When Replication Agent executes this script successfully, all Replication Agent objects are removed from the primary database. In the event that the `partdeinit.sql` script fails for some reason, some Replication Agent objects may be removed from the primary database and some Replication Agent objects may remain. To finish removing Replication Agent objects after a script execution failure:

Invoke **ra_admin**:

```
ra_admin deinit, force
```

When you use the **force** keyword, Replication Agent continues executing the `partdeinit.sql` script, even when errors are encountered, until the script is finished.

Transaction Log Truncation

The Replication Agents for Oracle, Microsoft SQL Server, and UDB support both automatic and manual truncation of the transaction log.

You can enable or disable automatic log truncation at any time, and you can truncate the Replication Agent transaction log manually at any time, with automatic log truncation either enabled or disabled.

Note: Depending on the type of database and the Replication Agent configuration, Replication Agent truncates either the database online logs or archive logs. Sybase recommends that you configure Replication Agent to truncate the database archive logs. See the *Replication Agent Primary Database Guide* for details.

When the Replication Agent truncates its transaction log, either automatically or manually, the truncation point is determined by the most recent LTM Locator received from the primary Replication Server. The Replication Agent truncates not only based on the truncation point, but also keeps at least the same number of archive files as there are log groups.

Automatic Truncation

You have two options for automatic transaction log truncation:

- Automatic truncation each time the Replication Agent receives a new LTM Locator value from the primary Replication Server
- Periodic truncation on a time interval you specify

Enabling Automatic Log Truncation

Automatically truncate the transaction log.

1. Log in to the Replication Agent instance with the administrator login.
2. Enable automatic log truncation, and specify the type of automatic truncation:

   - To enable automatic log truncation at a specified time interval:

     ```
     ra_config truncation_type, interval
     ra_config truncation_interval, N
     ```

     where *N* is the number of minutes between automatic truncations.

     Note: The maximum **truncation_interval** value is **720**.
   - To enable automatic log truncation whenever the Replication Agent receives a new LTM Locator value from the primary Replication Server:

     ```
     ra_config truncation_type, locator_update
     ```

See the *Replication Agent Reference Manual* for more information about the **truncation_interval** and **truncation_type** configuration parameters.

Disabling Automatic Log Truncation

Disable automatic truncation.

1. Log in to the Replication Agent instance with the administrator login.
2. Disable automatic log truncation:

   ```
   ra_config truncation_type, command
   ```

Note: If the value of the **truncation_type** parameter is **interval**, and the value of the **truncation_interval** parameter is **0** (zero), automatic log truncation is effectively disabled.

Truncating the Replication Agent Transaction Log Manually

If automatic log truncation is disabled, you must periodically truncate the Replication Agent transaction log manually.

To manually truncate the transaction log:

1. Log in to the Replication Agent instance with the administrator login.
2. Truncate the Replication Agent transaction log:

   ```
   pdb_truncate_xlog
   ```

   The **pdb_truncate_xlog** command is asynchronous; it does not return success or failure, unless an immediate error occurs.

See the *Replication Agent Reference Manual* for more information about the **pdb_truncate_xlog** command.

Note: As an alternative to the Replication Agent automatic log truncation feature, use a scheduler utility to execute the **pdb_truncate_xlog** command in a script.

Primary Database Backup

The Replication Agent does not support automatically backing up and restoring Replication Agent objects in the primary database.

Sybase recommends that you use the database backup utilities provided by your primary database vendor to periodically back up the Replication Agent transaction log objects in the primary database.

Replication Agent System Database (RASD) Management

Replication Agent uses an embedded database, managed by SQL Anywhere, for the RASD.

RASD

Each instance of Replication Agent depends on the information in its RASD to recognize database structure or schema objects in the transaction log.

When you create a Replication Agent instance, the RASD is created automatically, but it contains no information until you initialize the Replication Agent instance using the **ra_admin init** command. When you initialize a Replication Agent instance, the instance:

- Queries the primary database to get information about the database structure or schema
- Stores information about the database schema in its RASD

Note: Initializing Replication Agent is one of the tasks required to set up the replication system, and it has several prerequisites.

After the RASD is initially populated, its contents are synchronized with the primary database automatically during normal replication (without intervention).

If replication does not occur, the contents of the RASD become stale (not synchronized with the primary database), and you should rebuild them before use.

DDL Commands

Most of the common data definition language (DDL) commands and system procedures executed in the primary database are recorded in the transaction log, and they are replicated to the replicate database. When it processes those DDL transactions for replication, the Replication Agent updates the RASD automatically.

Note: DDL replication is not available in Replication Agent for UDB.

If a DDL command or system procedure produces a change in the primary database schema and the Replication Agent cannot recognize that command or procedure and update its RASD automatically, a replication failure occurs if a subsequent transaction changes data in an object that is not recorded in the RASD. In that event, you must quiesce the primary database and reinitialize Replication Agent to force it to update the RASD.

Each time it processes a DDL transaction that affects an existing database object, the Replication Agent creates a new version of the object metadata in its RASD. The version of each object is identified by the LTM Locator value of the DDL transaction that changed it.

Previous versions of objects must be kept in the RASD long enough to allow system recovery. For example, replaying a transaction that involved an object before it was changed by DDL could produce an error (or data inconsistency) with the current version of the object.

Note: The Replication Agent does not support replaying transactions from a restored transaction log.

Object Versions and LTM Locator Values

The Replication Agent determines which version of each object to use by comparing the current object version string with the current LTM Locator value. If the current LTM Locator value is greater than or equal to the value of the object version, the current object metadata is used. If the current LTM Locator value is less than the value of the object version, a previous version of the object metadata must be used.

Without periodic truncation, the size of the RASD can grow indefinitely, as more and more versions of object metadata are added.

See also

- *Replication Agent Instance Creation* on page 11
- *RASD Truncation* on page 88
- *Updating the RASD* on page 84

Invalid Device Paths

Specifying an invalid device path with the **ra_devicepath** command results in the failure of a subsequent attempt to reinitialize. Also, any log device information in the RASD is cleared.

Note: The **ra_devicepath** command is available only for Replication Agent for Microsoft SQL Server.

For example, this **ra_devicepath** command specifies an invalid path:

```
1> ra_devicepath 1, C:\invalid_path\invalid1.log
2> go
```

The results of the **ra_helpdevice** command show the device status corresponding to this path as “INVALID.” A subsequent attempt to reinitialize fails because of the invalid path:

```
1> ra_admin refresh
2> go
Msg 32000, Level 20, State 0:
```

```
Server 'myserver', Procedure 'ra_admin refresh', Line 1:
Command <ra_admin refresh> failed - Replication initialization
failed because: C:\invalid_path\invalid1.log (The system cannot find
the file specified)
```

After this initialization failure, the **ra_helpdevice** command returns no information because the log device repository has been cleared.

To avoid clearing the log device repository, verify any new device path before updating the log device repository with the **ra_devicepath** command.

Updating the RASD

The RASD is usually updated automatically during normal replication activity.

Note: You should only force an update of the RASD with the recommendation of Sybase Technical Support when the RASD is suspected of being corrupt.

To force an update of the RASD:

1. Log in to the Replication Agent instance with the administrator login.
2. Get the current status of the Replication Agent instance:

   ```
   ra_status
   ```
3. If the Replication Agent is in Admin state, skip this step and go to step 4.

   If the Replication Agent is in Replicating state:

   a) Suspend replication for the Replication Agent instance:

      ```
      suspend
      ```
   b) Verify that the Replication Agent instance is in Admin state:

      ```
      ra_status
      ```
4. Reinitialize the Replication Agent and force it to update the RASD:

   ```
   ra_admin refresh
   ```

   Note: With Replication Agent for Microsoft SQL Server and Replication Agent for UDB, the **ra_admin refresh** command does not overwrite any marking information or configurations. Also, it does not overwrite any existing path information to the log devices in the RASD if all the log information in the RASD matches that returned by the primary data server.

   For each transaction log or device identified in the RASD, if any information does not match the information returned by the primary data server, **ra_admin refresh** overwrites the RASD record for that transaction log or device with the information returned by the primary data server.
5. Resume replication for the Replication Agent instance:

   ```
   resume
   ```

6. Verify that the Replication Agent is in Replicating state:

```
ra_status
```

See also

- *Troubleshooting* on page 117

Updating the Log Device Repository

Replication Agent stores information about primary log devices in its RASD when you initialize the Replication Agent instance. Log device information in the RASD is referred to as the log device repository.

Unlike other information in the RASD, you can update the log device repository at any time using the **ra_updatedevices** command.

Note: If any log device is added, dropped, extended, or moved at the primary database, the Replication Agent log device repository must be updated. If Oracle ASM is being used to manage redo logs and a disk is added to or dropped from an ASM disk group, the device repository should be updated. Sybase recommends that you coordinate all log device changes at the primary database with updating the Replication Agent log device repository.

When you update the log device repository, Replication Agent:

- Queries the primary database for information about all of its log devices.
- Compares the information returned by the primary database with the information recorded in the log device repository.
- Updates the log device repository with the new information returned by the primary database, if it finds information:
  - For existing log devices in the log device repository that does not match the information returned by the primary database, or
  - About new log devices in the information returned by the primary database.

If the path for a log device at the primary site is different from the path for the corresponding log device at the standby site, you must use **ra_devicepath** to specify the path to the log device recorded in the RASD.

Note: The **ra_devicepath** command is available only for Replication Agent for Microsoft SQL Server. The primary database does not have to be quiesced when you update the Replication Agent log device repository.

1. Log in to the Replication Agent instance with the administrator login.
2. Get the current status of the Replication Agent instance:

```
ra_status
```

3. If the Replication Agent is in Admin state, skip this step and go to step 4.

If the Replication Agent is in Replicating state:

a) Suspend replication for the Replication Agent instance:

```
suspend
```

b) Verify that the Replication Agent instance is in Admin state:

```
ra_status
```

4. If you coordinate log device changes at the primary database with updating the Replication Agent log device repository, make the log device changes at the primary database after the Replication Agent is in Admin state.
5. Update the log device repository in the RASD:

```
ra_updatedevices
```

6. To specify the path for a log device, use **ra_devicepath**:

```
ra_devicepath device, dev_path
```

where:

- *device* is the device ID.
- *dev_path* is the alternate path (optional) that Replication Agent should use to access the log device.

Note: You must invoke **ra_devicepath** once for each log device whose path you need to specify.

7. Start replication for the Replication Agent instance:

```
resume
```

You can update the log device repository as often as necessary to accommodate log device changes at the primary database.

Backing Up the RASD

As with any database, you should periodically back up the RASD to prevent data loss in the event of a device failure. Each backup is saved in a separate directory, the name of which consists of the date and time the backup occurred. The most recent backup is also stored in the `backup` directory in addition to its own dated directory.

Note: Sybase recommends that you always back up the RASD before you truncate it. You should also synchronize RASD backups with primary database backups so that, in the event of a primary database restore, the RASD is restored to the same relative point.

The Replication Agent places RASD backup files in the directory identified by the **rasd_backup_dir** configuration parameter. You can back up the RASD at any time, when the Replication Agent instance is in any state.

1. Log in to the Replication Agent instance with the administrator login.
2. Back up the RASD:

```
rasd_backup
```

After the backup completes successfully, the Replication Agent returns a confirmation message.

If the Replication Agent cannot find the directory identified in the **rasd_backup_dir** parameter, or if it cannot write the RASD backup files in that directory (for example, because of a permission problem), it returns an error. You must correct the cause of the error before you can successfully back up the RASD.

Resetting the RASD Administrator Password

Use the **ra_config** command to reset the the Replication Agent System Repository (RASD) administrator password.

1. Log in to the Replication Agent instance with the administrator login.
2. Reset the RASD administrator password:

   ```
    ra_config asa_password, password
   ```

   where, *password* is a new password that complies with the password security requirements.

Restoring the RASD

If the RASD becomes corrupt (for example, because of a device failure), you can restore the database from the most recent backup files.

The Replication Agent retrieves the RASD backup files from the directory identified by the **rasd_backup_dir** configuration parameter. See the *Replication Agent Reference Manual* for more information about the **rasd_backup_dir** parameter.

Note: To restore the RASD, the Replication Agent instance must be in Admin state.

To restore the RASD from backup files:

1. Log in to the Replication Agent instance with the administrator login.
2. Get the current status of the Replication Agent instance:

   ```
   ra_status
   ```
3. If the Replication Agent is in Admin state, skip this step and go to step 4.

   If the Replication Agent is in Replicating state:

   a) Suspend replication for the Replication Agent instance:

   ```
   suspend
   ```
   b) Verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```
4. Restore the RASD:

   ```
   rasd_restore backup_directory
   ```

where *backup_directory* contains the backup you want to restore (for example, `2012-07-04_14.21.34`). If you issue **rasd_restore** without any parameters, only the most recent backup is restored.

After the restore operation completes successfully, the Replication Agent returns a message to confirm that the RASD restore was successful.

If the Replication Agent cannot find the directory identified in the **rasd_backup_dir** parameter, or if it cannot read the RASD backup files in that directory (for example, because of a permission problem), it returns an error. You must correct the cause of the error to restore the RASD.

5. Resume replication for the Replication Agent instance:

```
resume
```

See also

- *Troubleshooting* on page 117

RASD Truncation

To keep the RASD from growing indefinitely, you can periodically truncate older versions of its primary database object metadata.

Note: Back up the RASD using **rasd_backup** before you truncate it.

The RASD stores definitions for two types of database objects:

- Articles – tables and stored procedures that are marked for replication.
- Users – database users who apply transactions in the primary database.

Use the **ra_truncatearticles**, **ra_truncateusers**, and **rasd_trunc_schedule** commands to manage the size of the RASD. For information about these commands, see the *Replication Agent Reference Manual*.

Note: You can truncate the RASD at any time, when the Replication Agent instance is in any state.

See also

- *Backing Up the RASD* on page 86

Truncating Older Versions of Articles in the RASD

Truncate articles from the RASD.

1. Log in to the Replication Agent instance with the administrator login.
2. Truncate articles in the RASD:

```
ra_truncatearticles NNN
```

where *NNN* is an LTM Locator value that identifies the oldest noncurrent version of any article to be kept.

All noncurrent versions of all articles that are less than the LTM Locator value you specify are truncated from the RASD. If the current (most recent) version of an article is older than the version identified by the LTM Locator value, it is not truncated.

Truncating Older Versions of Users in the RASD

Truncate users from the RASD.

1. Log in to the Replication Agent instance with the administrator login.
2. Truncate users in the RASD:

   ```
   ra_truncateusers NNN
   ```

   where *NNN* is an LTM Locator value that identifies the oldest noncurrent version of any user to be kept.

All noncurrent versions of all users that are less than the LTM Locator value you specify are truncated from the RASD. If the current (most recent) version of a user is older than the version identified by the LTM Locator value, it is not truncated.

Default Host and Port Number Configuration

The RASD, an embedded SQL Anywhere database, starts when the Replication Agent starts. By default, the SQL Anywhere **host** value is **localhost**, and the SQL Anywhere **port number** is the Replication Agent **port number +1**.

If you cannot start the Replication Agent instance because these values conflict with the host environment, change them by editing the Replication Agent configuration parameters **asa_host** and **asa_port**, found in the instance configuration file. For example:

```
$SYBASE/RAX-15_5/<instance>/<instance>.cfg
```

where **<instance>** is the name of your Replication Agent instance.

Note: You must restart the Replication Agent after changing these configurations.

Replicated Transactions and Procedure Identification

In a Sybase transaction replication system, the Replication Agent and Replication Server components both provide features that allow you to identify (or select) the transactions that you want to replicate. You do not need to replicate all transactions, or all data-changing operations, in the primary database.

The ability to select transactions for replication is particularly useful when you need to implement a replication system to support an application that uses some of the tables in a database, but not all of them.

By marking tables, you identify the specific tables in the primary database for which transactions are replicated. Transactions that affect the data in marked tables are referred to as replicated transactions.

Note: If a transaction affects data in both marked and unmarked tables, only the operations that affect data in marked tables are replicated.

By marking stored procedures, you identify (or select) the specific procedures in the primary database that are to be replicated as applied functions. When a marked procedure is invoked in the primary database, its invocation is replicated, along with its input parameter values, to the replicate database.

The ability to select procedures for replication is particularly useful when you need to implement a replication system to support an application that uses stored procedures, or when replicating a single procedure invocation is more efficient than replicating numerous, individual data-changing operations that are produced by a single procedure invocation.

Note: Before you can mark tables or stored procedures for replication, you must create the Replication Agent transaction log objects.

See also

- *Replication Agent System Database (RASD) Management* on page 82

Marking and Unmarking Tables

Individual tables to be replicated must be marked. You can mark tables explicitly using the **pdb_setreptable** command or automatically during **ra_admin init** processing when the **pdb_automark_tables** configuration parameter is set to true.

Note: The **pdb_automark_tables** configuration parameter is not supported for UDB.

To replicate transactions that affect the data in a table in the primary database, that table must be marked for replication, and replication must be enabled for the marked table.

Marking a table can be separate from enabling replication for that table. If the value of the **pdb_dflt_object_repl** parameter is true, replication is enabled automatically at the time a table is marked.

See also

- *Enabling and Disabling Replication for Marked Tables* on page 95

Table Marking with Replication Agent for Oracle

When a table is marked for replication with Replication Agent for Oracle, Replication Agent connects to the RASD and records the mark status for the table in the RASD article for that table.

When a table is marked, any subsequent operations that affect the data in that table are replicated.

Table Marking and Unmarking with Replication Agent for Microsoft SQL Server

When a table is marked for replication with Replication Agent for Microsoft SQL Server, Replication Agent logs in to the primary database and executes commands to turn on logging of changes in the Microsoft SQL Server transaction log.

When a table marked for replication is unmarked with Replication Agent for Microsoft SQL Server, Replication Agent logs in to the primary database and executes commands to turn off logging of changes in the Microsoft SQL Server transaction log.

Table Marking with Replication Agent for UDB

Replication Agent for UDB performs tasks at the primary database when a table is marked for replication.

Replication Agent:

- Logs in to the primary database and sets the value of the table **DATA CAPTURE** option to **DATA CAPTURE CHANGES**.
- Adds a row to the Replication Agent marked objects table in the primary database. Each row in the marked objects table lists attributes of a table marked for replication in the primary database.

Table Unmarking with Replication Agent for UDB

Replication Agent for UDB performs tasks at the primary database when a table is unmarked.

Replication Agent:

- Logs in to the primary database and restores the value of the table **DATA CAPTURE** option to the value it had before the table was marked.
- Adds a row to the Replication Agent marked objects table in the primary database. Each row in the marked objects table lists attributes of a table unmarked for replication in the primary database.
- Deletes the table row in the Replication Agent marked objects table.

When a table is unmarked, any subsequent operations that affect the data in that table are ignored (not replicated).

Marking a Table in the Primary Database for Replication

Mark tables for replication with any Replication Agent instance.

Prerequisites

Before you can mark tables for replication, you must create the Replication Agent transaction log objects.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if the table is marked in the primary database:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the primary database table that you want to mark for replication.

   If **pdb_setreptable** returns information that the specified table is marked for replication, you do not need to continue this procedure.

   If **pdb_setreptable** returns information that the specified table is not marked, continue this procedure to mark the table for replication.
3. If there is no table replication definition, only a database replication definition, and no table replication definition is to be added before replication, either:
   a) When the table in the replicate database has the same name as the table in the primary database, use:

   ```
   pdb_setreptable pdb_table, mark
   ```
   b) When the table in the replicate database has a different name from the table in the primary database, use:

   ```
   pdb_setreptable pdb_table, rep_table, mark
   ```

   where *rep_table* is the name of the table in the replicate database.
4. If there is a table replication definition or one is to be added before replication, do one of this regardless of whether or not there is also a database replication definition:
   a) When the primary table in the table replication definition has the same name as the table in the primary database, use:

   ```
   pdb_setreptable pdb_table, mark
   ```

   Note: If the table in the replicate database has the same name as the primary table in the table replication definition, you can use the **with all tables named** clause in the replication definition in the primary Replication Server. For example,

   ```
   create replication definition my_table_repdef
   with primary at data_server.database
   with all tables named pdb_table ...
   ```

   If the table in the replicate database has a different name from the primary table in the table replication definition, the table replication definition must map to the table in the replicate database. For example,

   ```
   create replication definition my_table_repdef
   with primary at data_server.database
   with primary table named pdb_table
   with replicate table named rep_table ...
   ```

b) When the primary table in the table replication definition has a different name from the table in the primary database, use:

```
pdb_setreptable pdb_table, rdpri_table, mark
```

where *rdpri_table* is the name of the primary table in the table replication definition. The table replication definition must map to the table in the replicate database.

Note: If the table in the replicate database has the same name as the primary table in the table replication definition, you can use the **with all tables named** clause in the replication definition in the primary Replication Server. For example,

```
create replication definition my_table_repdef
with primary at data_server.database
with all tables named rdpri_table ...
```

If the table in the replicate database has a different name from the primary table in the table replication definition, the table replication definition must map to the table in the replicate database. For example,

```
create replication definition my_table_repdef
with primary at data_server.database
with primary table named rdpri_table
with replicate table named rep_table ...
```

5. When you mark a table for replication, optionally specify that the table owner must be included when matching to an owner-qualified replication definition.

   - If the owner mode is set, then the owner name is used when matching the replication definition in Replication Agent.
   - If the owner mode is not set, then the owner name is not used by Replication Agent for replication definition name matching.

   To specify that the table owner must be included when matching to an owner-qualified replication definition, use the **owner** keyword after the **mark** keyword:

   ```
   pdb_setreptable pdb_table, mark, owner
   ```

   where*pdb_table* is the name of the table that you want to mark for replication.

   Note: The table owner name returned from the primary database must be the same as the owner name specified in the replication definition for the table.

6. If **pdb_dflt_object_repl** is:

   - true (the default), the table you marked for replication is ready for replication immediately after **pdb_setreptable** returns successfully. You can skip the next step for using **pdb_setreptable** to enable replication for a marked table.
   - false, you must enable replication for the table, as described in the next step.

7. Enable replication for the table:

   ```
   pdb_setreptable pdb_table, enable
   ```

After replication is enabled for the table, the Replication Agent can begin replicating transactions that affect data in that table.

Unmarking a Table in the Primary Database

Unmark primary database tables.

Prerequisites

For UDB only, Replication Agent must be in Admin state when unmarking.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if the table is marked in the primary database:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the primary database table that you want to unmark.

   If **pdb_setreptable** returns information that the specified table is marked, continue this procedure to unmark the table.

   If **pdb_setreptable** returns information that the specified table is not marked, you need not continue this procedure.
3. Disable replication for the table:

   ```
   pdb_setreptable pdb_table, disable
   ```
4. Unmark the table:

   ```
   pdb_setreptable pdb_table, unmark
   ```

   To force the unmark, use:

   ```
   pdb_setreptable pdb_table, unmark, force
   ```
5. Confirm that the table is no longer marked for replication:

   ```
   pdb_setreptable pdb_table
   ```

Note: You can unmark all marked objects in the primary database by invoking the **pdb_setreptable** command with the **all** keyword, unless the primary database is UDB.

Enabling and Disabling DDL Replication

Before you enable DDL replication, you must set the **ddl_username** and **ddl_password** configuration parameters to the user name that Replication Server uses at the replicate database when executing the DDL commands. This user name must be different from the maintenance user that was configured in the Replication Server replicate connection.

For details, see the *Replication Agent Reference Manual*.

Note: DDL replication is available only for Oracle and Microsoft SQL Server. See the *Replication Agent Primary Database Guide* for more information on the DDL commands that are not replicated.

To temporarily suspend replication of DDL, use the **pdb_setrepddl** command to disable replication of DDL. To resume replication of DDL, use the **pdb_setrepddl** command to enable replication. See the *Replication Agent Reference Manual* for details on using the **pdb_setrepddl** command.

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

Enabling and Disabling Replication for Marked Tables

To temporarily stop replication for a marked table (for example, when maintenance operations are performed in the primary database), you can disable replication for a marked table without affecting replication for other tables in the primary database. Then, when you are ready to resume replication from that table, you can enable replication for that table without affecting other tables in the database.

To replicate transactions that affect the data in a table, that table must be marked for replication, and replication must be enabled for the marked table.

Replication Agent for UDB has a marked objects table that contains an entry for each marked table in the primary database. Each marked table row contains a flag indicating whether replication is enabled or disabled for the marked table. Replication Agent for Oracle and Replication Agent for Microsoft SQL Server have articles in the RASD. An article is an object that has a one-to-one relationship to the table and has a marked indicator.

When replication is disabled for a marked object, the marking infrastructure remains in place, but no transactions for that object are sent to Replication Server.

See also

- *Marking and Unmarking Tables* on page 90

Enabling Replication for a Marked Table

Enable replication for a marked table.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if the table is marked in the primary database:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the primary database table for which you want to enable replication.

   If **pdb_setreptable** returns information that the specified table is marked and has replication disabled, continue this procedure to enable replication for the table.

Note: A table must be marked for replication before replication can be enabled or disabled for the table.

3. Enable replication for the table:

   ```
   pdb_setreptable pdb_table, enable
   ```

   After replication is enabled for the table, any transaction that affects the data in that table is captured for replication.

4. Verify that replication is now enabled for the table:

   ```
   pdb_setreptable pdb_table
   ```

Disabling Replication for a Marked Table

Disable replication for a marked table.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if the table is marked in the primary database:

   ```
   pdb_setreptable pdb_table
   ```

   where *pdb_table* is the name of the primary database table for which you want to disable replication.

   If **pdb_setreptable** returns information that the specified table is marked and has replication enabled, continue this procedure to disable replication for the table.

Note: A table must be marked for replication before replication can be enabled or disabled for the table.

3. Disable replication for the table:

   ```
   pdb_setreptable pdb_table, disable
   ```

   After replication is disabled for the table, transactions that affect the data in that table are not captured for replication until replication is enabled again.

4. Verify that replication is now disabled for the table:

   ```
   pdb_setreptable pdb_table
   ```

Enabling and Disabling Replication for LOB Columns

In this document, all columns that contain large-object (LOB) datatypes are referred to as LOB columns, regardless of the actual datatype name used by the primary database vendor. By default, LOB columns within a primary table are marked for replication.

You can enable replication for each LOB column you want to replicate, in addition to marking and enabling replication for the table that contains the LOB column.

- If the value of the **pdb_dflt_column_repl** parameter is true (the default), replication is enabled automatically for all LOB columns in a table at the time the table is marked.

- If the value of the **pdb_dflt_column_repl** parameter is false, replication is not enabled automatically for any LOB columns in a table at the time the table is marked.

When a table is marked for replication and replication is enabled for that table but not for a LOB column in that table, any part of a transaction that affects the LOB column is not replicated. The portion of a transaction that affects all other non-LOB columns is replicated if the table is marked for replication and replication is enabled for the table.

See also

- *Marking and Unmarking Tables* on page 90

Enabling Replication for a LOB Column in a Marked Table

Enable replication for a LOB column.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col
   ```

   - where *pdb_table* is the marked table that contains the LOB column.
   - where *pdb_col* is the LOB column for which you want to enable replication.

   If **pdb_setrepcol** returns information that the LOB column has replication disabled, continue this procedure to enable replication for the column.

   Note: The table that contains the LOB column must be marked for replication before replication can be enabled or disabled for a LOB column.
3. Enable replication for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col, enable
   ```

   After replication is enabled for the LOB column (and if replication is enabled for the table that contains the column), any transaction that affects the data in that column is replicated.
4. Verify that replication is now enabled for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col
   ```

Disabling Replication for a LOB Column in a Marked Table

Disable replication for LOB column.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the LOB column:

   ```
   pdb_setrepcol pdb_table, pdb_col
   ```

   where:
   - where *pdb_table* is the marked table that contains the LOB column.
   - where *pdb_col* is the LOB column for which you want to disable replication.

If **pdb_setrepcol** returns information that the LOB column has replication enabled, continue this procedure to disable replication for the column.

Note: The table that contains the LOB column must be marked for replication before replication can be enabled or disabled for a LOB column.

3. Disable replication for the LOB column:

```
pdb_setrepcol pdb_table, pdb_col, disable
```

After replication is disabled for the LOB column, transactions that affect the data in that column are not replicated unless replication is enabled for that column again.

4. Verify that replication is now disabled for the LOB column:

```
pdb_setrepcol pdb_table, pdb_col
```

Marking and Unmarking Stored Procedures

Replication Agent supports Replication Server function replication by replicating the invocation of stored procedures in the primary database.

Replication Agent can replicate both applied functions and request functions:

- Applied functions are stored procedures that are executed in the primary database and generate transactions that affect data in the primary database.
- Request functions are stored procedures that are invoked in one database (for example, a replicate database), then executed in another database (for example, a primary database).

Note: Stored procedure replication is not supported for UDB.

Replication Agent does not distinguish between these two function types, except to supply a specific user and password for use with request functions. If you are using request functions, you must supply the **function_username** and **function_password** configuration parameters.

For more information about applied and request functions, see *Replication Server Administration Guide > Managing Replicated Functions*.

To replicate a stored procedure invoked in a primary database, the stored procedure must be marked for replication, and replication must be enabled for that stored procedure. (This is analogous to marking and enabling replication for tables.)

Note: Marking a stored procedure for replication is separate from enabling replication for the stored procedure. If the value of the **pdb_dflt_object_repl** parameter is true, replication is enabled automatically at the time a stored procedure is marked.

If a marked stored procedure performs operations that affect a marked table, the operations that affect the marked table are not captured for replication; only the invocation of the marked stored procedure is replicated.

When you mark a stored procedure for replication, Replication Agent creates a shadow-row procedure for that stored procedure. Replication Agent for Oracle and Replication Agent for UDB also modify the marked stored procedure by inserting a new:

- First step to execute the associated shadow-row procedure
- Last step to reexecute the shadow-row procedure with different parameters.

To temporarily suspend replication of a marked stored procedure (for example, when database maintenance operations are performed in the primary database), you can disable replication for the stored procedure.

When you unmark an object that has been marked for replication, the transaction log objects that were created to facilitate the replication for that object are removed from the primary database.

For more information on the Replication Server function replication feature, see the *Replication Server Administration Guide*.

See also

- *Enabling and Disabling Replication for Stored Procedures* on page 102

Marking a Stored Procedure for Replication

Mark stored procedures for replication.

Prerequisites

Before you can stored procedures for replication, you must create the Replication Agent transaction log objects.

For Oracle, DDL replication must be disabled during the marking of stored procedures. Because marking of a stored procedure modifies that stored procedure, you must first disable DDL replication to prevent the marking modifications from replicating to the replicate site.

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the stored procedure that you want to mark for replication.

   - If **pdb_setrepproc** returns information that the specified stored procedure is marked, you do not need to continue this procedure
   - If **pdb_setrepproc** returns information that the specified stored procedure is not marked, continue this procedure to mark the stored procedure for replication.
3. If there is no function replication definition, only a database replication definition, and no function replication definition is to be added before replication, either:
   - When the procedure in the replicate database has the same name as the procedure in the primary database, use:

```
pdb_setrepproc pdb_proc, mark
```

- When the procedure in the replicate database has a different name from the procedure in the primary database, use:

  ```
  pdb_setrepproc pdb_proc, rep_proc, mark
  ```

  where *rep_proc* is the name of the procedure in the replicate database.

4. If there is a function replication definition, or if a replication definition is to be added before replication, do one of the following, regardless of whether or not there is also a database replication definition:

   - When the function replication definition has the same name as the procedure in the primary database, use:

     ```
     pdb_setrepproc pdb_proc, mark
     ```

     - If the procedure in the replicate database has the same name as the function replication definition, there is no need to use the **deliver as** clause. For example,

       ```
       create function replication definition pdb_proc
       with primary at data_server.database ...
       ```

     - If the procedure in the replicate database has a different name from the name of the function replication definition, the function replication definition must map to the procedure in the replicate database. For example,

       ```
       create function replication definition pdb_proc
       with primary at data_server.database
       deliver as 'rep_proc' ...
       ```

   - When the name of the function replication definition is different from the procedure in the primary database, use:

     ```
     pdb_setrepproc pdb_proc, rdpri_proc, mark
     ```

     where *rdpri_proc* is the name of the function replication definition. The function replication definition must map to the procedure in the replicate database.

     - If the procedure in the replicate database has the same name as the function replication definition, there is no need to use the **deliver as** clause. For example,

       ```
       create function replication definition rdpri_proc
       with primary at data_server.database ...
       ```

     - If the procedure in the replicate database has a different name from the function replication definition, the function replication definition must map to the procedure in the replicate database. For example,

       ```
       create function replication definition rdpri_proc
       with primary at data_server.database
       deliver as 'rep_proc' ...
       ```

5. Enable replication for the marked stored procedure:

   ```
   pdb_setrepproc pdb_proc, enable
   ```

   After replication is enabled for the stored procedure, you can begin replicating invocations of that stored procedure in the primary database.

Note: If your stored procedure is in Oracle and you disabled DDL replication during stored procedure marking, remember to reenable DDL replication. Because marking a stored procedure modifies it, you must first disable DDL replication to prevent the marking modifications from replicating to the standby site.

See also

- *Enabling and Disabling DDL Replication* on page 94

Unmarking a Stored Procedure

When you unmark a stored procedure, Replication Agent removes the transaction log objects that were created when the stored procedure was marked. For Oracle, DDL replication must be disabled during the unmarking of stored procedures.

To unmark stored procedures:

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the stored procedure that you want to unmark.

   - If **pdb_setrepproc** returns information that the specified stored procedure is marked, continue this procedure to unmark the stored procedure.
   - If **pdb_setrepproc** returns information that the specified stored procedure is not marked, you do not need to continue this procedure.
3. Disable replication of the stored procedure:

   ```
   pdb_setrepproc pdb_proc, disable
   ```
4. Unmark the stored procedure:

   ```
   pdb_setrepproc pdb_proc, unmark
   ```

   To force the unmark, use:

   ```
   pdb_setrepproc pdb_proc, unmark, force
   ```
5. Confirm that the stored procedure is no longer marked for replication:

   ```
   pdb_setrepproc pdb_proc
   ```

You can unmark all marked stored procedures in the primary database by invoking **pdb_setrepproc** with the **all** keyword.

Note: If your stored procedure is in Oracle and you disabled DDL replication during stored procedure marking, remember to reenable DDL replication. Because marking a stored procedure modifies it, you must first disable DDL replication to prevent the marking modifications from replicating to the standby site.

See also

- *Enabling and Disabling DDL Replication* on page 94

Enabling and Disabling Replication for Stored Procedures

To temporarily suspend replication of a stored procedure, use the **pdb_setrepproc** command to disable replication for the marked stored procedure. When you are ready to resume replication of the marked stored procedure, use the **pdb_setrepproc** command to enable replication.

Note: Procedure replication is not supported for UDB.

To replicate invocations of a stored procedure in the primary database, the stored procedure must be marked for replication, and replication must be enabled for that stored procedure; no procedures are marked by default for replication.

Marking a stored procedure for replication is separate from enabling replication for the stored procedure.

See also

- *Marking and Unmarking Stored Procedures* on page 98

Enabling Replication for a Marked Stored Procedure

Enable replication of stored procedures.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the stored procedure stored procedure you want to enable replication for.

   If **pdb_setrepproc** returns information that the specified stored procedure is marked and has replication disabled, continue this procedure to enable replication for the stored procedure.

   Note: A stored procedure must be marked for replication before replication can be enabled or disabled for the stored procedure.
3. Enable replication for the marked stored procedure:

   ```
   pdb_setrepproc pdb_proc, enable
   ```

   After replication is enabled for the stored procedure, any invocation of that stored procedure is replicated.
4. Verify that replication is now enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

Disabling Replication for a Marked Stored Procedure

Disable replication of stored procedures.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

   where *pdb_proc* is the name of the stored procedure for which you want to disable replication.

   If **pdb_setrepproc** returns information that the specified stored procedure is marked and has replication enabled, continue this procedure to disable replication for the stored procedure.

   Note: A stored procedure must be marked for replication before replication can be enabled or disabled for the stored procedure.
3. Disable replication of the stored procedure:

   ```
   pdb_setrepproc pdb_proc, disable
   ```

   After replication is disabled for the stored procedure, any invocation of that stored procedure is not captured for replication until replication is enabled again.
4. Verify that replication is now disabled for the stored procedure:

   ```
   pdb_setrepproc pdb_proc
   ```

Marking and Unmarking Oracle Sequences

Replication Agent supports replication of sequences in the primary database. To replicate a sequence invoked in a primary database, the sequence must be marked for replication and replication must be for all of that sequence. (This is analogous to marking and enabling replication for tables.)

Note: Marking a sequence for replication is separate from enabling replication for the sequence. If the value of the **pdb_dflt_object_repl** parameter is true, replication is enabled automatically at the time a sequence is marked

Oracle does not log information every time a sequence is incremented. Sequence replication occurs when the Replication Agent captures the system table updates that occur when the sequence's cache is refreshed. Therefore, the sequence value replicated when a sequence is marked for replication is the “next” sequence value to be used when the current cache expires. The result is that not every individual increment of a sequence is replicated, but the standby site always has a value greater than the available cached values at the primary site.

Note: Sequence replication is supported only for Oracle.

To temporarily suspend replication of a marked sequence, disable replication for the sequence.

See also

- *Enabling and Disabling Replication for Sequences* on page 105
- *Unmarking a Sequence* on page 105

Marking a Sequence for Replication

Mark an Oracle sequence for replication.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence that you want to mark for replication.

   - If **pdb_setrepseq** returns information that the specified sequence is marked, you do not need to continue this procedure.
   - If **pdb_setrepseq** returns information that the specified sequence is not marked, continue this procedure to mark the sequence for replication.
3. Mark the sequence for replication.

   pdb_setrepseq allows you to mark the primary sequence to be replicated and specify a different sequence name to use in the replicate database.

   - To mark the sequence for replication when the sequence name you want to increment at the replicate site has a different name, use:

     ```
     pdb_setrepseq pdb_seq, mark
     ```

     Note: Replicating a sequence with a different name than the provided name is consistent with other marking commands but is not a typical configuration.
   - To mark the sequence for replication using a different sequence name, use:

     ```
     pdb_setrepseq pdb_seq, rep_seq, mark
     ```

     where *rep_seq* is the name of the sequence in the replicate database.

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

     If the value of **pdb_dflt_object_repl** is:

     - true (the default) – The sequence marked for replication with **pdb_setrepseq** is ready for replication after you invoke **pdb_setrepseq** successfully. You can skip step 4 in this procedure.
     - false – You must enable replication for the sequence before replication can take place.

4. Enable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, enable
   ```

After replication is enabled for the sequence, you can begin replicating invocations of that sequence in the primary database.

Unmarking a Sequence

Unmark an Oracle sequence.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence that you want to unmark.

   - If **pdb_setrepseq** returns information that the specified sequence is marked, continue this procedure to unmark the sequence.
   - If **pdb_setrepseq** returns information that the specified sequence is not marked, you do not need to continue this procedure.
3. Disable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, disable
   ```
4. Unmark the sequence:

   ```
   pdb_setrepseq pdb_seq, unmark
   ```

   To force the unmark, use:

   ```
   pdb_setrepseq pdb_seq, unmark, force
   ```
5. Confirm that the sequence is no longer marked for replication:

   ```
   pdb_setrepseq pdb_seq
   ```

Enabling and Disabling Replication for Sequences

To temporarily suspend replication of a marked sequence, you can use the **pdb_setrepseq** command. When you are ready to resume replication of the marked sequence, use **pdb_setrepseq** again to reenable replication.

Note: By default, no sequences are marked for replication.

To replicate updates of a sequence in the primary database, the sequence must be marked for replication and replication must be enabled for that sequence.

Marking a sequence for replication is separate from enabling replication for the sequence.

See also

- *Marking a Sequence for Replication* on page 104

Enabling Replication for a Marked Sequence

Enable replication for an Oracle sequence.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence for which you want to enable replication.

   If **pdb_setrepseq** returns information that the specified sequence is marked and has replication disabled, continue this procedure to enable replication for the sequence.

   Note: A sequence must be marked for replication before replication can be enabled or disabled for the sequence.
3. Enable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, enable
   ```

   After replication is enabled for the sequence, any invocation of that sequence is replicated.
4. Verify that replication is now enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

Disabling Replication for a Marked Sequence

Disable replication for an Oracle sequence.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence for which you want to disable replication.

   If **pdb_setrepseq** returns information that the specified sequence is marked and has replication enabled, continue this procedure to disable replication for the sequence.

   Note: A sequence must be marked for replication before replication can be enabled or disabled for the sequence.
3. Disable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, disable
   ```

   After replication is disabled for the sequence, any invocation of that sequence is not captured for replication until replication is enabled again.
4. Verify that replication is now disabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

Replication Agent Tuning

Tune or optimize Replication Agent performance by adjusting some of the Replication Agent configuration parameters.

You can set or change a Replication Agent configuration parameter with the **ra_config** command.

Because the Replication Agent overwrites its entire configuration file whenever **ra_config** or **ra_set_login** is invoked, Sybase recommends that you do not edit the configuration file. Also, each Replication Agent instance reads its configuration file only at start-up. You must use the **ra_config** command if you want a new configuration parameter value to take effect before the instance is shut down and restarted.

All Replication Agent configuration parameters can be changed when the Replication Agent instance is in Admin state. Some configuration parameters cannot be changed when the instance is in Replicating state.

For more information about the **ra_config** command and Replication Agent configuration parameters, see the *Replication Agent Reference Manual*.

Tuning Customization

Generally, the Replication Agent default configuration values provide optimal performance.

However, there may be certain situations where the configuration should be changed to suit or optimize your particular environment.

Adjusting the Size and Volume of the Replication Agent System Logs

By default, the system logs produced by the Replication Agent are a fixed size. They roll over occasionally to prevent continual disk consumption.

You can increase the size of a log and adjust the number of backup files to save log data for a longer period of time. You can decrease these parameters to increase the unused space in your environment.

1. Log in to the Replication Agent administration port, and verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

2. Set the values of these Replication Agent configuration parameters for the primary database. Increase these values to increase the size and number of backup files. Decrease these values to make more space available in your environment:

   ```
    ra_config log_backup_files, n
   ```

   ```
    ra_config log_wrap, m
   ```

Spinning Prevention

Replication Agent uses the configuration parameters **scan_sleep_increment** and **scan_sleep_max** to pause scanning when the end of the log is reached. This prevents Replication Agent from continually “spinning” on the end of the log.

By default, Replication Agent pauses up to 60 seconds before a new transaction appears because it was sleeping. When you need the maximum possible latency for a transaction to be less than the 60-second default, you can reduce the scan parameters. This results in additional CPU usage when the end of the log is reached.

Conversely, if CPU maximization is a greater concern than latency, you can increase these parameters to allow Replication Agent to use less CPU on an inactive log, at the cost of having the latency of the next transaction increased.

Note: These parameters have effect only when the end of the log has been reached and there is no additional activity to be replicated. By default, Replication Agent immediately rescans (without pause) when the end of the log has not been reached.

Parameters and Variables

The parameters that affect performance and tuning are listed here.

Parameters and Defaults at Instance Creation

Under most circumstances, Replication Agent provides the best performance when you create an instance with these default configuration property values:

Parameter	Default
column_compression	true
compress_ltl_syntax	true
lti_batch_mode	true
lti_formatter_count	3
lti_max_buffer_size	5000
lti_update_trunc_point	10000
ltl_batch_size	40000
ltl_origin_time_required	false
ltl_send_only_primary_keys	true
rs_packet_size	2048
structured_tokens	true

Parameter	Default
use_rssd	true

Note: If the size of an LTL command is larger than the value of **ltl_batch_size**, the LTL command is sent to Replication Server separately, without running in batch mode. You can improve Replication Agent performance by increasing the value of **ltl_batch_size**.

Parameters Affecting Performance Throughput

These configuration properties can affect performance throughput and may need tuning based on environmental factors, such as hardware resources, primary database schema, and transaction profile:

Parameter	Default
log_read_block_count	64
lr_max_op_queue_size	1000
lr_max_scan_queue_size	1000
lti_max_buffer_size	5000
rasd_object_cache_size	1000

Note: The **log_read_block_count** parameter is used only for Microsoft SQL Server databases.

Depending on the replication environment and system resources as well as the database schema and transaction profile, the maximum memory configuration can also affect performance throughput. The Replication Agent maximum memory configuration is determined by the RA_JAVA_MAX_MEM environment variable if this variable has been set.

Parameters Affecting Performance Latency

These configuration parameters affect performance latency. However, the defaults for these parameters are prioritized for performance throughput, and adjustments made to these parameters to decrease performance latency may negatively affect performance throughput.

Parameter	Default
dump_batch_timeout	5
scan_sleep_increment	5
scan_sleep_max	60

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

See also

- *Default Heap Size* on page 125

Optimal Performance Tuning

Knowledge of the replication environment and Replication Agent statistics is required to tune Replication Agent for optimal performance. A trial-and-error approach may be necessary to achieve optimal performance for each environment.

Memory Use

The maximum memory available to Replication Agent is the primary consideration: If too little memory is allocated for the configuration, schema, and transaction profile, Java Virtual Machine (JVM) garbage collection can degrade performance significantly. There are five Replication Agent statistics that provide information about memory use. Specifically, by monitoring the percentage of maximum memory used over time, you can determine whether or not the amount of memory is sufficient for a configuration.

The queue (or buffer) and cache size parameters are the primary configuration parameters that determine memory use. Using the **ra_statistics** or the **ra_statrack** command to monitor the queue and cache sizes, along with the percentage of memory used, can help you determine whether to change the configuration to accommodate the amount of memory use allowed or, conversely, whether to change the amount of memory use allowed to accommodate the configuration. By decreasing the queue and cache sizes, the percentage of maximum memory use should decrease, resulting in decreased garbage collection and improved performance. Conversely, increasing the maximum memory available should result in improved performance. In general, if the amount of memory used never drops below 50% of the maximum, garbage collection may be hampering Replication Agent performance.

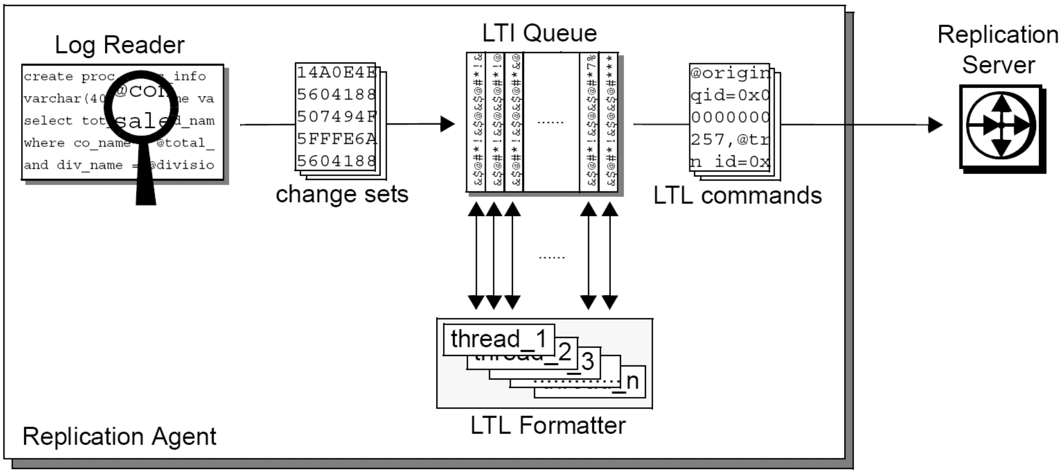
Threads

Replication Agent is a multithreaded application, and several threads are active during processing, even though the log records are usually read and processed serially. This means that, in most cases, each thread is dependent on another thread, so a bottleneck in one thread can result in poor replication performance in the entire system.

Parallel LTL Formatting

Replication Agent can format LTL commands in parallel, improving performance on machines with multiple processors. The LTL formatter is an internal component of Replication Agent that converts change sets into LTL commands. In versions of Replication Agent earlier than 15.6, the LTL formatter has been a performance bottleneck because items in the log transfer interface (LTI) queue could be processed only serially. Replication Agent now provides a multithreaded LTL formatter that processes items in the LTI queue in parallel. This may improve performance on machines with multiple processors.

Figure 4: Replication Agent LTL Formatting



Replication Agent uses two configuration parameters for configuring the size of the LTI queue and the number of LTL formatter threads: **lti_max_buffer_size** and **lti_formatter_count**. Replication Agent also provides LTI statistics to assist you in performance and tuning.

Statistics and Queue Bottlenecks

Replication Agent statistics, analyzed over time, identify bottlenecks in Replication Agent. The queue (or buffer) size statistics provide a likely indication of where a Replication Agent bottleneck is occurring. For example, if the Log Transfer Interface (LTI) inbound queue is at or near capacity, but the LTI outbound queue is at or near zero, the LTI inbound queue—where LTL is formatted—is probably the bottleneck decreasing replication performance. In this way, queue sizes can identify specific areas of trouble:

- An LTI outbound queue size at or near capacity indicates that the Replication Agent is waiting on Replication Server. In this case, all queues in the Replication Agent are also at or near capacity.
- An LTI inbound queue size at or near capacity but with an LTI outbound queue size at or near zero indicates a bottleneck in Replication Agent LTL formatting. This may result from the primary database character set being different than the Replication Server character set, resulting in Replication Agent having to convert character data between character sets. Alternately, the bottleneck may result from the Replication Server RSSD not being used effectively, either because the RSSD is not being used at all—the **use_rssd** parameter is set to **false**—or because minimal columns functionality—as determined by the **column_compression** and **lti_send_only_primary** keys parameters—is not being used.
- A Log Reader operation queue size at or near capacity but with LTI queue sizes at or near zero indicates a bottleneck in Replication Agent operation processing. This may result from a problem with RASD access or with a cache configuration.

- A queue size at or near zero may indicate a bottleneck on the log scan, since the Log Reader scan queue is the first queue used in Replication Agent processing. This bottleneck may result from input-output inefficiency, possibly due to the read block count being too low or to Replication Agent having to read a log over the network rather than locally.

Character Set Conversion
Character set differences in the replication environment can also affect replication performance. For example, if the primary database character set is different from the Replication Server character set, Replication Agent performs character set conversion for all character data that is replicated. If Replication Agent has to perform character set conversion, a warning message is logged in the Replication Agent system log on replication start-up.

See also

- *Available Memory* on page 124

End-to-End Tuning

It may be beneficial in some situations to sacrifice optimal Replication Agent performance to achieve better Replication Server performance and better end-to-end performance through the entire replication system.

For example, in an Oracle-to-ASE replication system using High Volume Adaptive Replication (HVAR), better end-to-end replication performance may be achieved using the same Replication Agent configuration that is used for autocorrection. When a table is configured for autocorrection replication, the **column_compression** and **ltl_send_only_primary_keys** parameters are ignored for the table, and the primary Oracle database is configured to log all columns. This decreases Replication Agent performance but may improve the overall performance of the replication system.

HVAR and Oracle

No primary database reconfiguration is necessary for UDB or Microsoft SQL Server to take advantage of HVAR. Oracle, by default, logs only minimal columns for update operations, so supplemental logging must be enabled for all columns in tables marked for replication.

To enable supplemental logging from Replication Agent for Oracle, use the **ra_set_autocorrection** command as described in the *Replication Agent Reference Manual*.

Note: Replication Agent for Oracle does not support use of the autocorrection feature for large-object (LOB), *LONG*, *LONG RAW*, or user-defined datatypes. See the *Replication Agent Primary Database Guide* for information about the limitations of the autocorrection feature.

Replication Agent Security

Replication Agent provides enhancements to operate in a secure network environment.

User Authentication

An instance requires a user ID and password when you login. You must provide a user ID and password when you create a Replication Agent instance using the **ra_admin** utility, either at the command line or in the response file.

The default Replication Agent password policy requires a password to be at least six characters but not more than 255 characters. For the RASD, Replication Agent automatically generates a user ID and password. You can change this password later.

Note: LDAP is not supported. Replication Agent supports only a single administration login.

Configurable Password Policy

You can configure a flexible and secure password policy for Replication Agent using the password parameters.

The password parameters are:

- *min_password_len*
- *max_password_len*
- *password_lowercase_required*
- *password_uppercase_required*
- *password_numeric_required*
- *password_special_required*
- *password_expiration*

By default, the *min_password_len* parameter value is six characters, and the *max_password_len* paramter value is 255 characters. All other password parameters are disabled.

See the *Replication Agent Reference Manual* for more information about the password parameters.

Dynamic Password Encryption

Replication Agent uses a dynamic key for password encryption.

When you create a Replication Agent instance, a random key is generated automatically and stored in the **instance_rand** configuration parameter.

You can use the **ra_regenerate_keys** command to regenerate the value of the **instance_rand** configuration property. When the **ra_regenerate_keys** command is invoked, all encrypted passwords in the userinfo and password tables are reencrypted.

Secure Sockets Layer

The Replication Agents for Oracle, Microsoft SQL Server, and IBM DB2 UDB support the use of the secure sockets layer (SSL) for connections to and from Replication Agent instances.

The SSL protocol runs above TCP/IP and below application protocols such as remote method invocation (RMI) or Tabular Data Stream™ (TDS). SSL is supported for connection with the execption of the connection to the RSSD.

As a client, a Replication Agent instance can use SSL in connecting to servers, including:

- Replication Server
- Primary data server

The Replication Agent administrator and client applications can use SSL to encrypt connections to Replication Agent.

Replication Agent does not support SSL to the RSSD.

Note: The Replication Agent connectivity to the RSSD can be disabled. The Replication Agent connectivity to the RSSD is only for informational purposes intended to improve the replication performance.

See the *Replication Agent Reference Manual*.

Auditing

Replication Agent logs information about successful and unsuccessful login attempts and command execution.

Replication Agent also logs information when configuration properties are changed.

Best Practices for Replication Agent Security

Observe these best practices.

Replication Agent Confidential Data at the Operating System Level

When you create a resource file to create, configure, or initialize a Replication Agent instance, the resource file contains passwords in plain text, as well as host, port number, and username information.

To protect this confidential information, before creating the resource file, set the file and folder permissions for exclusive read and write access to the owner:

- For UNIX, set the umask value to `077`.
- For Windows, set the folder security properties.

When the response file is no longer needed, delete it.

Replication Agent does not Restrict Access to Administrative Functions to Local Access

After you install, configure, and start a Replication Agent instance, you can log in either remotely (from another host) or locally (from the same host), depending on where you have installed Replication Server and **isql** or Sybase Control Center for Replication.

You can restrict access to administrative functions by changing the interfaces file entry to localhost or 127.0.0.1. For example, change:

```
[my_ra]
 query=NLWNSCK,my_host,10002
```

To:

```
[my_ra]
 query=NLWNSCK,localhost,10002
```

Troubleshooting

Review the basic troubleshooting procedures for Replication Agent and the replication system.

Two types of failures can occur in your replication system: command and replication. Command failures occur when you are in setting up your replication system. They return specific error messages that help you troubleshoot the problem. Replication failures occur after the replication system has been set up and replicated transactions do not appear in the replicate database.

Often, problems that prevent replication from occurring do not result in an error message from any replication system component. For example, a component may not recognize a problem in its own configuration that prevents replication from starting.

In a functioning Replication Agent system—one that has previously replicated transactions successfully—most system problems result in an error message from one or more of the system components. However, some problems that interrupt replication might not be interpreted as errors by the system components. In that case, replication fails but no error message is returned.

Command Errors

Troubleshoot error messages that are returned from a command or that appear in the log file.

Connection Refused

Error message: `Could not connect to <jdbc:sybase:Tds:localHost:5001/emb>: JZ006: Caught IOException: java.net.ConnectException: Connection refused: connectJZ006:`

Explanation: The Replication Agent attempted to connect to a Sybase server on a host called *localHost* and port 5001. The error indicates that no server was found.

Action: This is a usually a configuration error. Either the server that Replication Agent is attempting to connect to is not running, or the host and port information configured in Replication Agent is incorrect.

- Verify that your server is running.
- Verify that your Replication Agent is configured with the correct host and port information.
- Test your connection after you have verified them.

See also

- *Connection Configuration Parameter Settings* on page 44
- *Network Connectivity Test* on page 49

Replication Agent Failure

Troubleshoot Replication Agent failure.

Verifying that a Primary Database Object Is Marked for Replication

In a Sybase transaction replication system, both Replication Agent and Replication Server allow you to select the objects that you want to replicate. You do not need to replicate all objects or all data-changing operations in the primary database.

If a primary database object (such as a table or stored procedure) is not replicating, verify the object that you intended to replicate is marked.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine if the object is already marked:
   - For a table:
     ```
     pdb_setreptable pdb_table
     ```
     where *pdb_table* is the name of the primary database table that you want to verify is marked for replication.
   - For a LOB column:
     ```
     pdb_setrepcol pdb_table, pdb_col
     ```
     - where *pdb_table* is the marked table that contains the LOB column.
     - where *pdb_col* is the LOB column.
   - For a stored procedure:
     ```
     pdb_setrepproc pdb_proc
     ```
     where *pdb_proc* is the name of the stored procedure in the primary database that you want to verify is marked for replication.
   - For DDL:
     ```
     pdb_setrepddl object
     ```
     where object is a table, procedure, or sequence name. See the *Replication Agent Reference Manual* for details on using the **pdb_setrepddl** command.
   - For sequences:
     ```
     pdb_setrepseq pdb_seq
     ```
     where *pdb_seq* is the name of the sequence that you want to verify is marked for replication.

See also

- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128
- *Enabling and Disabling Replication for LOB Columns* on page 96
- *Marking and Unmarking Stored Procedures* on page 98
- *Enabling and Disabling DDL Replication* on page 94
- *Marking and Unmarking Oracle Sequences* on page 103
- *Marking a Table in the Primary Database for Replication* on page 91

Replication Agent Instance Availability Monitoring

You can configure Sybase Control Center (SCC) to monitor the availability of the Replication Agent instance and to notify you when the Replication Agent instance requires attention.

See *Sybase Control Center 3.2.6 > Sybase Control Center 3.2.6 for Replication > Manage and Monitor*.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128

Checking the Current Replication Agent Status

The status of the Replication Agent instance indicates whether it is in Replicating state or in Admin state.

No replication takes place when the Replication Agent instance is in Admin state.

1. Log in to the Replication Agent instance with the administrator login.
2. Get the current status of the Replication Agent instance:

   ```
   ra_status
   ```

   This command returns the current state of the Replication Agent instance and any current activity, for example:

   ```
   State  Action
   ------ ----------------------------
   ADMIN  Waiting for operator command
   (1 row affected)
   ```

See the *Replication Agent Reference Manual* for more information about the **ra_status** command.

When the Replication Agent instance is in one of these states, take the suggested actions.

State	Action
Admin	Start replication and put the Replication Agent instance in Replicating state by executing the Replication Agent **resume** command. If the Replication Agent instance returns to Admin state after you invoke the **resume** command, there is at least one unresolved problem that prevents the instance from going to Replicating state.
Replicating	No action is required. The instance is operating normally.
Replicating (Resynchronization)	No action is required. The instance is in the process of resynchronizing the primary and replicate databases. When resynchronization completes, the Replication Agent instance automatically transitions to the Replicating state.
Replication Down	Replication has stopped due to an error. Examine the Replication Agent error log to diagnose and fix the error. Then, resume the Replication Agent instance.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122

- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128
- *Replication Agent States* on page 65
- *Starting Replication* on page 70

Primary Database Log Access

Replication Agent may fail if it cannot read the primary database log.

An instance of Replication Agent for Oracle or Replication Agent for Microsoft SQL Server may fail to function properly on Windows if User Account Control (UAC) is enabled and the instance is not started with the `Run as administrator` option. This problem occurs when Replication Agent does not have permission to read the primary database log on Microsoft Windows Vista, Microsoft Windows Server 2008, Microsoft Windows Server 2008 R2, and Microsoft Windows 7. The Replication Agent system log file may contain messages indicating a problem accessing the primary database log:

- For Replication Agent for Oracle:

  ```
  java.io.FileNotFoundException:
  C:\SOMEPATH\REDO01.LOG (Access is denied)
  ```
- For Replication Agent for Microsoft SQL Server:

  ```
  Failed to synchronize log device with message
  C:\SOMEPATH\MSSQL\DATA\pdb_log.ldf (Access is denied)
  ```

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128

Replication Agent Logs

The Replication Agent system log files contain warning and error messages, as well as information about the Replication Agent connections to the primary database and the primary Replication Server.

Look for the most recent command you executed at the bottom of the log file to find the most recent message. The logs are located in the `$SYBASE/RAX-15_5/`*`inst_name`*`/log` directory, where *inst_name* is the name of the Replication Agent instance.

This is sample output from an Oracle instance log file:

```
T. 2012/11/16 10:23:13.482 LogMinerScanner
com.sybase.ds.oracle.logmnr.LogMinerScanManag Scanner
<LogMinerScanner> start scan locator =
0000000000145d8d00000001000000490000000000000000000145d8d00000000
I. 2012/11/16 10:23:13.482 INFORMATION
com.sybase.ds.oracle.log.OracleLogReader      LOB Cache initialized.
T. 2012/11/16 10:23:13.497 LogMinerScanner
com.sybase.ds.oracle.logmnr.LogMinerScanner   Scanner
<LogMinerScanner_1> start SCN = 1334669
I. 2012/11/16 10:23:14.200 INFORMATION
com.sybase.ra.ltm.LTM                        Replication Agent changed
to <REPLICATING> state.
I. 2012/11/16 10:23:14.356 INFORMATION
com.sybase.ra.conn.RAConnectionMgr          Attempting connection to
URL: jdbc:oracle:thin:@tomservo:1521:oradb112:
W. 2012/11/16 10:23:15.637 WARNING
com.sybase.ds.oracle.logmnr.txctx.DMLTransPro LogMiner <UNKNOWN>
operation has invalid operation content: OPID
<0x0000.00145e65.0000:0001.00000049.0000007a.0148>, Timestamp
<2012-11-16 10:22:00.0>, XID <0x0002.000c.00000382>, Serial# <531>,
Session# <146>, Session Info <UNKNOWN>, User Name <UNKNOWN>, Object
ID <0>, REL_FILE# <0>, DATA_BLK# <0>, DATA_OBJD#<0>, Operation <N/A>,
Op Code <54> Rollback <0> SSN <0> CSF <0> SQL Redo <N/A>
```

where:

- The first column displays a single character indicating the type of message:
  - I = information
  - W = warning
  - E = error
  - T = trace
  - S = severe
- The second column is a timestamp indicating when the message was written.
- The third column is a description.
- The fourth column identifies the Java class that produced the error.

Note: The fifth and sixth columns appear only when configuration property **log_trace_verbose** is set to true.

- The fifth column includes the method.
- The sixth column includes the line number.
- The final column is a text description of the message.

Note: In some cases, the information in a specific column is not consistent with these descriptions. In these cases, other information is generated that Technical Support uses to determine from where the message was generated.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128

Using ra_statistics to Check Replication Agent Operations and Performance

The **ra_statistics** command returns activity-related statistics that you can use to evaluate Replication Agent operations and performance.

By comparing the statistics returned when you first run the command to the statistics returned after you have successfully replicated something that you know works, you can analyze the differences in the statistics and troubleshoot where the problem lies. The statistics help you determine if the instance is:

- Scanning the transaction log
- Processing replicated transactions
- Sending LTL to the Replication Server

1. Log in to the Replication Agent instance with the administrator login.
2. Verify that you are in Replicating state. If you are not, change the state to Admin.
3. Get statistics for all of the Replication Agent components and the Java VM:

   ```
   ra_statistics
   ```
4. Save the statistics returned to use as a baseline for comparison.

5. Perform activity against the object that is not being replicated. For example, update a table that is not being replicated.
6. Repeat step 2.

   Note: Be sure to allow enough time for the Replication Agent to process the transaction.
7. Compare the newly returned statistics activity with the baseline.

   If the value returned for `Total Maintenance User operations filtered` increases, you are executing the transaction as the Replication Server maintenance user for this Replication Server connection. By default, these transactions are not sent to Replication Server. You must either connect to the primary database as a different user, or you can set the configuration value of **filter_maint_userid** to **false**.

For more information about the **ra_statistics** command, see the *Replication Agent Reference Manual*.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128
- *Connection Configuration Parameter Settings* on page 44

Available Memory

Diagnose and troubleshoot memory shortages.

If you are running out of memory, you see this error message:

```
java.lang.OutOfMemoryError
```

When you are running out of memory, either the Replication Agent drops out of Replicating state or the entire Replication Agent server stops executing.

To support adjusting the amount of memory available to the JRE, all of the executable scripts (or batch files) in the Replication Agent `bin` directory refer to an environment variable named RA_JAVA_MAX_MEM. All Replication Agent instance `RUN` scripts also reference the RA_JAVA_MAX_MEM environment variable.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118

- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128

Default Heap Size

Replication Agent does not set the RA_JAVA_MAX_MEM environment variable in the executable or `RUN` scripts, enabling the JVM to use the default maximum heap size.

In Java 6, the default maximum heap size is the larger of:

- One fourth of the physical memory
- 1GB

See the Java 6 documentation for information about the default initial and maximum heap sizes and for recommendations on how to set the defaults.

Default Maximum Memory Setting

The default maximum memory can be set in three ways.

You can set RA_JAVA_MAX_MEM:

- In the `RUN` script file
- In the `ra` script file.
- As an environment variable.

The setting of RA_JAVA_MAX_MEM in the `RUN` script file overrides the setting of the environment variable and the setting in the `ra` script file. If RA_JAVA_MAX_MEM is not set in the `RUN` script file, the setting of the environment variable overrides any value set in the `ra` script file. The setting for RA_JAVA_MAX_MEM in the `ra` script file is used only if RA_JAVA_MAX_MEM is set neither in the `RUN` script file nor in the environment variable.

You can override the JVM default for the maximum amount of memory available to the JRE for all instances of Replication Agent or for one specific instance of Replication Agent.

Overriding the Default for All Instances

Override the default maximum memory setting for all instances of Replication Agent.

1. Open the **ra** script file for editing.
    - For Windows, edit the `%SYBASE%\RAX-15_5\bin\ra.bat` file.

- For UNIX, edit the `$SYBASE/RAX-15_5/bin/ra.sh` file.

2. Uncomment lines in the **ra** script file.
   - For Windows, uncomment this line:
     ```
     set RA_JAVA_MAX_MEM=512m
     ```
   - For UNIX, uncomment these lines:
     ```
     RA_JAVA_MAX_MEM=512m
     export RA_JAVA_MAX_MEM
     ```
     Note: In UNIX, spaces are not allowed on either side of the equals sign.
3. If necessary, replace “512m” with a value appropriate for your replication environment.

   Note: The "m" here stands for megabytes.
4. Save the **ra** script file.

Overriding the Default for One Instance

Override the default maximum memory setting for one instance of Replication Agent.

1. Open the `RUN` script file for editing.
   - For Windows, edit the `%SYBASE%\RAX-15_5\`*<instance>*`\RUN_`*<instance>*`.bat` file.
   - For UNIX, edit the `$SYBASE/RAX-15_5/`*<instance>*`/RUN_`*<instance>*`.sh` file.

   Here, *<instance>* is the name of the Replication Agent instance.
2. Uncomment lines in the `RUN` script file.
   - For Windows, uncomment this line:
     ```
     set RA_JAVA_MAX_MEM=512m
     ```
   - For UNIX, uncomment these lines:
     ```
     RA_JAVA_MAX_MEM=512m
     export RA_JAVA_MAX_MEM
     ```
     Note: In UNIX, spaces are not allowed on either side of the equals sign.
3. If necessary, replace “512m” with a value appropriate for your replication environment.

   Note: The "m" here stands for megabytes.
4. Save the `RUN` script file.

Debugging LTL

LTL (Log Transfer Language) is the syntax used to communicate or distribute replication data to Replication Server.

LTL is the principal output from a Replication Agent.

To debug problematic LTL:

1. Log in to the running Replication Agent instance using the administrator login.
2. Verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

   If the Replication Agent instance is not in the Admin state, quiesce the Replication Agent instance:

   ```
   quiesce
   ```
3. Set the values of these Replication Agent configuration parameters for the primary database:

   ```
   trace LTITRACELTL, true
   ```
4. Start replication for the Replication Agent instance:

   ```
   resume
   ```
5. When new replication activity is generated, check the `LTITRACELTL.log` file in the log directory to debug your problem.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Skipped DDL Commands* on page 128
- *JVM Thread Dump* on page 128

Producing More Verbose LTL

By default, the LTL generated by the Replication Agent is compressed to reduce the amount of data sent to Replication Server. If you require more verbose output to help debug a problem, change these configuration parameters.

To output verbose LTL:

1. Log in to the running Replication Agent instance using the administrator login.
2. Verify that the Replication Agent instance is in Admin state:

   ```
   ra_status
   ```

   If the Replication Agent instance is not in the Admin state, quiesce the Replication Agent instance:

   ```
   quiesce
   ```

3. Set the values of these Replication Agent configuration parameters for the primary database:

```
ra_config column_compression, false
ra_config compress_ltl_syntax, false
ra_config structured_tokens, false
```

4. When new replication activity is generated, check the `LTITRACELTL.log` file in the log directory to debug your problem.

Skipped DDL Commands

If you move the truncation point to the end of the primary database transaction log with the **ra_locator move_truncpt** command, you have skipped over DDL commands that may have been read by Replication Agent to keep the RASD synchronized with the primary database.

To resolve a discrepancy between the primary database contents and the RASD contents, either:

- Refresh the RASD with the **ra_admin refresh** command, which ensures that the RASD is refreshed with current schema information from the primary database for all objects, or,
- If you know that only one object has been affected, unmark and then re-mark the object affected by the DDL.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *JVM Thread Dump* on page 128

JVM Thread Dump

A thread dump contains information about active threads in the Replication Agent JVM and can therefore be useful in troubleshooting a Replication Agent deadlock or performance issue.

Obtaining a thread dump on a Windows machine

In the command window in which you started Replication Agent, press Ctrl-Break.

Obtaining a thread dump on a UNIX or Linux machine

- Find the process ID of the JVM in which Replication Agent is running using **ps** and **grep**:

```
%>ps -ef | grep RAX-15_5
```

The **e** and **f** options show full output for every running process. The **grep** command filters results on the string “RAX-15_5”. The result returned looks something like this:

```
sybase 12345 67890 0 14:21 pts/1 00:00:22
/software/sybase/RAX-15_5/JRE6/bin/java -server
```

where 12345 is the process ID (PID) of the Replication Agent Java process.

- Terminate the process with the **kill** command and **QUIT** option:

```
%>kill -QUIT pid
```

where *pid* is the PID you obtained from the last step.

On UNIX or Linux, it is customary to start the Replication Agent process in the background, so the console in which the Replication Agent process was started may no longer be available when you decide to obtain a thread dump. The JVM always prints a thread dump to the console in which the Replication Agent process was started, so you should always redirect process output to a file when you start Replication Agent. For example:

```
%>RUN_inst_name >! output.txt
```

where *inst_name* is the server name of the Replication Agent instance.

Note: On AIX, the process output contains a path to the javacore file containing the thread dump.

See also

- *Verifying that a Primary Database Object Is Marked for Replication* on page 118
- *Replication Agent Instance Availability Monitoring* on page 119
- *Checking the Current Replication Agent Status* on page 119
- *Primary Database Log Access* on page 121
- *Replication Agent Logs* on page 122
- *Using ra_statistics to Check Replication Agent Operations and Performance* on page 123
- *Available Memory* on page 124
- *Debugging LTL* on page 126
- *Skipped DDL Commands* on page 128

Replication Server Troubleshooting Issues

Use Replication Server commands to check for the most common replication problems.

For more detailed information about diagnosing and solving Replication Server problems, see the *Replication Server Troubleshooting Guide*.

Replication Definitions and Subscriptions Check

Verify that you created replication definitions with the appropriate information and that you have defined and activated subscriptions for all replication definitions.

Checking the Status and Operation of the Replication Server

Replication Server provides several **admin** commands you can use to check its status and operation.

1. Log in to the Replication Server with a user login that has "sa" permission.
2. Check the current status of the Replication Server:

   ```
   admin health
   ```

   This command returns the current status of the Replication Server, as shown in this example:

   ```
   Mode           Quiesce     Status
   -------        -------     ------
   NORMAL         FALSE       HEALTHY
   ```

   If the Replication Server status is `SUSPECT`, use the **admin who_is_down** command to check for Replication Server threads that may be down or attempting to connect to other servers.
3. Check the current status of the Replication Server primary database connection (the connection from the Replication Agent to the primary Replication Server):

   ```
   admin show_connections
   ```

   You can also use the **admin who, dsi** command to get more information about the Replication Agent connection in the primary Replication Server.

   Note: Use the **admin show_connections** or **admin who, dsi** command output to verify that the primary data server and primary database names are correct for the Replication Agent connection in the primary Replication Server.

See the *Replication Server Reference Manual* for more information about the **admin show_connections** and **admin who** commands.

Verifying that rs_username Login has Appropriate Permissions

Verify the Replication Agent login in Replication Server.

The Replication Server **connect source lti** command:

- Verifies that the Replication Server database connection used by the Replication Agent exists in the primary Replication Server

- Verifies that the login name specified in the Replication Agent **rs_username** parameter has permission to connect to the primary Replication Server as a data source
- Returns a version string that shows the highest numbered version of LTL that the primary Replication Server supports

Refer to the *Replication Agent Installation Guide > Installation and Setup Worksheet* for the parameter values required for this task.

1. Log in to the primary Replication Server with the Replication Agent user login name specified in the **rs_username** configuration parameter.
2. Execute:

   ```
   connect source lti pds.pdb version
   ```

   where:
   - *pds* is the value specified for the Replication Agent **rs_source_ds** configuration parameter.
   - *pdb* is the value specified for the Replication Agent **rs_source_db** configuration parameter.
   - *version* is the proposed LTL version number.

   Enter **999** for the value of the LTL version number. Replication Server returns the highest numbered version of LTL that it supports.
3. Disconnect from the primary Replication Server as **rs_username**, and then log in to the Replication Agent instance with the administrator login and invoke the **resume** command.

For more information about the **connect source lti** command, see the *Replication Server Design Guide* and *Replication Agent Reference Manual*.

Verifying Stable Queues

Check the Replication Server stable queues to determine which transactions are being processed or ignored, and to determine whether transactions are open (not committed).

To display information about SQM and SQT threads:

1. Log in to the primary Replication Server and execute the **admin who, sqm** command.
2. View the results to determine the number of duplicate messages being detected and ignored, and the number of blocks being written in the Replication Server stable queues.
3. In the primary Replication Server, execute the **admin who, sql** command.
4. View the results to find open transactions.

See the *Replication Server Reference Manual* for more information about the **admin who** command.

Performance Monitoring

You can monitor the performance of Replication Server using the **rs_ticket** and **rs_ticket_report** Replication Server stored procedures, which respectively reside at the primary and replicate databases. Replication Agent provides support for these stored procedures through the Replication Agent **rs_ticket** command.

For detailed information about the **rs_ticket** and **rs_ticket_report** Replication Server stored procedures, see the *Replication Server Reference Manual*. For information about the **rs_ticket** Replication Agent command, see the *Replication Agent Reference Manual*.

Materializing Subscriptions to Primary Data

Learn about bulk materialization, the process of materializing subscriptions to primary tables in a non-Sybase database, and how to use bulk materialization to set up replication from primary tables in a primary database.

Materialization

Materialization is the creation and activation of subscriptions and the copying of data from the primary database to the replicate database, thereby initializing the replicate database.

Before you can replicate data from a primary database, you must set up and populate each replicate database so that it is in a state consistent with that of the primary database.

There are two types of subscription materialization supported by the Sybase Replication Server:

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

Note: Replication Agent does not support atomic materialization.

See the *Replication Server Administration Guide* for more information on subscription materialization methods.

Bulk Materialization

Sybase recommends that you use bulk materialization to materialize subscriptions to primary data in a non-Sybase database.

To use bulk materialization:

- Define, activate, and validate the subscription (or create the subscription without materialization)
- Unload the subscription data at the primary site
- Move the unloaded data to the replicate database
- Load data into the replicate tables
- Resume the database connection from the replicate Replication Server to the replicate database so that the replicate database can receive replicated transactions
- Resume replication at the Replication Agent instance

There are two bulk materialization options for subscriptions to primary data in a non-Sybase database:

- Atomic bulk materialization
- Nonatomic bulk materialization

Data Unloading and Loading

Part of subscription materialization is unloading subscription data from the primary table so it can be loaded into the replicate table.

Subscription data is the data in the primary table that is requested by the subscription.

Unload Utilities

Data-unloading utilities are usually provided with the primary data server software. You can use one of the OEM-supplied unloading utilities or a database unload utility of your choice.

Load Utilities

If you are using Adaptive Server Enterprise as the data server for your replicate database, you can use **bcp** to load subscription data into the replicate database. If you are using a non-Sybase data server as the data server for your replicate database, you can use the load utility of your choice to load subscription data into the replicate database.

See *Sybase Adaptive Server Enterprise Utility Programs* for more information about using **bcp** to load subscription data into a replicate database in Adaptive Server Enterprise.

Atomic Bulk Materialization

Atomic bulk materialization assumes that all applications updating the primary table can be suspended while a copy of the table is made. This copy is loaded at the replicate site.

You can use this method to retrieve data from the primary database if you can suspend updates to the primary data.

Prepare for Atomic Bulk Materialization

Verify that you are ready for atomic bulk materialization.

Before you start an atomic bulk materialization procedure, verify that:

- The primary table exists and contains data.
- You have a user ID with ownership or **select** privilege on the primary table (or a column to be replicated in the primary table).
- The replicate table exists and contains the appropriate columns.
- You successfully configured every Replication Server in your replication system.

- You created the replication definition correctly at the primary Replication Server.
- You successfully created the Replication Agent transaction log in the primary database.
- You marked and enabled replication for the primary table in the primary database.
- You started the Replication Agent instance and put it in the Replicating state.

Performing Atomic Bulk Materialization

Perform atomic bulk materialization.

1. Log in to the replicate Replication Server as the system administrator (**sa**):

   ```
   isql -Usa -Psa_password -SRRS_servername
   ```

   where:
   - *sa* is the system administrator user ID.
   - *sa_password* is the password for the system administrator user ID.
   - *RRS_servername* is the name of the replicate Replication Server.
2. Define the subscription at the replicate Replication Server:

   ```
   1> define subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> [where search_conditions]
   5> go
   ```

   The *dataserver.database* name must match the name you used for your replicate database.
3. Check the subscription at both the primary and replicate Replication Servers. Use this command to verify that the subscription status is DEFINED:

   ```
   1> check subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> go
   ```
4. Lock the primary table to prevent primary transaction activity. This prevents updates to the primary table during materialization.
5. Unload the subscription data at the primary site using your preferred database unload method to select or dump the data from the primary table.

   Note: When unloading subscription data from the primary table, make sure you select only the columns specified in the replication definition and the rows specified in the subscription.
6. Activate the subscription using the **with suspension** option at the replicate Replication Server:

   ```
   1> activate subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> with suspension
   5> go
   ```

7. Wait for the subscription to become active at both the primary and replicate Replication Servers. Execute **check subscription** at both the primary and replicate Replication Servers to verify that the subscription status is ACTIVE.

   When the subscription status is ACTIVE at the replicate Replication Server, the database connection for the replicate database is suspended.
8. Restore the primary table to read-write access (unlock).
9. Load the subscription data into the replicate database using **bcp** or the preferred database load utility for your site.

   Note: Before loading the subscription data into the replicate table, make sure that any data manipulation to be performed by Replication Agent (such as *datetime* conversion) or by Replication Server function strings is applied to the unload file.
10. From the replicate Replication Server, resume the database connection for the replicate database:

```
1> resume connection
2> to dataserver.database
3> go
```

11. Validate the subscription at the replicate Replication Server:

```
1> validate subscription subscription_name
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

12. Wait for the subscription to become valid at both the primary and replicate Replication Servers, then execute **check subscription** at both the primary and replicate Replication Servers to verify that the status is **VALID.**

When you complete this procedure, the subscription is created, the replicate data is consistent with the primary data, and replication is in progress.

See also

- *Troubleshooting* on page 117

Nonatomic Bulk Materialization

Nonatomic bulk materialization assumes applications updating the primary table cannot be suspended while a copy of the table is made. Therefore, a nonatomic materialization requires the use of the Replication Server autocorrection feature to get the replicate database synchronized with the primary database.

Note: You cannot use nonatomic materialization if the **replicate minimal columns** feature is set for the replication definition for the primary table.

Prepare for Nonatomic Bulk Materialization

Perform some tasks to prepare for nonatomic bulk materialization.

Before you start a nonatomic bulk materialization procedure, verify that:

- The primary table exists and contains data.
- You have a user ID with ownership or **select** privilege on the primary table (or a column to be replicated in the primary table).
- The replicate table exists and contains the appropriate columns.
- You successfully configured every Replication Server in your replication system.
- You created the replication definition correctly at the primary Replication Server.
- You successfully created the Replication Agent transaction log in the primary database.
- You marked and enabled replication for the primary table in the primary database.
- You started the Replication Agent instance and put it in the Replicating state.

Performing Nonatomic Bulk Materialization

Perform nonatomic bulk materialization.

1. Log in to the replicate Replication Server as the system administrator (**sa**):

   ```
   isql -Usa -Psa_password -SRRS_servername
   ```

   where:
   - *sa* is the system administrator user ID.
   - *sa_password* is the password for the system administrator user ID.
   - *RRS_servername* is the name of the replicate Replication Server.
2. At the replicate Replication Server, turn on the autocorrection feature:

   ```
   1> set autocorrection on
   2> for replication_definition
   3> with replicate at dataserver.database
   4> go
   ```
3. At the replicate Replication Server, define the subscription:

   ```
   1> define subscription subscription_name
   2> for replication_definition
   3> with replicate at dataserver.database
   4> with suspension
   5> go
   ```

   The *dataserver.database* name must match the name you used for your replicate database.
4. In the primary database, invoke **rs_marker** to activate the subscription.
5. Check the subscription at both the primary and replicate Replication Servers. Use this command to verify that the subscription status is **ACTIVE**:

   ```
   1> check subscription subscription_name
   2> for replication_definition
   ```

```
3> with replicate at dataserver.database
4> go
```

When the subscription status is **ACTIVE** at the replicate Replication Server, the database connection for the replicate database is suspended.

6. Unload the subscription data at the primary site using the preferred database unload method for your site to select or dump the data from the primary tables.

Note: When unloading subscription data from the primary table, make sure you select only the columns specified in the replication definition and the rows specified in the subscription.

7. In the primary database, invoke **rs_marker** to validate the subscription.
8. Wait for the subscription to become valid at both the primary and replicate Replication Servers, then execute **check subscription** at both the primary and replicate Replication Servers to verify that the status is **VALID.**
9. Load the subscription data into the replicate database using **bcp** or the preferred database load utility for your site.

Note: Before loading the subscription data into the replicate table, make sure that any data manipulation that is to be performed by Replication Agent (such as *datetime* conversion) or by Replication Server function strings is applied to the unload file.

10. From the replicate Replication Server, resume the database connection for the replicate database:

```
1> resume connection
2> to dataserver.database
3> go
```

11. Wait for the subscription to become valid at both the primary and replicate Replication Servers, then execute **check subscription** at both the primary and replicate Replication Servers to verify that the status is **VALID.**

    When the subscription status is **VALID** at the replicate Replication Server, the replicate database is synchronized with the primary database and you can turn off autocorrection.

12. Turn off the autocorrection feature at the replicate Replication Server:

```
1> set autocorrection off
2> for replication_definition
3> with replicate at dataserver.database
4> go
```

When you complete this procedure, the subscription is created, the replicate data is consistent with the primary data, and replication is in progress.

See the *Replication Server Commands Reference* for information about Replication Command Language (RCL) commands. See the *Replication Server Administration Guide* for information about configuring Replication Servers and materialization methods.

See also

- *Troubleshooting* on page 117

Glossary

This glossary describes Replication Server Options terms.

- **Adaptive Server® –** the brand name for Sybase relational database management system (RDBMS) software products.

    - Adaptive Server® Enterprise manages multiple, large relational databases for high-volume online transaction processing (OLTP) systems and client applications.
    - Sybase®IQ manages multiple, large relational databases with special indexing algorithms to support high-speed, high-volume business intelligence, decision support, and reporting client applications.
    - SQL Anywhere® (formerly Adaptive Server Anywhere) manages relational databases with a small DBMS footprint, which is ideal for embedded applications and mobile device applications.

  See also *DBMS* and *RDBMS*.
- **atomic materialization –** a materialization method that copies subscription data from a primary database to a replicate database in a single, atomic operation. No changes to primary data are allowed until the subscription data is captured at the primary database. See also *bulk materialization* and *nonatomic materialization*.
- **BCP utility –** a bulk copy transfer utility that provides the ability to load multiple rows of data into a table in a target database. See also *bulk copy*.
- **bulk copy –** an Open Client™ interface for the high-speed transfer of data between a database table and program variables. Bulk copying provides an alternative to using SQL **insert** and **select** commands to transfer data.
- **bulk materialization –** a materialization method whereby subscription data in a replicate database is initialized outside of the replication system. You can use bulk materialization for subscriptions to table replication definitions or function replication definitions. See also *atomic materialization* and *nonatomic materialization*.
- **client –** in client/server systems, the part of the system that sends requests to servers and processes the results of those requests. See also *client application*.
- **client application –** software that is responsible for the user interface, including menus, data entry screens, and report formats. See also *client*.
- **commit –** an instruction to the DBMS to make permanent the changes requested in a transaction. See also *transaction*. Contrast with *rollback*.
- **database –** a collection of data with a specific structure (or schema) for accepting, storing, and providing data for users. See also *data server*, *DBMS*, and *RDBMS*.
- **database connection –** a connection that allows Replication Server to manage the database and distribute transactions to the database. Each database in a replication system

can have only one database connection in Replication Server. See also *Replication Server* and *route*.

- **data client –** a client application that provides access to data by connecting to a data server. See also *client*, *client application*, and *data server*.
- **data distribution –** a method of locating (or placing) discrete parts of a single set of data in multiple systems or at multiple sites. Data distribution is distinct from data replication, although a data replication system can be used to implement or support data distribution. Contrast with *data replication*.
- **data replication –** the process of copying primary data to remote locations and synchronizing the copied data with the primary data. Data replication is different from data distribution. Replicated data is a stored copy of data at one or more remote sites throughout a system, and it is not necessarily distributed data. Contrast with *data distribution*. See also *transaction replication*.
- **data server –** a server that provides the functionality necessary to maintain the physical representation of a table in a database. Data servers are usually database servers, but they can also be any data repository with the interface and functionality a data client requires. See also *client*, *client application*, and *data client*.
- **datatype –** a keyword that identifies the characteristics of stored information on a computer. Some common datatypes are: *char*, *int*, *smallint*, *date*, *time*, *numeric*, and *float*. Different data servers support different datatypes.
- **DBMS –** an abbreviation for database management system, a computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The DBMS can include the user interface for using the database, or it can be a standalone data server system. Compare with *RDBMS*.
- **ERSSD –** an abbreviation for Embedded Replication Server System Database, which manages replication system information for a Replication Server. See also *Replication Server*.
- **failback –** a procedure that restores the normal user and client access to a primary database, after a failover procedure switches access from the primary database to a replicate database. See also *failover*.
- **failover –** a procedure that switches user and client access from a primary database to a replicate database, particularly in the event of a failure that interrupts operations at the primary database, or access to the primary database. Failover is an important fault-tolerance feature for systems that require high availability. See also *failback*.
- **function –** a data server object that represents an operation or set of operations. Replication Server distributes operations to replicate databases as functions. See also *stored procedure*.
- **function string –** a string that Replication Server uses to map a function and its parameters to a data server API. Function strings allow Replication Server to support heterogeneous replication, in which the primary and replicate databases are different types, with different SQL extensions and different command features. See also *function*.

- **gateway –** connectivity software that allows two or more computer systems with different network architectures to communicate.
- **inbound queue –** a stable queue managed by Replication Server to spool messages received from a Replication Agent. See also *outbound queue* and *stable queue*.
- **interfaces file –** a file containing information that Sybase Open Client and Open Server™ applications need to establish connections to other Open Client and Open Server applications. See also *Open Client* and *Open Server*.
- **isql –** an Interactive SQL client application that can connect and communicate with any Sybase Open Server application, including Adaptive Server, Replication Agent, and Replication Server. See also *Open Client* and *Open Server*.
- **Java –** an object-oriented programming language developed by Sun Microsystems. A platform-independent, “write once, run anywhere” programming language.
- **Java VM –** the Java Virtual Machine. The Java VM (or JVM) is the part of the Java Runtime Environment (JRE) that is responsible for interpreting Java byte codes. See also *Java* and *JRE*.
- **JDBC –** an abbreviation for Java Database Connectivity. JDBC is the standard communication protocol for connectivity between Java clients and data servers. See also *data server* and *Java*.
- **JRE –** an abbreviation for Java Runtime Environment. The JRE consists of the Java Virtual Machine (Java VM or JVM), the Java Core Classes, and supporting files. The JRE must be installed on a machine to run Java applications, such as Replication Agent. See also *Java VM*.
- **LAN –** an abbreviation for “local area network,” a computer network located on the user premises and covering a limited geographical area (usually a single site). Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary can be subject to some form of regulation. Contrast with *WAN*.
- **latency –** in transaction replication, the time it takes to replicate a transaction from a primary database to a replicate database. Specifically, latency is the time elapsed between committing an original transaction in the primary database and committing the replicated transaction in the replicate database.

  In disk replication, latency is the time elapsed between a disk write operation that changes a block or page on a primary device and the disk write operation that changes the replicated block or page on a replicate device.

  See also *transaction replication*.
- **LOB –** an abbreviation for large object, a large collection of data stored as a single entity in a database.
- **Log Reader –** an internal component of Replication Agent that interacts with the primary database to capture transactions for replication. See also *Log Transfer Interface* and *Log Transfer Manager*.

- **Log Transfer Interface –** an internal component of Replication Agent that interacts with Replication Server to forward transactions for distribution to Replication Server. See also *Log Reader* and *Log Transfer Manager*.
- **Log Transfer Language –** the proprietary protocol used between Replication Agent and Replication Server to replicate data from the primary database to Replication Server. See also *Log Reader* and *Log Transfer Interface*.
- **Log Transfer Manager –** an internal component of Replication Agent that interacts with the other Replication Agent internal components to control and coordinate Replication Agent operations. See also *Log Reader* and *Log Transfer Interface*.
- **maintenance user –** a special user login name in the replicate database that Replication Server uses to apply replicated transactions to the database. See also *replicate database* and *Replication Server*.
- **materialization –** the process of copying the data from a primary database to a replicate database, initializing the replicate database so that the replication system can begin replicating transactions. See also *atomic materialization*, *bulk materialization*, and *nonatomic materialization*.
- **Multi-Path Replication™ –** Replication Server feature that improves performance by enabling parallel paths of data from the source database to the target database. These multiple paths process data independently and are applicable when sets of data can be processed in parallel without transactional consistency requirements between them.
- **nonatomic materialization –** a materialization method that copies subscription data without a lock on the primary database. Changes to primary data are allowed during data transfer, which may cause temporary inconsistencies between the primary and replicate databases. Contrast with *atomic materialization*. See also *bulk materialization*.
- **ODBC –** an abbreviation for Open Database Connectivity, an industry-standard communication protocol for clients connecting to data servers. See also *client*, *data server*, and *JDBC*.
- **Open Client –** a Sybase product that provides customer applications, third-party products, and other Sybase products with the interfaces needed to communicate with Open Server applications. See also *Open Server*.
- **Open Client application –** An application that uses Sybase Open Client libraries to implement Open Client communication protocols. See also *Open Client* and *Open Server*.
- **Open Server –** a Sybase product that provides the tools and interfaces required to create a custom server. See also *Open Client*.
- **Open Server application –** a server application that uses Sybase Open Server libraries to implement Open Server communication protocols. See also *Open Client* and *Open Server*.
- **outbound queue –** a stable queue managed by Replication Server to spool messages to a replicate database. See also *inbound queue*, *replicate database*, and *stable queue*.
- **primary data –** the data source used for replication. Primary data is stored and managed by the primary database. See also *primary database*.

- **primary database –** the database that contains the data to be replicated to another database (the replicate database) through a replication system. The primary database is the source of replicated data in a replication system. Sometimes called the active database. Contrast with *replicate database*. See also *primary data*.
- **primary key –** a column or set of columns that uniquely identifies each row in a table.
- **primary site –** the location or facility at which primary data servers and primary databases are deployed to support normal business operations. Sometimes called the active site or main site. See also *primary database* and *replicate site*.
- **primary table –** a table used as a source for replication. Primary tables are defined in the primary database schema. See also *primary data* and *primary database*.
- **primary transaction –** a transaction that is committed in the primary database and recorded in the primary database transaction log. See also *primary database*, *replicated transaction*, and *transaction log*.
- **quiesce –** to cause a system to go into a state in which further data changes are not allowed. See also *quiescent*.
- **quiescent –** in a replication system, a state in which all updates have been propagated to their destinations. Some Replication Agent and Replication Server commands require that you first quiesce the replication system.

  In a database, a state in which all data updates are suspended so that transactions cannot change any data, and the data and log devices are stable.

  This term is interchangeable with quiesced and in quiesce. See also *quiesce*.
- **RASD –** an abbreviation for Replication Agent System Database. Information in the RASD is used by the primary database to recognize database structure or schema objects in the transaction log.
- **RCL –** an abbreviation for Replication Command Language, the command language used to manage Replication Server. See also *Replication Server*.
- **RDBMS –** an abbreviation for relational database management system, an application that manages and controls relational databases. Compare with *DBMS*. See also *relational database*.
- **relational database –** a collection of data in which data is viewed as being stored in tables, which consist of columns (data items) and rows (units of information). Relational databases can be accessed by SQL requests. Compare with *database*. See also *SQL*.
- **replicate data –** A set of data that is replicated from a primary database to a replicate database by a replication system. See also *primary database*, *replication system*, and *replicate database*.
- **replicate database –** a database that contains data replicated from another database (the primary database) through a replication system. The replicate database is the database that receives replicated data in a replication system. Contrast with *primary database*. See also *replicate data*, *replicated transaction*, and *replication system*.

- **replicated transaction –** a primary transaction that is replicated from a primary database to a replicate database by a transaction replication system. See also *primary database*, *primary transaction*, *replicate database*, and *transaction replication*.
- **replicate site –** the location or facility at which replicate data servers and replicate databases are deployed to support normal business operations during scheduled downtime at the primary site. Contrast with *primary site*. See also *replicate database*.
- **Replication Agent –** an application that reads a primary database transaction log to acquire information about data-changing transactions in the primary database, processes the log information, and then sends it to a Replication Server for distribution to a replicate database. See also *primary database* and *Replication Server*.
- **replication definition –** a description of a table or stored procedure in a primary database, for which subscriptions can be created. The replication definition, maintained by Replication Server, includes information about the columns to be replicated and the location of the primary table or stored procedure. See also *Replication Server* and *subscription*.
- **Replication Server –** a Sybase software product that provides the infrastructure for a transaction replication system. See also *Replication Agent*.
- **replication system –** a data processing system that replicates data from one location to another. Data can be replicated between separate systems at a single site, or from one or more local systems to one or more remote systems. See also *transaction replication*.
- **rollback –** an instruction to a database to back out of the changes requested in a unit of work (called a transaction). Contrast with *commit*. See also *transaction*.
- **route –** A one-way message stream from a primary Replication Server to a replicate Replication Server. Routes carry data-changing commands (including those for RSSDs) and replicated functions (database procedures) between separate Replication Servers. See also *Replication Server*.
- **RSSD –** an abbreviation for Replication Server System Database, which manages replication system information for a Replication Server. See also *Replication Server*.
- **SQL –** an abbreviation for Structured Query Language, a nonprocedural programming language used to process data in a relational database. ANSI SQL is an industry standard. See also *transaction*.
- **stable queue –** a disk device-based, store-and-forward queue managed by Replication Server. Messages written into the stable queue remain there until they can be delivered to the appropriate process or replicate database. Replication Server provides a stable queue for both incoming messages (the inbound queue) and outgoing messages (the outbound queue). See also *database connection*, *Replication Server*, and *route*.
- **stored procedure –** a data server object that represents an operation or set of operations. This term is often used interchangeably with *function*.
- **subscription –** a request for Replication Server to maintain a replicated copy of a table, or a set of rows from a table, in a replicate database at a specified location. See also *replicate database*, *replication definition*, and *Replication Server*.

- **table –** in a relational DBMS, a two-dimensional array of data or a named data object that contains a specific number of unordered rows composed of a group of columns that are specific for the table. See also *database*.
- **transaction –** a unit of work in a database that can include zero, one, or many operations (including **insert**, **update**, and **delete** operations), and that is either applied or rejected as a whole. Each SQL statement that modifies data can be treated as a separate transaction, if the database is so configured. See also *SQL*.
- **transactional consistency –** A condition in which all transactions in the primary database are applied in the replicate database, and in the same order that they were applied in the primary database.
- **transaction log –** generally, the log of transactions that affect the data managed by a data server. Replication Agent reads the transaction log to identify and acquire the transactions to be replicated from the primary database. See also *Replication Agent*, *primary database*, and *Replication Server*.
- **transaction replication –** a data replication method that copies data-changing operations from a primary database to a replicate database. See also *data replication*.
- **UDB –** IBM DB2 Universal Database (formerly IBM DB2 for Linux, UNIX, and Windows).
- **WAN –** an abbreviation for "wide area network," a system of local-area networks (LANs) connected together with data communication lines. Contrast with *LAN*.

Index