SYBASE®
An SAP® Company

New Features Guide

Replication Server® 15.7.1

DOCUMENT ID: DC00783-01-1571-01
LAST REVISED: April 2012

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 san 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 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, IBM DB2 UDB, and Microsoft SQL Server. 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 UDB – for IBM DB2 on Linux, Unix, and Windows

New Features in Replication Server 15.7.1

Replication Server® 15.7.1 includes performance, usability, security, and database support enhancements.

Enhancements to Adaptive Server Replication Support

Replication Server 15.7.1 supports Adaptive Server replication.

Adaptive Server Data Compression

Replication Server supports the Adaptive Server data compression feature.

With Adaptive Server version 15.7, you can use data compression, which lets you use less storage space for the same amount of data, reduce cache memory consumption, and improve performance because of lower I/O demands. Adaptive Server can compress regular data and large object (LOB) datatypes such as `text`, `image`, and `unitext`. See the *Adaptive Server Enterprise Compression Users Guide*.

Adaptive Server stores data in-row or off-row. Adaptive Server stores in-row data in a location that is physically contiguous to the row metadata. Adaptive Server stores LOB data off-row in other locations because of the size of the data. There is a pointer in-row to the actual location of off-row data.

Replication Server does not perform any decompression and replicates the compressed LOB columns from the primary Adaptive Server database, in compressed format, and without decompressing the text values. See *Replication Server Administration Guide Volume 1 > Manage RepAgent and Support Adaptive Server > Adaptive Server Data Compression* for support for compressed data replication between Adaptive Server databases.

Version Support

- Adaptive Server – version 15.7 ESD #1 and later for both primary and replicate databases. See *Replication Server Release Bulletin > Product Compatibility > Replication Server Interoperability* for Adaptive Server versions compatible with Replication Server 15.7.1.
- Replication Server – version 15.7.1 and later for the primary and replicate Replication Server

In-row Off-Row LOB

Replication Server supports the changes in in-row off-row LOB support in Adaptive Server 15.7 and later.

See *Adaptive Server Enterprise > New Features Summary > New Features in Adaptive Server Version 15.7 > Changes for Large Objects > In-Row Off-Row LOB*.

Master Key and rs password

Set the master key password and **rs password** attributes to continue replication.

In Adaptive Server, when you create the syb_extpasswdkey service key with the master key and you have not set the master key password in memory manually or automatically, the Adaptive Server RepAgent is blocked at startup and **sp_who** shows "MASTER KEY SLEEP" until you set the master key password. Each replication path has one **rs password** attribute that RepAgent uses to log in to Replication Server. When you drop the syb_extpasswdkey service key, Adaptive Server resets all the existing RepAgent **rs password** attributes. If you enter **sp_encryption helpextpasswd**, you see "Needs Reset". You must reset all the **rs password** attributes to continue replication.

See *Adaptive Server Enterprise > Encrypted Columns Users Guide > Securing External Passwords and Hidden Text > Service Keys*.

Password Expiration Intervals for Master Database Replication

If you set up Adaptive Server master database replication in a warm standby environment, Sybase recommends setting longer password expiration intervals on the standby master database compared to the expiration intervals on the active master database. This allows the active master database to control any change of passwords and allows replication of password changes to proceed.

See *Replication Server > Administration Guide Volume 2 > Manage Warm Standby Applications > Replication of the Master Database in a Warm Standby Environment for ASE*.

Support for Adaptive Server Commands and System Procedures

Support is extended for the replication of several Adaptive Server commands and system procedures.

DDL commands and system procedures supported for replication:

- **alter login**
- **alter login profile**
- **alter...modify owner** – Replication Server treats tables with different owners as different tables. You must make the relevant change to the table replication definitions if you use **alter...modify owner** to change the owner for an Adaptive Server replicated table. See *Replication Server Administration Guide Volume 1 > Manage Replicated Tables >*

Modify Replication Definitions > Alter Replication Definitions > Changes You Can Make to the Replication Definition > Changing Table Owner.

- **create login**
- **create login profile**
- **drop login**
- **drop login profile**
- **sp_hidetext**

System procedures supported for master database replication:

- **sp_addexternlogin**
- **sp_dropexternlogin**
- **sp_maplogin**
- **sp_addremotelogin**
- **sp_dropremotelogin**
- **sp_addserver**
- **sp_dropserver**

See *Replication Server > Reference Manual > Adaptive Server Commands and System Procedures >* ***sp_reptostandby*** *> Supported DDL Commands and System Procedures*.

Multi-Path Replication

Replication Server 15.7.1 includes enhancements to Multi-Path Replication™ support.

Heterogeneous Multi-Path Replication

Replication Server 15.7.1 extends supports for Multi-Path Replication to replication systems with heterogeneous databases.

As of Replication Server 15.7, you can distribute transactions over multiple replication paths from a primary Adaptive Server database to Replication Server, through dedicated routes between Replication Servers and over multiple connections to a replicate Adaptive Server database. See *Replication Server > Administration Guide Volume 2 > Performance Tuning > Multi-Path Replication*.

In version 15.7.1, you can build multiple replication paths and dedicated routes between databases such as Adaptive Server and Sybase IQ, and Adaptive Server and Oracle.

See *Replication Server Heterogeneous Replication Guide > Sybase IQ as Replicate Data Server > Multi-Path Replication to Sybase IQ* to configue multipath replication to Sybase IQ databases. See *Replication Server Heterogeneous Replication Guide > Heterogeneous Multi-Path Replication* for heterogeneous multipath replication scenarios.

Supported Heterogeneous Database Multipath Replication Systems

Table 1. Primary and Replicate Database Pairs Supported in Heterogeneous Multipath Replication Systems

Primary Database	Replicate Database
Adaptive Server	Sybase IQ
Oracle	Sybase IQ
Adaptive Server	Oracle
Oracle	Adaptive Server
Oracle	Oracle

Note: Replication from an Oracle primary database requires the Replication Agent for Oracle which is included in Replication Server Options.

Table 2. Database Versions Supported for Multipath Replication

Database	Supported Versions
Adaptive Server	15.7 and later
Oracle	Oracle 10g and 11g. See *Replication Server Options > Replication Server Options Release Bulletin > Product Summary > Product Compatibility*.
Sybase IQ	15.1 and later. See *Replication Server Release Bulletin > Product Compatibility > Replication Server Interoperability* .

License

Multi-Path Replication is licensed as part of the Advanced Services Option. Replication to Sybase IQ using RTL is available in the Real-Time Loading Edition (RTLE). See *Replication Server Installation Guide > Planning Your Installation > Obtaining a License*.

Replication from Oracle requires Replication Agent for Oracle. See *Replication Server Options > Replication Agent Installation Guide > Planning Your Installation > Licensing*.

Distribution by Connection

In a Multi-Path Replication environment, you can use different distribution modes to achieve parallel replication and improved replication performance by distributing the replication load

from the primary database through the available primary replication paths originating from the database.

With Replication Server 15.7 and Adaptive Server 15.7, you can bind objects, such as tables and stored procedures, to specific replication paths to enable the replication of these objects in parallel. See *Replication Server Administration Guide Volume 2 > Performance Tuning > Multi-Path Replication > Multiple Primary Replication Paths > Binding Objects to a Replication Path*.

With Replication Server 15.7.1 and Adaptive Server 15.7 ESD #1, you can distribute the replication load by connection. The default mode is distribution by object binding. The Adaptive Server RepAgent does not support more than one distribution mode at a time.

In distribution by connection, the Adaptive Server RepAgent assigns transactions originated by different client processes to the available replication paths. Over time, data distribution balances across all available paths. Performance improves and replication load distribution is more uniform if there are many RepAgent paths available and the number of client processes is large.

See *Replication Server Administration Guide: Volume 2 > Performance Tuning > Multi-Path Replication > Paraallel Transaction Streams > Distrubution Modes for Multi-Path Replication > Distribution by Connection*.

Database Support

Replication Server supports distribution by connection for multipath replication between primary and replicate Adaptive Server databases:

- Primary database – Adaptive Server 15.7 ESD #1 and later.
- Replicate database – Adaptive Server 15.7 and later

See *Replication Server Release Bulletin > Product Compatibility > Replication Server Interoperability* .

License

Multi-Path Replication is licensed as part of the Advanced Services Option. See *Replication Server Installation Guide > Planning Your Installation > Obtaining a License*.

Changes to Adaptive Server Monitoring Tables

If you select to distribute the replication load by connection, use the fields in the `monRepSenders` Adaptive Server monitoring table to provide a statistical snapshot of data distibution, and analyze Adaptive Server performance.

Table 3. monRepSenders

Field	Description
`NumberOfCommandsProcessed`	Number of commands, such as **insert**, **delete**, **begin trans**, and **commit trans** that each RepAgent sender thread processes to generate LTL.
`AvgBytesPerCmd`	Ratio of `NumberOfBytesSent` to `NumberOfCommandsProcessed`.

See *Replication Server Administration Guide Volume 2 > Performance Tuning > Multi-Path Replication > Adaptive Server Monitoring Tables for Multiple Replication Paths*.

See *Adaptive Server Enterprise > Performance and Tuning Series: Monitoring Tables > Introduction to Monitoring Tables > Monitoring Tables in Adaptive Server*.

High-Volume Adaptive Replication and Real-Time Loading

Replication Server 15.7.1 improves memory utilization and support for large transactions in high-volume adaptive replication (HVAR) and real-time loading (RTL).

The improvements include:

- SQT memory consumption control – you can control the maximum memory consumed by unpacked commands in the DSI SQT cache during transaction profiling in HVAR and RTL.

  See:
  - HVAR – *SQT Memory Consumption Control for HVAR* in the *Replication Server Administration Guide*.
  - RTL – *SQT Memory Consumption Control for RTL* in the *Replication Server Heterogeneous Replication Guide*.
- Net-change database size estimation and transaction profiling – Replication Server does not mark a transaction as noncompilable even if the transaction is larger than the DSI SQT cache size.

  See *Net-Change Database Size Estimation and Transaction Profiling* in the *Replication Server Administration Guide Volume 2*.
- Full incremental compilation – Replication Server uses HVAR mode, which is more efficient than continuous replication mode, to compile and replicate large transactions.

See *Full Incremental Compilation for HVAR* in the *Replication Server Administration Guide Volume 2*.

Improvements to Security

Replication Server 15.7.1 introduces several improvements to password security administration and password encryption, and provides the ability to audit commands associated with configuration changes.

Concealing Passwords During Input

Use **isql** options to conceal a password as you type the password when you use **alter user** or **create user**.

See *Concealing Password Input* in the *Replication Server Administration Guide Volume 1*.

Password Policy Administration

You can enforce requirements such as minimum length, mandatory character types, and an expiration period, for user passwords.

Set password requirements for an individual when you create a Replication Server user, or at the server level for all users.

Use the `rs_dictionary` RSSD system table to store character combinations that are not allowed in passwords when you set the **simple_passwords_allowed** parameter to true.

See *Password Configuration Options for All Users* in the *Replication Server Administration Guide Volume 1*.

Password Encryption

Replication Server 15.7.1 changes the encryption algorithm for passwords and decrypts and encrypts all existing passwords according to the new algorithm, even if the existing passwords are encrypted . When you upgrade to Replication Server 15.7.1 and later, the change in encryption takes effect only after you set the site version to 1571 and later.

Replication Server uses password encryption instead of clear text when storing all passwords for new Replication Server installations. Replication Server uses the new algorithm to encrypt all passwords stored in the `rs_users` and `rs_maintusers` RSSD system table, and in the Replication Server configuration file. Replication Server 15.7.1 introduces the `rs_encryptionkeys` RSSD system table and the **RS_random** attribute in the configuration file to support password encryption. Replication Server automatically generates installation specific random values for the `rs_password_key` row in the system table and for the **RS_random** attribute when you start the upgraded Replication Server and Replication Server does not find the values in the table or the configuration file.

You can regenerate the random values for the password encryption keys in the system table and configuration file with the **alter encryption key rs_password_key regenerate** command

With the changes to password security requirements, the **password_encryption** parameter is deprecated.

See *Password Encryption* in the *Replication Server Administration Guide Volume 1* .

Removal of Default Passwords for Replication Server Configuration

With Replication Server 15.7.1, **rs_init** does not provide, suggest, construct, or use default passwords.

You cannot enter USE_DEFAULT or UNCHANGED for the passwords that you specify with several attributes in the Replication Server resource file:

Password Attribute	User ID
rs.rs_rs_sa_pass	Replication Server sa login
rs.rs_idserver_pass	ID Server user
rs.rs_rssd_prim_pass	RSSD primary user
rs.rs_rssd_maint_pass	RSSD maintenance user
rs.rs_rs_pass	Replication Server login name
rs.rs_ltm_rs_pass	Log transfer manager login ID
rs.rs_db_maint_password	Database maintenance user

Instead, enter a password that complies with password security requirements that the administrator has enforced. See the list of resource files and a revised sample resource file in *Replication Server Configuration Guide > Configure Replication Server and Add Databases Using rs_init > rs_init With a Resource File > Using a Resource File for* **rs_init** *> Syntax and Parameters for a Resource File* .

sa User Password Reset

You can reset the password for the sa user if you lose or forget the password.

See *Replication Server Administration Guide Volume 1 > Manage Replication Server Security > Manage Replication Server User Security > Manage Replication Server Login Names and Passwords > Resetting a Lost or Forgotten sa User Password*.

Command Auditing

Enable command auditing for Replication Server to record information about users and commands that users enter at the Replication Server.

See *Replication Server Administration Guide Volume 1 > Manage Replication Server Security > Command Auditing*.

System Table Support for Password Security

To support password security, Replication Server uses the `rs_passwords`, `rs_dictionary`, and `rs_encryptionkeys` system tables and adds new columns to `rs_users`.

See *Replication Server Reference Manual > Replication Server System Tables*.

Security Recommendations

Recommendations for Replication Server security issues such as for performimg administrative tasks, SSL, encryption, permissions and roles, and the configuration file..

- As a best practice, perform administration tasks only on the local Replication Server host. By default, Replication Server does not prevent an administrator who knows the Replication Server host name and port number, from accessing and administering the Replication Server remotely.
- Wait for a master database transaction such as creating a new user or changing a password, to replicate successfully to all replicate Adaptive Servers before executing a user database transaction such as creating a table, that depends on the master database transaction.

  Replication Server maintains the transaction commit order for transactions executed within a single Adaptive Server database. However, Replication Server does not maintain such an order for transactions executed across multiple Adaptive Server databases. For example, at the primary Adaptive Server:
  - To create a master database transaction such as creating the mylogin user , use the sa user to enter:

    ```
    sp_addlogin 'mylogin', 'password'
    go
    use mydb
    go
    sp_adduser
    'mylogin'
    go
    ```
  - To create a user database transaction such as creating the `mytab` table with the mylogin user ID, enter:

    ```
    use mydb
    go
    create table mytab (mycol int)
    go
    ```

  It is possible for Replication Server to replicate the **create table** command before **sp_addlogin** procedure which causes the **create table** to fail on the replicate Adaptive Server because the mylogin user does not yet exist at the replicate database.
- Replication Server can use Secure Sockets Layer (SSL) to provide session-based security. SSL uses certificates issued by certificate authorities (CAs) to establish and verify identities.

If a SSL certificate is compromised, you must request for a new certificate from the CA with a new Replication Server name and certificate number.

- The administrator should control permissions on the Replication Server log to provide read-only access to auditors. By default, any user that you create in Replication Server, who has not been granted any roles, has read-only access to RSSD tables sufficient for a support role.
- Consider disk-level encryption for sensitive data in stable queues.

  Even with connectivity based on SSL between the primary and replicate databases and Replication Server, Replication Server must persist data temporarily in the stable queues, and this persisted data is not encrypted.
- Sybase recommends that you use SSL for connections or routes that transmit sensitive data. The Replication Server Secure Sockets Layer (SSL) Advanced Security option provides session-based security.
- Replication Server stores initial configuration properties such as host name, port, user name, and password, in a file with the `.res` suffix that the **rs_init** utility uses. Set the appropriate umask permissions in UNIX or directory permission in Windows for the `.res` file, or delete the file if you do not require it.

  Although **rs_init** does not require the `.res` file after the initial configuration, Replication Server stores the file in the operating system file system protected only by the operating system permissions.

Performance Enhancements

Replication Server 15.7.1 includes several changes to improve replication performance.

Asynchronous Parser, ASCII Packing, and Direct Command Replication

Obtain improvements throughout the replication process during data transformation and transport by utilizing the asynchronous parser, ASCII packing, and inbound and outbound direct command replication features together.

With Replication Server 15.7, you can use direct replication for inbound commands to reduce command transformation and I/O in the inbound replication path between the Replication Server EXEC and DIST modules.

With Replication Server 15.7.1, the asynchronous parser and outbound direct command replication features improve replication performance between Replication Agent and the Executor thread and between the DIST and DSI modules, while ASCII packing reduces reduces stable queue storage consumption.

Use:

- Asynchronous parser – to reduce the time Replication Agent waits for the Executor by configuring additional Executor threads to parse commands from Replication Agent
- ASCII Packing – with the asynchronous parser to reduce the stable queue storage space consumed by packed commands in the inbound queue
- Direct replication for inbound commands – to reduce command transformation and I/O in the inbound replication path between the Replication Server EXEC and DIST modules
- Direct replication for outbound commands – to reduce command transformation and I/O in the outbound replication path between the Replication Server DIST and DSI modules

You can gain maximum performance improvements and reduction in queue storage consumption by using all of these features together. Instead of configuring each of the features separately, use **async_parser** with **alter connection** to configure them at the same time with their default values. You can also set **async_parser** on and then set the individual parameters independently to fine tune and balance performance and resource consumption.

See *Replication Server Administration Guide Volume 2 > Performance Tuning > Suggestions for Using Tuning Parameters > Asynchronous Parser, ASCII Packing, and Direct Command Replication*.

Usability Improvements

Replication 15.7.1 includes several changes to improve usabilty.

Reduce Replication Definitions for Customized Function Strings in Warm Standby and MSA Environments

In a replication system containing only Adaptive Server databases, you do not need to create replication definition for a primary table or stored procedure in a warm standby environment or multisite availability (MSA) environment if the sole purpose of the replication definition is to specify a customized function string for the replicate table or stored procedure.

With Replication Server 15.7, you do not need to create a replication definition for a primary table, in a warm standby environment or multisite availability (MSA) environment, if the sole purpose of the replication definition is to specify primary-key columns, or quoted table or column names. See *Primary Key Columns and Quoted Table or Column Names* in the *Replication Server Administration Guide Volume 1*.

With Replication Server 15.7.1, you can create a customized function string directly against a replicate or standby table or stored procedure without defining a replication definition for the table or stored procedure. This type of function string is called a target-scope function string, which further reduces requirements for replication definitions in a warm standby or MSA environment.

See *Target-Scope Customized Function Strings* in the *Replication Server Administration Guide Volume 1*.

Stored Procedure Support

Use the **rs_helpobjfstring** stored procedure to display information about target-scope function strings. See *Replication Server Reference Manual > RSSD Stored Procedures > **rs_helpobjfstring***.

System Table Support

Replication Server introduces the `rs_targetobjs` system table to store information about target tables or stored procedures. Replication Server does not replicate the values in `rs_targetobjs` to the RSSDs of other Replictation Servers. `rs_targetobjs` is in the STS cache with (`objname`, `objowner`, `dbid`, `objtype`) as the STS primary cache key. Use **sts_full_cache_rs_targetobjs** to enable or disable full caching of the table:

```
configure replication server set sts_full_cache_rs_targetobjs to
{on|off}
```

The default for **sts_full_cache_rs_targetobjs** is off.

Replication Server changes the datatype for the `attributes` column in the `rs_funcstrings` table from `smallint` to `int`. .

See *Replication Server Reference Manual > Replication Server System Tables > `rs_targetobjs`*.

Simplified Upgrade

Replication Server offers a simplified process for upgrading user databases and RSSDs.

With Replication Server 15.6, you can use **sysadmin upgrade, "route"** to upgrade routes seamlessly instead of the Replication Manager plug-in to Sybase Central™.

With Replication Server 15.7.1, you can use an upgrade process that is simplified further to automatically upgrade the embedded Replication Server System Database (ERSSD) or Replication Server System Database (RSSD). Replication Server also connects to each user database that it has maintenance user access to and automatically applies the upgrade scripts to the database.

See *Replication Server Configuration Guide > Upgrade or Downgrade Replication Server > Upgrading Replication Server > Upgrading RSSD or ERSSD with repserver*.

Systems Management Tools

With Replication Server 15.7.1, Replication Server installation media does not include Replication Manager and Replication Monitoring Services (RMS).

Replication Server 15.7.1 includes Sybase Control Center which you can use to manage your replication system. See *Sybase Control Center 3.2.6 for Replication*.

To continue using Replication Manager and Replication Monitoring Services, download and install these tools under Replication Server in the Sybase Web site.

Route upgrade in the Replication Manager plug-in to Sybase Central is deprecated. Use the **sysadmin upgrade "route"** Replication Server command instead. See *Upgrading Routes* in the *Replication Server Configuration Guide*.

See also

- *Simplified Upgrade* on page 16
- *Downloading Sybase EBFs and Maintenance Reports* on page 181

New Features in Replication Server 15.7

Replication Server® 15.7 includes performance, usability, process, and database support enhancements.

Replication Server Licensing

Replication Server 15.7 is released as the Enterprise Edition.

Replication Server 15.7 includes enhancements to real-time loading (RTL) replication to Sybase® IQ. If you are using the Replication Server Real-time Loading Edition, you can use the RTL enhancements by upgrading to Replication Server 15.7.

See *Replication Server Installation Guide > Planning Your Installation > Obtaining a License*.

See also

- *Real-Time Loading and High-Volume Adaptive Replication* on page 32

Multi-Path Replication

Use multiple replication paths to increase replication throughput and performance, and reduce contention.

Multi-Path Replication™ supports the replication of data through different streams, while still maintaining data consistency within a path, but not adhering to the commit order across different paths.

A replication path encompasses all the components and modules between the Replication Server and the primary or replicate database. In multipath replication, you can create multiple primary replication paths for multiple Replication Agent connections from a primary database to one or more Replication Servers, and multiple replicate paths from one or more Replication Servers to the replicate database. You can configure multi-path replication in warm standby and multisite availability (MSA) environments. You can convey transactions over dedicated routes between Replication Servers to avoid congestion on shared routes, and you can dedicate an end-to-end replication path from the primary database through Replication Servers to the replicate database, to objects such as tables and stored procedures.

See *Replication Server Administration Guide Volume 2 > Performance Tuning > Multi-Path Replication*.

License

Multi-Path Replication is licensed as part of the Advanced Services Option. See *Replication Server Installation Guide > Planning Your Installation > Obtaining a License*.

System Requirements

Replication Server supports multipath replication between Adaptive Server databases where the primary data server is Adaptive Server 15.7 and later.

Performance Enhancements

Replication Server 15.7 includes several changes to improve replication performance.

SQM Command Cache

Use the SQM command cache to store parsed data from the Executor thread that the Distributor thread can retrieve directly, and therefore improve replication performance.

The Executor thread transfers LTL commands from a Replication Agent to Replication Server. The Executor thread parses the LTL commands and stores them in an internal parsed format. The parsed data is then packed in binary format. The Executor thread sends the binary data to the SQM thread so that the Executor thread can receive new data from the Replication Agent. The SQM thread stores the binary data in the SQM cache until the data is written to the inbound stable queue. The Distributor thread retrieves the binary data, restores the data to the original format, and determines where to send the data.

Set **cmd_direct_replicate** on for the Executor thread to send internal parsed data along with the binary data. Replication Server stores the parsed data in a separate SQM command cache. The parsed data in the SQM command cache maps to the binary data stored in SQM cache. When required, the Distributor module can retrieve and process data from parsed data directly, and save time otherwise spent parsing binary data.

Use the **sqm_cmd_cache_size** and **sqm_max_cmd_in_block** parameters to set the the SQM command cache memory configuration.

See *Replication Server Administration Guide Volume 2 > Performance Tuning > Suggestions for Using Tuning Parameters > SQM Command Cache*.

Executor Command Cache

Use the Executor command cache to cache column names and datatypes for a primary Adaptive Server database table, when a Sybase RepAgent initially sends an **insert**, **delete**, or **update** LTL command for that table.

Metadata such as column name and datatype are part of the table schema that RepAgent sends as well as the data associated with an **insert**, **delete**, or **update** command. However, with caching:

- RepAgent sends the metadata and data associated with an **insert**, **update**, or **delete** command only when the RepAgent processes an operation for that specific table the first time since the RepAgent started, or since a connection with Replication Server was restarted. RepAgent does not send the table metadata when RepAgent subsequently processes transactions for that table.
- RepAgent can resend metadata and data if there is not enough memory in the RepAgent to keep all the schema definitions.
- RepAgent sends the metadata and data of a table when the RepAgent processes a modification on a specific table after the table schema has been changed, for example, after an Adaptive Server **alter table** operation.

To replicate subsequent operations on the same table, RepAgent sends only the column data, since the Replication Server Executor command cache stores the metadata. The combination of RepAgent metadata reduction and using the Replication Server Executor command cache improves replication performance because caching:

- Reduces the time spent by RepAgent packing metadata into the Log Transfer Language (LTL) packet.
- Reduces network traffic by increasing the amount of data sent in each packet.
- Allows RepAgent to dedicate the time saved to scanning the primary database log instead of packing metadata.
- Allows the Replication Server Executor to process tables with large number of columns more efficiently.

Note: The cache contains only metadata from tables that have been modified by an **insert**, **update**, or **delete** operation.

See *Replication Server Administration Guide Volume 2 > Performance Tuning > Suggestions for Using Tuning Parameters > Executor Command Cache*.

System Requirements

Table metadata reduction requires LTL version 740 or later, and Adaptive Server 15.7 or later.

Higher Limit for sqm_cache_size

The maximum limit for **sqm_cache_size** has been increased to 4096 from the previous limit of 512.

Increasing the upper limit for **sqm_cache_size** allows Replication Server to keep more transactions in cache, which affects the overall performance of cache searches.

See

- *Replication Server Reference Manual > Replication Server Commands >* ***configure replication server***
- *Replication Server Administration Guide Volume 2 > Performance Tuning > Configuration Parameters that Affect Performance > Replication Server Parameters that Affect Performance*

Dedicated Daemon For Deleting Segments

Set **sqm_async_seg_delete** to on to enable a dedicated daemon for deleting segments and improve performance for inbound and outbound queue processing.

You must set **sqm_async_seg_delete** at the server level with **configure replication server**.

Default: on

You must restart Replication Server for any change to the parameter setting to take effect.

Since **sqm_async_seg_delete** is on by default, Replication Server may require a larger partition when you upgrade to version 15.7 or later. See:

- *Replication Server Configuration Guide > Preparation for Installing and Configuring Replication Server > Plan the Replication System > Initial Disk Partition for Each Replication Server.*
- *Replication Server Administration Guide Volume 1 > Replication Server Technical Overview > Transaction Handling with Replication Server > Stable Queues > Partitions for Stable Queues.*
- Replication Server Reference Manual > Replication Server Commands > **alter partition**.

Usability and Process Improvements

Replication 15.7 includes several changes to improve usabilty and processes.

Reduce the Use of Replication Definitions

In a replication system containing only Adaptive Server databases, you can reduce the need for replication definitions for tables in a warm standby environment or multisite availability (MSA) environment because RepAgent for Adaptive Server uses Log Transfer Language

(LTL) to specify a table or column name that may be quoted, and whether a table column is part of the table primary key.

Since RepAgent sends the primary key and quoted identifier information to Replication Server, you do not need a replication definition if the sole purpose of the replication definition is to specify primary key and quoted identifier information. The reduced requirement for replication definitions makes it easier to manage a replication environment involving databases with many tables, tables with many columns, or tables that change schema frequently. Replication performance improves for tables that are currently without replication definitions, as RepAgent directly provides Replication Server with table primary key information, so that Replication Server packs only the primary key columns in the **where** clauses of **update**, **delete**, and **select** commands.

You do not need to create a replication definition for a primary table if the sole purpose of the replication definition is to specify some or all of the following:

- The primary-key columns
- When there is a table or column name that may be quoted.

See *Replication Server Administration Guide Volume 1 > Manage Replicated Objects Using Multisite Availability > Reduce the Use of Replication Definitions and Subscriptions* to configure the replication system to reduce replication definitions.

System Requirements

RepAgent sends primary-key and quoted identifier information only with LTL version 740 or later, which is supported by Adaptive Server 15.7 and later, and Replication Server 15.7 and later.

Changes to rs_functions

Several new system tables have been added to Replication Server 15.7 in place of `rs_functions` .

Replication Server 15.7 adds these system tables:

- `rs_clsfunctions` – stores class-wide functions.
  See *Reference Manual > Replication Server System Functions > rs_clsfunctions*.
- `rs_objfunctions` – stores object-wide functions.
  See *Reference Manual > Replication Server System Functions > rs_objfunctions*.
- `rs_asyncfuncs` – stores information about user-defined functions against replication definitions. The same rows are also stored in `rs_objfunctions`.
  See *Reference Manual > Replication Server System Functions > rs_asyncfuncs*.

In versions of Replication Server earlier than 15.7, `rs_functions` stored information about class-wide and object-wide functions, which are cached by *funcname*. As many replication definitions share the same function name, such as **rs_insert**, **rs_update**, and **rs_delete**, the rows for object-wide functions should not be cached by *funcname*. In version

15.7 and later, the rows in `rs_functions` are split into two categories and stored in `rs_clsfunctions` and `rs_objfunctions`.

In version 15.7 and later, `rs_functions` is no longer a table. To support Replication Server backward compatibility, `rs_functions` is maintained as a view from the union of `rs_clsfunctions` and `rs_objfunctions`.

See `rs_asyncfuncs`, `rs_clsfunctions`, and `rs_objfunctions` in *Replication Server Reference Manual > Replication Server System Tables*.

- *Reference Manual > Replication Server System Tables >* `rs_asyncfuncs`
- *Reference Manual > Replication Server System Tables >* `rs_clsfunctions`
- *Reference Manual > Replication Server System Tables >* `rs_objfunctions`

Memory Consumption Controls

The configuration parameter **memory_limit** has been enhanced to control memory consumption and prevent Replication Server from automatically shutting down when it exceeds the defined value of available memory. A new configuration parameter, **memory_control**, manages the memory control behavior of threads in Replication Server.

With version 15.7, you can configure Replication Server to show warning messages when the memory consumption exceeds a defined threshold percentage of the total available memory. Two new configurable parameters support this enhanced memory management:

- **mem_warning_thr1** – specifies the threshold percentage of the total memory used before the first warning message is generated.
  Default: 80% of **memory_limit** value.
  Range: 1 – 100.
- **mem_warning_thr2** – specifies the threshold percentage of the total memory used before the second warning message is generated.
  Default: 90% of **memory_limit** value.
  Range: 1 – 100.

In addition, Replication Server 15.7 also addresses the issue of automatic shutdown of Replication Server when it exceeds the available memory defined by **memory_limit**. In Replication Server, the threads that require significant amount of memory are:

- DSI
- EXEC
- SQT

In version 15.7, these threads execute memory control by performing a memory usage check before receiving or processing new data. During memory control, if the memory usage is found to be high, thread functioning is adjusted by:

- Stopping the thread from grouping new data, and cleaning and processing existing data; or,
- Making the thread go into a sleep mode such that it does not receive new data until memory is available.

There are three new server-level configuration parameters for managing flow control in EXEC, DST, and SQT threads:

- **mem_thr_dsi** – specifies the percentage of the total memory used to force the DSI thread to stop populating the SQT cache.

  Default: 80% of **memory_limit** value.
- **mem_thr_exec** – specifies the percentage of the total memory used to force the EXEC thread to stop receiving commands from RepAgent.

  Default: 90% of **memory_limit** value.
- **mem_thr_sqt** – specifies the percentage of the total memory used to force the SQT thread to flush the largest transaction from its cache.

  Default: 85% of **memory_limit** value.

A new server-level configuration parameter, **memory_control**, manages the memory control behavior of threads. Valid values for **memory_control** are enable (the default value) or disable. In this way, Replication Server controls the memory consumption and does not shut down because of memory issues.

Use **configure replication server** to alter the default values for the new configuration parameters. Use **admin config** to view the default or existing values.

See:

- *Replication Server Reference Manual > Replication Server Commands >* ***configure replication server***
- *Replication Server Administration Guide Volume 2 > Performance Tuning > Configuration Parameters that Affect Performance > Replication Server Parameters that Affect Performance*

Monitor Thread Information

Use **admin who** to provide information on the memory control behavior of the thread:

State	Description
Controlling Mem	The thread is executing memory control.
Sleeping For Mem	The thread is sleeping until memory is available.

See *Replication Server Reference Manual > Replication Server Commands >* ***admin who***.

Memory Management Statistics

Use **admin stats** to view the memory management statistics. Memory counters are enabled in the `rsh` module. To report the memory counters, use:

```
admin stats,rsh display_name instance_id
```

where:

- *display_name* – is the name of a counter. Use **rs_helpcounter** to obtain valid display names. *display_name* is used only with *module_name*.
- *instance_id* – identifies a particular instance of a module such as SQT or SQM. To view instance IDs, execute **admin who** and view the *Info* column. For `rsh` module, the *SPID* must be used. To view *SPID*, execute **admin who** and view the *Spid* column.

See *Replication Server Reference Manual > Replication Server Commands >* ***admin stats***.

Unicode Enhancements

A new Data Server Interface (DSI) configuration parameter, **unicode_format**, has been added to support sending Unicode data in `U&"` format.

Earlier versions of Replication Server required you to set your character set to UTF-8 for replicating all Unicode datatypes such as `unichar`, `univarchar`, and `unitext`. Replication Server 15.7 removes this limitation, and can now send Unicode data in either of these formats:

- Character string
- `U&"`

Adaptive Server Enterprise also supports these Unicode data formats.

When configuring your Replication Server, set **unicode_format** to one of these values:

- `string` – unicode characters are converted to character string format. For example, the `string` “hello” is sent out as “hello”.
- `ase` – unicode characters are sent out in `U&''` format. For example, the `string` “hello” is sent out as “U&'\0068\0065\006c\006c\006f’ ”. The two-byte unicode value is sent in network order as required by Adaptive Server Enterprise.

unicode_format is a semidynamic parameter; either restart the connections or restart Replication Server for the changes to take effect.

See *Replication Server Reference Manual > Replication Server Commands >* ***configure replication server***.

Requesting SySAM License Information

Replication Server 15.7 introduces a new command, **sysadmin lmconfig**, for configuring and showing license management-related information.

See *Replication Server Reference Manual > Replication Server Commands >* ***sysadmin lmconfig***.

Subscription Name Extension

In Replication Server 15.7 or later, the subscription length limit is increased to 255 characters from the previous limit of 30 characters in `rs_subscriptions`.

The *subname* column in the `rs_subscriptions` system table changed from *varchar(30)* to *varchar(255)*.

See Replication Server Reference Manual> Replication Server System Tables >`rs_subscriptions`.

Mixed-version Replication Environment

In a mixed-version replication environment, you must ensure that your replicate Replication Server and your primary Replication Server version are both 15.7 or later.

Stripping Trailing Zeros

Set **varbinary_strip_trailing_zeros** to off to enable the replication of trailing zeros in `varbinary` values.

The default setting of on strips trailing zeros from `varbinary` values. The default setting has been the behaviour of all Replication Server versions earlier than 15.7.

You must set **varbinary_strip_trailing_zeros** at the server level with **configure replication server**. You need not restart Replication Server, or suspend and resume connections for any change in the parameter to take effect.

Sybase Control Center for Replication and Data Assurance

Sybase Control Center provides a single comprehensive Web administration console for real-time performance, status, and availability monitoring of large-scale Sybase enterprise servers. It includes historical monitoring, threshold-based alerts and notifications, alert-based script execution, and intelligent tools for identifying performance and usage trends.

Sybase Control Center for Replication provides status information at a glance, using server monitors and a heat chart for displaying the availability or status of a specific server. The server monitors display high-level information, such as server version and platform. The

server monitors also display critical performance counters to aid you in troubleshooting replication performance.

Sybase Control Center for Data Assurance (DA) supports data comparison from a primary database to one or more replicate databases. You can also schedule comparison tasks. Sybase Control Center for Data Assurance uses and deploys its own SQL Anywhere® database, which stores system and configuration settings, tasks, and task history. An automatic reconciliation option identifies missing, orphaned, and inconsistent rows when you create DA jobs.

To help you control the flow of data and configure replication parameters to improve server performance, Sybase Control Center for Replication provides a quick administration tool that you can easily access through every replication monitor.

In addition to the monitors, Sybase Control Center for Replication provides a topology view that graphically displays the servers, the connections between servers, data flow in the environment, and sources and targets for a replication path. Graphs and charts are also available for monitoring performance counters.

In *Sybase Control Center 3.2.4*, see *Sybase Control Center for Replication* and *Sybase Control Center for Data Assurance*.

License

Sybase Control Center is licensed free when you have a paid license for a product managed by Sybase Control Center such as Replication Server and Replication Server Data Assurance option. Evaluation licenses are also available. See *Sybase Control Center Installation Guide > Planning Your Installation > Obtaining a License*.

Feature Comparison Between Sybase Control Center for Replication and Sybase Central for Replication

Compare support for the replication systems management between Sybase Control Center for Replication version 3.2.3 (SCC Replication), and version 15.0 and later of the Replication Manager Plug-in (RMP) of Sybase Central and Replication Monitoring Services (RMS).

Feature	SCC for Replication 3.2.3	RMP 15.x	RMS 15.x
Monitor Replication Server and Replication Agent status	X	X	X
Monitor Replication Server and Replication Agent availability	X	X	X
Monitor replication performance	X		

Feature	SCC for Replication 3.2.3	RMP 15.x	RMS 15.x
Monitor end-to-end latency (**rs_ticket** heartbeat)	X		X
Monitor paths that include ASE primary and replicate databases	X		
Monitor paths that include Sybase IQ replicate databases	X		
Monitor paths that include non-Sybase primary and replicate databases: Oracle, Microsoft SQL Server, and IBM DB2 for Linux, UNIX and Windows	X		
Monitor Replication Server statistics	X	X	X
Save historical performance statistics	X		
Chart historical performance statistics	X		
Display replication topology	X		
Trace replication paths	X		
Display Heat chart	X		
Configure Alert notifications	X		X
Configure Replication Server user security and roles	X		
Display license information for version 15.7 and later of Replication Server and Replication Agent	X		
Configure Replication Server and Replication Agent	X	X	X

Feature	SCC for Replication 3.2.3	RMP 15.x	RMS 15.x
Suspend and resume Replication Server and Replication Agent	X	X	X
Shut down Replication Server and Replication Agent		X	X
Group servers in a way that you choose (for example: geographically or functionally)	X		X
Use RCL and SQL editor to execute commands at Replication Server		X	
Use a command-line interface such as **isql**, to execute the API commands			X
Use a wizard to simplify setting up an Adaptive Server replication environment		X	
Set up Adaptive Server Rep-Agent thread		X	
Configure Adaptive Server automatic materialization		X	
Create and delete Replication Server connections		X	
Create and delete Replication Server logical connections		X	
Create and delete Replication Server routes		X	
Create and delete replication definitions and subscriptions for multisite availability (MSA)		X	

Feature	SCC for Replication 3.2.3	RMP 15.x	RMS 15.x
Create, alter, and delete Replication Server replication definitions and subscriptions		X	
Create, alter, and delete Replication Server users		X	
Upgrade Replication Server routes		X	
View Replication Server queue data		X	

Enhancements to Adaptive Server Replication Support

Replication Server 15.7 supports Adaptive Server replication.

Automatically Start RepAgent

With Adaptive Server 15.5 ESD #5 and later, you can use the **auto start** parameter with **sp_config_rep_agent** to specify whether RepAgent automatically starts when Adaptive Server restarts and recovers the database.

RepAgent starts automatically when Adaptive Server restarts if you have previously started RepAgent at least once with **sp_start_rep_agent** and you did not stop RepAgent with **sp_stop_rep_agent**. With Adaptive Server 15.5 ESD #5 and later, RepAgent also starts automatically if you set **auto start** to true with **sp_config_rep_agent**.

If you shut down RepAgent with **sp_stop_rep_agent**, RepAgent does not automatically start when the database comes online unless you previously set **auto start** to true. Otherwise, you must execute **sp_start_rep_agent** to start RepAgent.

The syntax is:

```
sp_config_rep_agent
[...
'auto start'[, 'true' | 'false']]
```

Set to true for RepAgent to start automatically when you restart Adaptive Server. The default is false.

In the *Replication Server Administration Guide Volume 1 > Manage RepAgent and Support Adaptive Server*, see:

- *Configuration Parameters Affecting RepAgent*

- *Starting RepAgent*
- *Stopping RepAgent*

Real-Time Loading and High-Volume Adaptive Replication

Replication Server 15.7 includes performance and usability improvements to real-time loading (RTL) and high-volume adaptive replication (HVAR).

License

Replication to Sybase IQ using RTL is available as part of the Real-Time Loading option.

Replication to Adaptive Server using HVAR is available as part of the Advanced Services Option.

See *Replication Server Installation Guide > Planning Your Installation > Obtaining a License*.

Database and Platform Support

- Sybase IQ – you can use real-time loading to replicate into Sybase IQ version 12.7 ESD #3 and later. See *Replication Server Release Bulletin > Product Compatibility > Replication Server Interoperability* for the latest supported Sybase IQ versions and platforms.
- Adaptive Server – Replication Server supports replication to Sybase IQ from Adaptive Server version 15.0.3 or version 15.5 and later.
- Oracle – Replication Server supports replication to Sybase IQ from Oracle 10g and 11g. See *Replication Server Options 15.5 > Release Bulletin Replication Agent 15.5 > Product Summary > Compatible Products*.

See also

- *Replication Server Licensing* on page 19

Memory Consumption Control

To reduce memory consumption in RTL, Replication Server supports full incremental compilation and allows you to control the size of net-change databases. To reduce memory consumption in HVAR, you can control the size of compilable groups.

Full Incremental Compilation for RTL

Full incremental compilation is only available for RTL. Full incremental compilation improves replication performance for RTL by reducing memory consumption during the processing of large compilable transactions that contain many commands.

Full incremental compilation can compile large transactions containing mixed **insert**, **delete**, or **update** operations. Replication Server uses full incremental compilation to apply a large compilable transaction to the replicate database, using multiple in-memory net-change database instances.

See *Replication Server Heterogeneous Replication Guide > Sybase IQ as Replicate Data Server > Sybase IQ Replicate Database Configuration > Memory Consumption Control > Full Incremental Compilation*.

Control Net-Change Database Size for RTL

Reduce memory consumption by the net-change database by triggering the net-change database to flush data to the replicate database once the net-change database size reaches a threshold size.

You can control the maximum net-change database size that Replication Server can generate. Once the size reaches the threshold you set, Replication Server stops compiling new commands and transactions into the compiled transaction that Replication Server is building in the net-change database, performs the bulk apply of the compiled group to the replicate database, clears the net-change database, and releases the memory consumed by the net-change database.

See *Replication Server Heterogeneous Replication Guide > Sybase IQ as Replicate Data Server > Sybase IQ Replicate Database Configuration > Memory Consumption Control > Net-Change Database Size*.

Control the Size of Compilable Groups for HVAR

Reduce memory consumption amd improve performance by setting a threshold for the size of large transactions that can be compiled.

Once the threshold is reached, Replication Server applies the large transaction using the continuous replication mode. Replication Server continues to use HVAR to compile smaller compilable transactions into groups and applies the compiled groups to the replicate database as soon as the group size reaches the threshold you set.

See *Replication Server Administration Guide Volume 2 > Performance Tuning > Advanced Services Option > High Volume Adaptive Replication > Memory Consumption Control*.

Setting Sybase IQ Database Options

You can use the **rs_session_setting** function with the **create function string** command to set the values for Sybase IQ parameters for the duration of the connection to the Sybase IQ replicate database. For example, you can set parameter values to optimize performance.

See *Replication Server Heterogeneous Replication Guide > Sybase IQ as Replicate Data Server > Sybase IQ Replicate Database Configuration > Replication Server Installation > Setting Sybase IQ Database Options*.

Schema Transformation and Datatype Translation

RTL or HVAR support replication even if the primary and replicate database schema or column datatypes differ.

You can use HVAR and RTL to replicate:

- A subset of columns in a primary table to the replicate table.
- Columns and tables even if the primary and replicate table and column names are different, by using replication definitions.
- Columns even if the primary and replicate column datatypes are different.

  Note: HVAR and RTL support for replication between different datatypes is equivalent to the existing column-level translation support that Replication Server provides with continuous-mode replication.

- To tables that have more columns than the primary tables:
  - Adaptive Server replicate tables – set the **NULL-able** attribute for the columns in the replicate table that you do not want Replication Server to populate.

    If you define a default value for a replicate Adaptive Server column, you need not set the **NULL-able** attribute for the column since the replicate Adaptive Server automatically fills the column with the default value.
  - Sybase IQ replicate tables – set the **NULL** attribute for the columns in the replicate table that you do not want Replication Server to populate.

    If you define a default value for a replicate Sybase IQ column, you need not set the **NULL-able** attribute for the column since the replicate Sybase IQ automatically fills the column with the default value.

Note: RTL and HVAR do not support customized function strings to modify how data is replicated to the replicate database.

Changes to Parameter Default Values

Replication Server 15.7 includes changes to the default values for several parameters. If you upgrade to Replication Server 15.7, Replication Server uses the default values from the earlier version.

Table 4. Changes to Parameter Default Values

Parameter	Old Value	New Value	Downgrading from Version 15.7	In *Replication Server Reference Manual > Replication Server Commands*, see:
dsi_compile_max_cmds	100,000 commands	10,000	Downgrading does not change the value you have set.	**alter connection**

Parameter	Old Value	New Value	Downgrading from Version 15.7	In *Replication Server Reference Manual > Replication Server Commands*, see:
num_msg_queues	178 Open Server™ message queues	300	Downgrading does not change the value you have set.	**configure replication server**
num_msgs	45,568 Open Server message queue messages	91,136	Downgrading does not change the value you have set.	**configure replication server**
num_threads	50 Open Server threads	150	Downgrading does not change the value you have set.	**configure replication server**
queue_dump_buffer_size	1000 bytes	32,768	Downgrading does not change the value you have set.	**configure replication server**
rsi_packet_size	2048 bytes	4096	Downgrading does not change the value you have set.	**alter route**
sts_cachesize	100 rows	1000	Downgrading does not change the value you have set.	**configure replication server**

See *Replication Server Reference Manual > Replication Server Commands* for descriptions of the parameters, examples and usage information.

Replication Server Data Assurance Option

Replication Server Data Assurance (DA) Option compares row data and schema between two or more Adaptive Server databases, and reports and optionally reconciles, discrepancies.

Replication Server Data Assurance Option is available as a separately licensed product for Replication Server and supports Replication Server versions 15.1 and later.

Replication Server Data Assurance Option is licensed through SySAM license manager and is available on multiple platforms. See Replication Server Data Assurance Option documentation for additional information.

New Feature in Replication Server Version 15.6 ESD #1

Replication Server 15.6 ESD #1 integrates Replication Server with Sybase IQ InfoPrimer.

Sybase IQ InfoPrimer provides effective capabilities for transforming and loading data into a Sybase IQ database, but its extract capability lacks the real-time monitoring of Replication Server that is needed to maintain a replicate Sybase IQ database with the most current data. The Replication Server Real-Time Loading (RTL) feature uses bulk operation processing and compiled operations to achieve high-performance replication, but Replication Server lacks the data transformation and loading capabilities of Sybase IQ InfoPrimer. With the integration of Replication Server and Sybase IQ InfoPrimer, you can maintain a near real-time copy of Adaptive Server data in a replicate Sybase IQ database with different schema than the source.

Replication Server and Sybase IQ InfoPrimer Integration

The integrated Replication Server and Sybase IQ InfoPrimer solution works in two parts: initial data materialization and ongoing data processing.

Materialization

The integrated Replication Server and Sybase IQ InfoPrimer solution performs a nonatomic bulk materialization of data from an Adaptive Server primary database to a replicate Sybase IQ database. The materialization is based on the Replication Server bulk materialization option and uses autocorrection where required.

Sybase IQ InfoPrimer creates staging tables on the replicate Sybase IQ database and performs the data-extract step of the materialization process on each primary database table. Transformation stored procedures execute against the stage tables, and the result is written to base tables. The base tables, also known as end-user tables, are then used for business analysis.

Ongoing Data Processing

For specified tables, Replication Server uses the same staging tables and transformation stored procedures that were created in the materialization phase. Where possible, Replication Server compiles and loads operations to the staging tables, after which Replication Server executes the transformation stored procedures to update the base tables. In this way, Replication Server maintains a near real-time copy of data in the replicate Sybase IQ database.

Licensing

Special licensing requirements apply to the integration of Replication Server and Sybase IQ InfoPrimer.

Table 5. Replication Server and Sybase IQ InfoPrimer Integration Licenses

Product	Features	Description	License
Replication Server 15.6 ESD #1	Real-Time Loading (RTL)	Allows replication to Sybase IQ from Adaptive Server. **Note:** You cannot use the Real-Time Loading Edition to replicate to Adaptive Server or Oracle.	REP_RTL_IQ
Sybase IQ InfoPrimer 15.3	Sybase IQ InfoPrimer	Used for the extraction and loading of data from Adaptive Server and transformation of data in Sybase IQ.	SY_INFOPRIMER _SERVER

Using the Replication Server and Sybase IQ InfoPrimer Integration

Use Sybase IQ InfoPrimer to materialize data into Sybase IQ with Replication Server materialization methods, and configure Replication Server to process updates made to primary data.

1. Before materialization:
   - Create an Extract and Load (EL) project in Sybase IQ InfoPrimer, selecting **Materialization with Replication Server**.

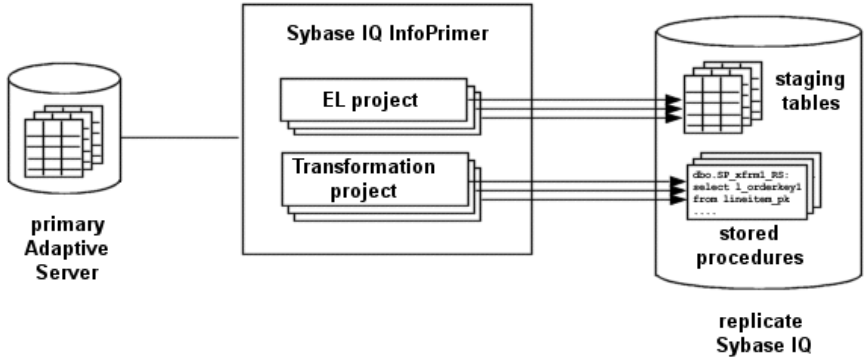
     In the RepServer tab of the EL project editor, you must also specify connection information for the primary Replication Server and the replicate Replication Server, if it is different from the primary. Sybase IQ InfoPrimer adds a command to the Processing tab. Do not modify or delete this command.

     For each source table, Sybase IQ InfoPrimer creates the required staging table definitions. Generate these staging tables on the replicate Sybase IQ database by selecting the **Create missing destination tables** icon on the Tables tab of the EL project editor.

     Note: If you are attempting to rematerialize, you must clear the `rs_status` table.
   - Create a SQL Transformation project, and model the transformation for each set of staging tables (insert, update, and delete) that have been generated in the replicate

Sybase IQ database. Use the SQL Transformation project to deploy each set of transformations as a stored procedure in the replicate Sybase IQ database.

Note: These transformation stored procedures truncate their corresponding staging tables when operations have been processed.

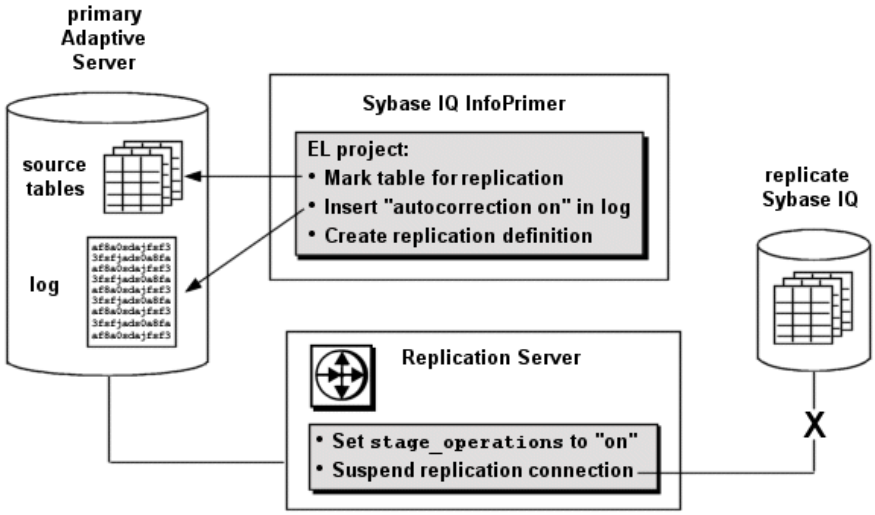


2. In your Replication Server instance, use the **stage_operations** connection parameter to configure the replicate database connection to stage operations for the tables specified in your EL project.

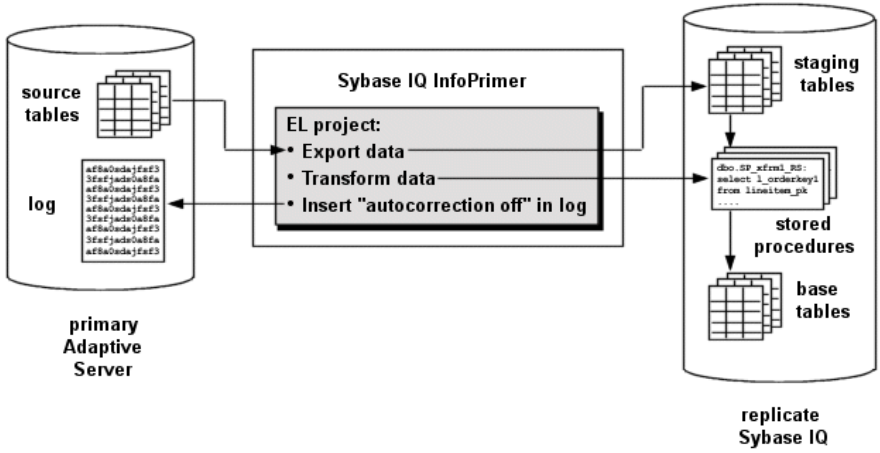
   Note: If **stage_operations** is set to on, Replication Server ignores the setting of **dsi_compile_enable** and enables RTL for the connection. Operations are compiled, as when **dsi_compile_enable** is set to on, and then staged.

   In Sybase IQ InfoPrimer, execute your EL project. For each primary table specified, the EL project:

   a) Marks the table for replication.
   b) Inserts an `autocorrection on` record in the primary database log, which results in suspension of the Replication Server replicate database connection.
   c) Creates a table replication definition in the RSSD.

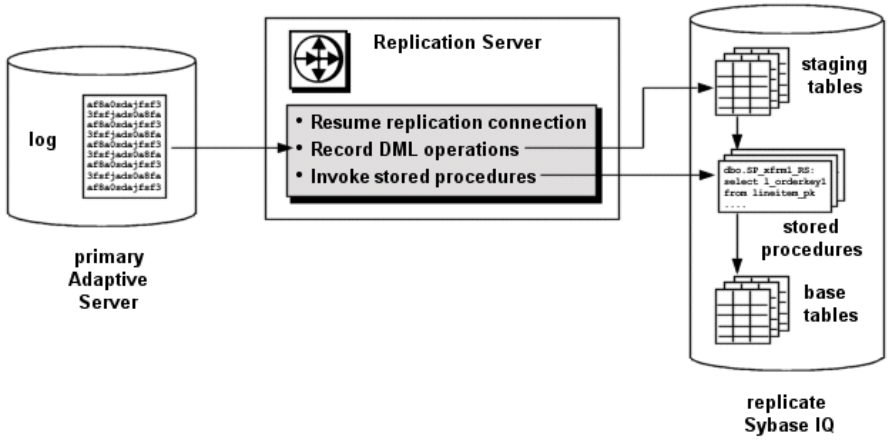


3. Your Sybase IQ InfoPrimer EL project exports primary data for each table into the corresponding staging tables on the replicate Sybase IQ, executes the transformation stored procedures, and inserts an `autocorrection off` record in the primary database log.



4. The Replication Server replicate database connection is resumed, and Replication Server processes any further changes to marked primary database tables using the staging tables and transformation stored procedures on the replicate Sybase IQ database.

Note: Sybase IQ InfoPrimer is only used for data migration and creating the staging tables and transformation stored procedures. It is not involved in replication.



Base Tables

Base tables contain data in its final form at the replicate Sybase IQ database.

Base table data can result from:

- SQL transformations – When the Replication Server replicate database connection has been configured to stage operations, the result of transformation stored procedures executing against the stage tables is written to the base tables.
- Replication – If a table has been excluded from staging, Replication Server bypasses the staging tables and replicates data directly to the base tables.

Staging Tables

If your Replication Server replicate database connection has been configured to stage operations logged for a primary table, these operations are compiled where possible and written to staging tables on the replicate Sybase IQ database.

For each table to be staged, there are three staging tables, each corresponding to DELETE, INSERT, and UPDATE operations:

- *owner_table_name*_DELETE_RS
- *owner_table_name*_INSERT_RS
- *owner_table_name*_UPDATE_RS

where *owner* and *table_name* are the owner and name of the corresponding primary database table. The names of these tables are generated by your EL project, and they cannot be changed.

Note: The Tables tab of your EL project displays only the insert staging table. However, the Table Creation window displays all three staging tables corresponding to a specified primary database table.

You must identify which primary database tables are to be staged in a Sybase IQ InfoPrimer EL project. You may also selectively exclude replicate tables from staging. For a table that has been excluded from staging, no corresponding staging tables need to be created, and data will be replicated from the primary table to a replicate table in the replicate Sybase IQ database.

If you configure a replicate database connection to stage tables but no staging tables exist in the replicate Sybase IQ database, the replicate database connection will be suspended. If a replication definition includes columns that are declared as identity columns, these will not be declared as identity columns in the corresponding staging tables.

Table Compilation

Compilation is not performed on noncompilable tables. Tables are considered noncompilable if they have RTL disabled, modified function strings, or minimal column replication enabled. Operations to noncompilable tables are captured in an ordered list and applied to the corresponding replicate table after compilation is complete.

Note: After Replication Server commits a staged operation, the transformation stored procedures truncate the corresponding staging tables. You should therefore not use the Replication Server **rs_subcmp** utility to validate staging tables.

Insert Staging Table Structure

Apart from changes and filtering applied by the corresponding replication definition, the insert staging table contains the same number of columns and the same column names as the primary table.

Delete Staging Table Structure

The delete staging table contains only the primary-key columns specified in the corresponding replication definition.

If no primary key is specified in the replication definition, the delete staging table contains all published columns except for:

- approximate numeric columns
- encrypted columns
- Java columns
- LOB columns

Note: Sybase recommends that you specify a primary key in your table replication definition to simplify processing and improve performance.

Update Staging Table Structure

The update staging table contains two columns for every primary-key column specified in the corresponding replication definition, one each for the column data before and after a change.

The update staging table also contains a column for each nonprimary-key column specified in the replication definition. To track whether changes have been made to data in these nonprimary-key columns, the update staging table contains one or more bitmap columns. Each bitmap column is of type `int` and can therefore track 32 non-primary key columns. A value of 1 constitutes a dirty bit, indicating that data has changed in the column corresponding to that bit position.

Note: The before-change and bitmap columns of the update staging table are not visible in the SQL Transformation project in Sybase IQ InfoPrimer.

Transformation Stored Procedures

For every primary database table that is staged, there should be a corresponding transformation stored procedure in the replicate Sybase IQ database. Replication Server executes these stored procedures against the staging tables, and the results are written to the base tables.

You must specify the transformations to be performed by these stored procedures in a Sybase IQ InfoPrimer SQL Transformation project and deploy the stored procedures to the replicate Sybase IQ database.

If you attempt to use stored procedures that do not exist in the replicate Sybase IQ database, or if a stored procedure fails to execute properly, the replicate database connection will be suspended.

Note: To ensure that you can see all the tables involved in a SQL Transformation project, do not select a schema in the project properties for the SQL Transformation project until you are ready to deploy your stored procedures to the replicate Sybase IQ database.

Parameters

Replication Server uses the **stage_operations** and **dsi_stage_all_ops** parameters to control table staging.

stage_operations

Set the **stage_operations** parameter of the **create connection** or **alter connection** command to have Replication Server write operations to staging tables for the specified connection.

You can configure staging for the replicate database connection. For example:

```
create connection to SYDNEY_IQ_RS.iq_db
 using profile rs_ase_to_iq;standard
 set username pubs2_maint
 set password pubs2_maint_pw
 set stage_operations to "on"
```

To selectively enable or disable staging for individual tables, use the **stage_operations** parameter of the **alter connection** command in reference to a specific replicate table. For example:

```
alter connection to SYDNEY_IQ_RS.iq_db
for replicate table named lineitem_5
set stage_operations to "off"
```

Here, Replication Server will not stage operations for the `lineitem_5` table but will instead replicate operations as normal.

Note: The **stage_operations** parameter can only be set for a connection to a Sybase IQ replicate (where the **dsi_dataserver_make** parameter is set to iq). The **dsi_dataserver_make** connection parameter is set appropriately when you use the Sybase IQ connection profile to create the connection.

dsi_compile_enable

If **stage_operations** is set to on, Replication Server ignores the setting of **dsi_compile_enable** and enables RTL for the connection. Operations are compiled, as when **dsi_compile_enable** is set to on, and then staged.

dsi_stage_all_ops

Use the **dsi_stage_all_ops** parameter of the **alter connection** command to prevent operation compilation for specified tables.

If table history must be preserved, as in the case of slowly changing dimension (SCD) tables, set **dsi_stage_all_ops** to on. For example:

```
alter connection to SYDNEY_IQ_RS.iq_db
for replicate table named lineitem_5
set dsi_stage_all_ops to "on"
```

Replication Server Components

Replication Server requires some additional components to support the integration with Sybase IQ InfoPrimer.

The rs_status Table

The `rs_status` table stores information about the progress of materialization.

Column	Datatype	Description
`schema`	`varchar (255)`	Owner of table being materialized
`table-name`	`varchar (255)`	Name of table being materialized

Column	Datatype	Description
action	varchar (1)	• I – initial load • A – autocorrection phase • R – replication
start-time	time-stamp	Time action was started
endtime	time-stamp	Time action completed
status	varchar (1)	• P – action in progress • X – execution complete • E – execution error
pid	int	Reserved

For example, if autocorrection is in progress for `my_table`, `rs_status` contains a row like this:

```
schema tablename action starttime                endtime status pid
------ --------- ------ ----------------------- ------- ------ ---
sys    my_table  A      2011-07-11 19:11:25.531         P
```

If autocorrection is complete for `my_table`, `rs_status` contains a row like this:

```
schema tablename action starttime
------ --------- ------ -----------------------
sys    my_table  A      2011-07-11 19:11:25.531
```

```
endtime                 status pid
----------------------- ------ ---
2011-07-11 19:12:14.326 X
```

There is no automatic cleanup of `rs_status` data. Before you attempt to rematerialize a table, you must delete its corresponding row from `rs_status`:

```
delete rs_status where tablename=tablename and schema=owner
```

Autocorrection Functions

Replication Server uses the **rs_autoc_on**, **rs_autoc_off**, and **rs_autoc_ignore** functions to update the `rs_status` table.

rs_autoc_on

Updates the `rs_status` table to indicate that autocorrection has been set to on.

Replication Server invokes **rs_autoc_on** when the Data Server Interface (DSI) encounters an **autocorrection on** record in the primary database log.

Examples

- **Example –** Creates an **rs_autoc_on** function string for **rs_iq_function_class**.

```
create function string rs_autoc_on
 for rs_iq_function_class
 output language
  'insert into rs_status (schema, tablename, action, starttime,
status) values
    (?rs_repl_objowner!sys?,
     ?rs_deliver_as_name!sys?,
     "A",
     current timestamp,
     "P");
  commit'
```

Usage

- The **rs_autoc_on** function has function-string-class scope.
- Replication Server creates an initial **rs_autoc_on** function string during installation.
- **rs_autoc_on** uses the *rs_deliver_as_name* system-defined variable, which indicates the table in the replicate database affected by autocorrection.
- **rs_autoc_on** uses the *rs_repl_objowner* system-defined variable, which indicates the owner of the table in the replicate database affected by autocorrection. If no owner is specified, **rs_repl_objowner** contains a single space.

rs_autoc_off

Updates the `rs_status` table to indicate that autocorrection has been set to off.

Replication Server invokes **rs_autoc_off** when it encounters an **autocorrection off** record in the primary database log.

Examples

- **Example –** Creates an **rs_autoc_off** function string for **rs_iq_function_class**.

```
create function string rs_autoc_off
 for rs_iq_function_class
 output language
  'update rs_status
   set endtime = current timestamp,
   status = "X" where schema = ?rs_repl_objowner!sys?
   and tablename = ?rs_deliver_as_name!sys?
   and action = "A" and endtime is null;
   insert into rs_status (schema, tablename, action, starttime,
status) values
   (?rs_repl_objowner!sys?,
    ?rs_deliver_as_name!sys?,
    "R",
    current timestamp,
```

```
    "P");
   commit'
```

Usage

- The **rs_autoc_off** function has function-string-class scope.
- Replication Server creates an initial **rs_autoc_off** function string during installation.
- **rs_autoc_off** uses the *rs_deliver_as_name* system-defined variable, which indicates the table in the replicate database affected by autocorrection.
- **rs_autoc_off** uses the *rs_repl_objowner* system-defined variable, which indicates the owner of the table in the replicate database affected by autocorrection. If no owner is specified, **rs_repl_objowner** contains a single space.

rs_autoc_ignore

Updates the `rs_status` table to indicate that autocorrection has failed and that DML is ignored for a table.

Replication Server invokes **rs_autoc_ignore** when a primary-key update is made during autocorrection.

Examples

- **Example –** Creates an **rs_autoc_ignore** function string for **rs_iq_function_class**.

```
create function string rs_autoc_ignore
 for rs_iq_function_class
 output language
  'update rs_status
   set endtime = current timestamp,
   status = 'E' where schema = ?rs_repl_objowner!sys?
   and tablename = ?rs_deliver_as_name!sys?
   and action = 'A' and endtime is null;
   commit'
```

Usage

- The **rs_autoc_ignore** function has function-string-class scope.
- Replication Server creates an initial **rs_autoc_ignore** function string during installation.
- **rs_autoc_ignore** uses the *rs_deliver_as_name* system-defined variable, which indicates the table in the replicate database affected by autocorrection.
- **rs_autoc_ignore** uses the *rs_repl_objowner* system-defined variable, which indicates the owner of the table in the replicate database affected by autocorrection. If no owner is specified, **rs_repl_objowner** contains a single space.

System Variables

The **rs_autoc_on** and **rs_autoc_off** functions use two system variables when updating the `rs_status` table.

- *rs_deliver_as_name* – specifies the name of the replicate table affected by autocorrection.
- *rs_repl_objowner* – specifies the owner of the replicate table affected by autocorrection.

Default Datatype Translation

Sybase IQ supports all Adaptive Server datatypes in their native formats, so no Adaptive Server-to-Sybase IQ datatype translation is required.

Unsupported Features

The integration of Replication Server with Sybase IQ InfoPrimer is limited to certain features and platforms.

The integration of Replication Server with Sybase IQ InfoPrimer does not support:

- any replicate database other than Sybase IQ
- any primary database other than Adaptive Server
- replicated stored procedures
- custom function strings
- any pre-staging operation transformations other than those provided by RTL
- any transformations following those performed by the transformation stored procedures in the replicate Sybase IQ database

New Features in Replication Server 15.6

Replication Server® 15.6 includes performance, usability, process, and database support enhancements.

Replication Server Licensing

Replication Server 15.6 introduces sub-capacity licensing and changes to product editions.

Subcapacity Licensing

Sybase now offers subcapacity licensing options for Replication Server allowing you to license a Sybase product on a subset of the CPUs available on a physical machine.

See *Installation Guide > Before You Begin > Preinstallation Tasks > Obtaining a License > Sub-capacity Licensing*.

Replication Server 15.6 Product Editions and Licenses

Replication Server 15.6 is released as two separate product editions—Enterprise Edition (EE) and Real-Time Loading Edition (RTLE) — that comprise different base and optional features, and which require separate licences.

Changes in RTLE for Replication Server 15.6

You can replicate from Oracle to Sybase IQ with Replication Server 15.6. Besides Replication Server, RTLE includes Replication Agent for Oracle to allow you to connect to an Oracle primary data server. The documentation for RTLE includes Replication Server Options product documentation in addition to Replication Server product documentation.

Table 6. Enterprise Edition Features and Licenses

Feature Type	Features	Description	License
Base	Replication Server	Replication Server features, excluding Advanced Services Option, ExpressConnect for Oracle, and real-time loading.	REP_SERVER
Optional	Advanced Services Option	Replication Server performance enhancements.	REP_HVAR_ASE

Feature Type	Features	Description	License
	ExpressConnect for Oracle	Provides Replication Server with the capability to connect directly to Oracle. See the Replication Server Options 15.5 product documentation.	REP_EC_ORA

Table 7. Real-Time Loading Edition Features and Licenses

Feature Type	Features	Description	License
Base	Replication Server	Replication Server features, excluding Advanced Services Option, ExpressConnect for Oracle, and real-time loading.	REP_SERVER
	Real-Time Loading (RTL)	Allows replication to Sybase IQ from Adaptive Server and Oracle. **Note:** You cannot use the Real-Time Loading Edition to replicate to Adaptive Server or Oracle.	REP_RTL_IQ
	Advanced Services Option	Replication Server performance enhancements.	REP_HVAR_ASE
	Replication Agent for Oracle.	Includes Replication Agent for Oracle to connect to Oracle as a primary data server.	RTLE includes a license for Replication Server Options.
Optional	None		

Obtain a License

Obtain valid SySAM licenses before you install Replication Server.

Sybase Software Asset Management (SySAM) performs license administration and asset management tasks for Sybase products. See *Installation Guide > Before You Begin > Preinstallation Tasks > Obtaining a License*.

Replication from Oracle to Sybase IQ Using Real-Time Loading

You can use real-time loading (RTL) to replicate from Oracle to Sybase IQ in the Real-Time Loading Edition (RTLE) of Replication Server 15.6.

License

Replication to Sybase IQ using RTL is available in the Real-Time Loading Edition product edition.

Database and Platform Support

- Sybase IQ – you can use real-time loading to replicate into Sybase IQ version 12.7 ESD #3 and later. See *Replication Server Release Bulletin > Product Compatibility > Replication Server Interoperability* for the latest supported Sybase IQ versions and platforms.
- Oracle – Replication Server 15.6 supports replication to Sybase IQ from Oracle 10g and 11g. See "Compatible products" in the *Replication Agent Release Bulletin for Linux, Microsoft Windows, and UNIX*.

Real-Time Loading Solution

RTL groups together as many compilable transactions as possible, compiles the transactions in the group into a net change, then uses the bulk interface in the replicate database to apply the net change to the replicate database.

When replicating into Sybase IQ replicate databases with identical database schema, RTL uses:

- Compilation – rearranges replicate data by each table, and each **insert**, **update**, and **delete** operation, and compiling the operations into net-row operations.
- Bulk apply – applies the net result of the compilation operations in bulk using the most efficient bulk interface for the net result. Replication Server uses an in-memory net-change database to store the net row changes that are then applied to the replicate database.

RTL improves performance for replication to Sybase IQ compared to the continuous replication mode and a staging solution for example, by using:

- Reduced number of external components – reduced maintenance costs and overhead, since there is no requirement for the staging database.
- Reduced latency – no overhead from the staging solution and with replication directly into Sybase IQ.
- Improved usability – the RTL configuration does not require any of: function-string mapping, DSI suspend and resume, data population from staging database to Sybase IQ, scheduling activities for the staging solution.

- Compilation and bulk apply – instead of sending every logged operation, RTL compilation removes the intermediate **insert**, **update**, or **delete** operations in a group of operations and sends only the final compiled state of a replicated transaction. Depending on the transaction profile, this generally means that Replication Server sends a smaller number of commands to Sybase IQ to process.

  Sybase IQ provides a bulk interface that improves **insert** operation performance compared with the SQL language mode operation. RTL takes advantage of the Sybase IQ bulk interface to improve performance for **insert** as well as **update** and **delete** operations.

  As Replication Server compiles and combines a larger number of transactions into a group, bulk operation processing improves; therefore, replication throughput and performance also improves. You can adjust group sizes to control the amount of data that is grouped together for bulk apply.

RTL Compilation and Bulk Apply

During compilation, RTL rearranges data to be replicated by clustering the data together based on each table, and each **insert**, **update**, and **delete** operation, and then compiling the operations into net row operations.

RTL distinguishes different data rows by the primary key defined in a replication definition. If there is no replication definition, all columns except for `text` and `image` columns are regarded as primary keys.

For the combinations of operations found in normal replication environments, and given a table and row with identical primary keys, RTL follows these compilation rules for operations:

- An **insert** followed by a **delete** results in no operation.
- A **delete** followed by an **insert** results in no reduction.
- An **update** followed by a **delete** results in a **delete**.
- An **insert** followed by an **update** results in an **insert** where the two operations are reduced to a final single operation that contains the results of the first operation, overwritten by any differences in the second operation.
- An **update** followed by another **update** results in an **update** where the two operations are reduced to a final single operation that contains the results of the first operation, overwritten by any differences in the second operation.

Other combinations of operations result in invalid compilation states.

Example 1

This is an example of log-order, row-by-row changes. In this example, `T` is a table created earlier by the command: **create table** `T(k int , c int)`

```
1. insert T values (1, 10)
2. update T set c = 11 where k = 1
3. delete T where k = 1
4. insert T values (1, 12)
5. delete T where k =1
6. insert T values (1, 13)
```

With RTL, the **insert** in 1 and the **update** in 2 can be converted to **insert** `T values (1, 11)`. The converted **insert** and the **delete** in 3 cancel each other and can be removed. The **insert** in 4 and the **delete** in 5 can be removed. The final compiled RTL operation is the last **insert** in 6:

```
insert T values (1, 13)
```

Example 2

In another example of log-order, row-by-row changes:

```
1. update T set c = 14 where k = 1
2. update T set c = 15 where k = 1
3. update T set c = 16 where k = 1
```

With RTL, the **update** in 1 and 2 can be reduced to the **update** in 2. The updates in 2 and 3 can be reduced to the single **update** in 3 which is the net-row change of k = 1.

Replication Server uses uses an **insert**, **delete**, and **update** table in an in-memory net-change database to store the net-row changes it applies to the replicate database. Net-row changes are sorted by replicate table and by type of operation—**insert**, **update**, or **delete**—and are then ready for bulk interface.

RTL directly loads **insert** operations into the replicate table. Since Sybase IQ does not support bulk **update** and **delete**, RTL loads **update** and **delete** operations into temporary worktables that RTL creates inside the IQ temporary store. RTL then performs **join-update** or **join-delete** operations with the replicate tables to achieve the final result. The worktables are created and dropped dynamically.

In Example 2, where compilation results in `update T set c = 16 where k = 1`:

1. RTL creates the *#rs_uT(k int, c int)* worktable.
2. RTL performs an **insert** into the worktable:
   ```
   insert into #rs_uT(k, c) location ‘idemo.db’ {select * from rs_uT}
   ```
3. RTL performs the **join-update**:
   ```
   update T set T.c=#rs_uT.c from T,#rs_uT where T.k=#rs_uT.k
   ```

As RTL compiles and combines a larger number of transactions into a group, bulk operation processing improves; therefore, replication throughput and performance also improves. You can control the amount of data that RTL groups together for bulk apply by adjusting RTL sizes with configuration parameters.

There is no data loss, although RTL does not apply row changes in the same order in which the changes are logged:

- For different data rows, the order in which row changes are applied does not affect the result.
- In the same row, applying **delete** before **insert** after compilation maintains consistency.

See also

- *RTL Configuration* on page 60

RTL Processing and Limitations

RTL applies only the net-row changes of a transaction while maintaining the original commit order, and guarantees transactional consistency even as it skips intermediate row changes.

This has several implications:

- **Insert** triggers do not fire, as the RTL process performs a bulk load of net new rows directly into the table. **Update** and **delete** triggers continue to fire when Replication Server applies the net results of compilation to the replicate database. However, row modifications that Replication Server compiles, and that are no longer in the net results, are invisible to the triggers. Triggers can detect only the final row images.

  Suppose you use Replication Server to audit user updates using a `last_update_user` column in a table schema with a trigger logic that associates a user to any column in the table modified by the user. If userA modifies `colA` and `colC` in the table and then userB modifies `colB` and `colD`, when the trigger fires, the trigger logic can detect only the last user who modified the table, and therefore the trigger logic associates userB as the user that modified all four columns. If you define triggers that contain similar logic where every individual row modification must be detected, you may have to disable RTL compilation for that table.
- RTL does not apply row changes in the same order in which the changes are logged. To apply changes to a replicated table in log order, disable RTL compilation for that table.
- If there are referential constraints on replicate tables, you must specify the referential constraints in replication definitions. To avoid constraint errors, RTL loads tables according to replication definitions.
- RTL for replication into Sybase IQ does not support customized function strings or any parallel DSI serialization methods, except for the default **wait_for_commit** method. RTL treats customized function strings as noncompilable commands.
- RTL does not compile some types of commands called noncompilable commands and some types of tables called noncompilable tables. Replication Server reverts to log-order, row-by-row continuous replication when it encounters these commands, transactions, or tables:
  - Noncompilable commands – stored procedures, SQL statements, system transactions, and Replication Server internal markers.
  - Noncompilable transactions – a transaction that contains noncompilable commands.
  - Noncompilable tables – tables with RTL disabled, with customized function strings, and with referential constraint relationships with tables that RTL cannot compile.
- RTL automatically changes primary-key updates to a **delete** followed by an **insert**.
- RTL ignores parameters such as **dsi_partition_rule** that can stop transaction grouping.
- If errors occur during RTL processing, Replication Server retries RTL with progressively smaller transaction groups until it identifies the transaction that failed RTL compilation, then applies the transaction using continuous replication.

  To realize the performance benefits of RTL, keep the primary and replicate databases synchronized to avoid the overhead of additional processing by Replication Server when

errors occur. You can set **dsi_command_convert** to **i2di,u2di** to synchronize the data although this also incurs a processing overhead. If the databases are synchronized, reset **dsi_command_convert** to **none**.

- RTL performs row-count validation to ensure replication integrity. The row-count validation is based on compilation. The expected row count is the number of rows remaining after compilation.
- When there are columns with `identity` datatype in a replication definition, Replication Server executes this Sybase IQ command in the replicate database:
  - **set temporary option identity_insert =** '*table_name*' before identity column inserts and updates
  - **set temporary option identity insert =** “” after identity column inserts and updates
- By default, Oracle performs minimal logging. Therefore, if you are using database replication definitions, either create table replication definitions or enable full logging to ensure the **update** command works correctly. If you choose to create table replication definitions, you can create the definitions in Replication Agent or Replication Server:
  - Replication Agent for Oracle – to automatically create replication definitions at Replication Server when one or more tables are marked for replication, either set **pdb_auto_create_repdefs** to **true** before you mark the table for replication or execute **rs_create_repdef** after you mark the table . See the *Replication Agent Reference Manual* in Replication Server Options.
  - Replication Server – execute **create replication definition** with the **send standby** clause to create the replication definition directly in Replication Server. See the *Replication Server Reference Manual*.

Sybase IQ Replicate Data Servers

The replicate Replication Server interacts directly with the replicate Sybase IQ data server by logging in to the Sybase IQ replicate database and applying the replicated transactions.

Replication Intrusions and Impacts on Sybase IQ

The only significant intrusions or impacts to the Sybase IQ replicate database are the system tables created in the Sybase IQ replicate database through the connection profile, and temporary tables created in the Sybase IQ replicate database to accommodate RTL bulk apply.

System Tables

The connection profile creates three tables in the Sybase IQ replicate database:

- `rs_threads` – used by Replication Server to detect deadlocks and to perform transaction serialization between parallel DSI threads. An entry is updated in this table each time a transaction is started and more than one DSI thread is defined for a connection.
- `rs_lastcommit` – contains information about replicated transactions applied to the replicate database. Each row in the `rs_lastcommit` table identifies the most recent committed transaction that was distributed to the replicate database from a primary database. Replication Server uses this information to ensure that all transactions are distributed.

The Replication Server **rs_get_lastcommit** function retrieves information about the most recent transaction committed in the replicate database. For non-ASE replicate databases, the **rs_get_lastcommit** function is replaced in the database-specific function-string class by the query required to access the `rs_lastcommit` table in the replicate database.

- `rs_ticket_history` – contains the execution results of Replication Server command **rs_ticket**. You can isssue the **rs_ticket** command for the primary database to measure the amount of time it takes for a command to move from the primary database to the replicate database. You can use this information to monitor Replication Server performance, module heartbeat, replication health, and table-level quiesce. The results of each execution of **rs_ticket** is stored in a single row of the `rs_ticket_history` table in the replicate database. You can query each row of the `rs_ticket_history` table to obtain results of individual **rs_ticket** executions, or to compare the results from different rows. Manually truncate the data in `rs_ticket_history` table if necessary.

Worktables

RTL creates temporary worktables inside the IQ temporary store of the Sybase IQ database to support RTL bulk apply.The worktables are created and dropped dynamically.

The amount of space required for the temporary tables in Sybase IQ depends on the amount of the data you expect to replicate to Sybase IQ. To adjust the Sybase IQ temporary database space to accommodate the temporary worktables, use the Sybase IQ **alter dbspace** command. See the Sybase IQ documentation for your version for more information. For example in Sybase IQ 15.0 and later:

```
ALTER DBSPACE dbspace-name ADD FILE FileHist3
'/History1/data/file3' SIZE 500MB
```

Replicate Database Connectivity for Sybase IQ

You do not need to use a database gateway when you use Sybase IQ as a replicate data server; the replicate Replication Server connects directly to the Sybase IQ replicate data server.

A Replication Server database connection name is made up of a data server name—**server_name**—and a database name—**db_name**. The replicate Replication Server looks for an `interfaces` file entry containing the Sybase IQ replicate database **server_name** specified in the database connection.

Use **dsedit** to make an entry in the Replication Server `interfaces` file to identify the host and port where the Sybase IQ replicate data server is listening. The `interfaces` file entry name must match the **server_name** portion of the Replication Server database connection. Restart Replication Server to activate the new entry in the Replication Server `interfaces` file. See *Replication Server Configuration Guide > Configure Replication Server and Add Databases Using* **rs_init** *> Configuring a New Replication Server > Editing the Interfaces File*.

Create an entry for the replicate Replication Server in the `interfaces` file of the Sybase IQ replicate server to allow Sybase IQ to connect to Replication Server and retrieve data when Replication Server sends an **INSERT ... LOCATION** statement to Sybase IQ.

Replication Server logs in to the Sybase IQ replicate data server using the **user_name** and **password** specified in the database connection. For Sybase IQ replicate databases, the **user_name** and **password** should be the maintenance user ID and password.

Sybase IQ Replicate Database Permissions

To apply transactions in a replicate database, Replication Server and Sybase IQ require a maintenance user ID.

Before replication can start, you must define the maintenance user ID at the Sybase IQ data server and grant authority to the ID to apply transactions in the replicate database. The maintenance user ID must have these permissions in the Sybase IQ replicate database:

- **RESOURCE** authority to create worktables and temporary indexes.
- **EXECUTE** permission to run the **sp_iqwho** stored procedure.
- **GRANT ALL** permission on all replicate tables.
- **UPDATE** authority on all replicate tables and **EXECUTE** authority on all replicate stored procedures.

Granting Authority to a Maintenance User ID

Grant DBA and RESOURCE authority if you are starting with a simple setup or are testing replication to Sybase IQ.

1. Use the Sybase IQ `rssetup.sql` sample script to create the maintenance user for Sybase IQ with relevant privileges.

   Warning! If there is already a maintenance user ID, the script resets the password to the default password.

```
grant connect to dbmaint identified by dbmaint
grant DBA to dbmaint
grant membership in group rs_systabgroup to dbmaint

-- Create a user for REPSRV to extract -- materialization data,
etc.
-- Give sa user access to any replicated tables
-- Give sa user access to REPSRV schema
grant connect to sa identified by sysadmin
grant DBA to sa
grant membership in group rs_systabgroup to sa

-- Allow sa and dbmaint to reference replicated tables created by
DBA
grant group to DBA
grant membership in group DBA to dbmaint
grant membership in group DBA to sa
go
```

This script is in the `scripts` directory within the Sybase IQ installation directory. For example, on UNIX platforms in:

- Sybase IQ versions earlier than 15.0 – `/$ASDIR/scripts`
- Sybase IQ 15.0 and later – `/$IQDIR15/scripts`

See the *Sybase IQ Installation and Configuration Guide* for locations of directories.

2. Verify that the Sybase IQ database is compatible with Transact-SQL® (For IQ DBA).

   See *Sybase IQ Reference: Statements and Options > Database Options > Transact-SQL Compatibility Options* and *Sybase IQ Reference: Building Blocks, Tables, and Procedures > Compatibility with Other Sybase Databases*.

3. Grant the appropriate permissions to all tables and stored procedures that are to participate in replication.

Sybase IQ Replicate Database Configuration Issues

Learn about the configuration issues for the Sybase IQ server.

Replication Server Installation

Replication Server automatically installs the required connection profile, which provides function strings and classes to support replication into Sybase IQ.

Connection Profiles

Connection profiles allow you to configure your connection with a pre-defined set of properties by setting the function-string class and error class, installing the user-defined datatypes (UDD) and translations for Sybase IQ, and creating the tables required for replication in the replicate Sybase IQ database.

The **rs_oracle_to_iq** connection profile is part of the Replication Server installation package, and it is registered when you install Replication Server. The connection profile:

- Customizes function strings, error classes, and user-defined datatypes. The function string replaces several default Replication Server function strings with custom function strings designed to communicate with a Sybase IQ data server and access the tables and procedures. These function strings are added to the Replication Server default **rs_iq_function_class**. RTL treats customized function strings as non-compilable commands.
- Customizes class-level datatype translations. Class-level translations identify primary datatypes and the replicate datatypes the data should be translated into. Class-level translation is supplied for the Sybase IQ replicate database by the **rs_oracle_to_iq** connection profile, which translates Oracle datatypes to Sybase IQ datatypes.
- Creates the `rs_threads`, `rs_lastcommit`, and `rs_ticket_history` tables in the Sybase IQ replicate database.
- Sets the default function-string class and error class connection properties to configure the connection to Sybase IQ:

```
set error class rs_iq_error_class
set function string rs_iq_function_class
```

Creating the Connection to Sybase IQ

Set up the connection to the replicate Sybase IQ database.

1. Use **create connection** with the **using profile** clause and the relavant connection profile, and specify your replicate Sybase IQ data server and database.
   For example to create a connection from an Oracle primary data server:

   ```
   create connection to IQSRVR.iqdb
   using profile rs_oracle_to_iq;standard
   set username to dbmaint
   set password to dbmaint
   go
   ```

   You can create multiple replication paths to the Sybase IQ database to distribute replication loads. Use a unique maintenance user ID for each path.
2. Use **admin who** to verify that Replication Server connects successfully to Sybase IQ.

Enable RTL

After you have granted the relevant permissions and connected to the replicate Sybase IQ database, you can enable and configure RTL for replication to Sybase IQ.

Use **dsi_compile_enable** to enable RTL for the connection. If you set **dsi_compile_enable** off, Replication Server uses continuous log-order, row-by-row replication mode. For example, set **dsi_compile_enable** off for an affected table if replicating net-row changes causes problems, such as when there is a trigger on the table that requires all operations on the table to be replicated in log order, and therefore compilation is not allowed.

Note: When you set **dsi_compile_enable** on, Replication Server disables **dsi_cmd_prefetch** and **dsi_num_large_xact_threads**.

To enable and configure RTL at the database level to affect only the specified database, enter:

```
alter connection to IQ_data_server.iq_database
set dsi_compile_enable to ‘on’
go
```

You can also enable and configure RTL at the server or table levels.

- Server level – affects all database connections to Replication Server:

  ```
  configure replication server
  set dsi_compile_enable to ‘on’
  ```
- Table level – affects only the replicate tables you specify. If you specify a parameter at both the table level and database level, the table-level parameter takes precedence over the database-level parameter. If you do not specify a table-level parameter, the setting for the parameter applies at the database level. To set a parameter for a table, use **alter connection** and the **for replicate table named** clause, for example:

```
alter connection to IQ_data_server.iq_database
for replicate table named dbo.table_name
set dsi_compile_enable to 'on'
```

Using the **for replicate table name** clause alters connection configuration at the table level. The configuration changes apply to replicate data from all the subscriptions and all the replication definitions of the tables you specify.

Note: For table-level configuration, you can use only **alter connection**, as Replication Server does not support the **for** clause with **create connection**.

After you execute **dsi_compile_enable**, suspend and resume the connection to the replicate Sybase IQ database.

RTL Configuration

You can also use parameters to determine when to start and stop transaction grouping and compilation.

- **dsi_compile_max_cmds** – specifies, in number of commands, the maximum size of a group of transactions. When RTL reaches the maximum group size for the current group that it is compiling, RTL starts a new group.

  If there is no more data to read, and even if the group does not reach the maximum number of commands, RTL completes grouping the current set of transactions into the current group. The default is 10,000 commands.
- **dsi_bulk_threshold** – specifies the number of net row change commands after compilation has occurred on a table for a command type, that when reached, triggers Replication Server to use bulk copy-in on that table for the same command type. The default is 20 net row change commands.
- **dsi_command_convert** – specifies how to convert a replicate command. A combination of these operations specifies the type of conversion:
  - **d** – delete
  - **i** – insert
  - **u** – update
  - **t** – truncate
  - **none** – no operation

  Combinations of operations for **dsi_command_convert** include **i2none**, **u2none**, **d2none**, **i2di**, **t2none**, and **u2di**. The operation before conversion precedes the "2" and the operations after conversion are after the "2". For example:
  - **d2none** – do not replicate the **delete** command. With this option, you do not need to customize the **rs_delete** function string if you do not want to replicate **delete** operations.
  - **i2di**,**u2di** – convert both **insert** and **update** to **delete** followed by **insert**, which is equivalent to an autocorrection. If you disable row count validation by setting **dsi_row_count _validation** off, Sybase recommends that you set

dsi_command_convert to **i2di,u2di** to avoid duplicate key errors and allow autosynschronization of databases during replication.

- **t2none** – do not replicate the **truncate table** command.

The default for **dsi_command_convert** is **none**, which means there is no command conversion.

- **dsi_compile_retry_threshold** – specifies a threshold value for the number of commands in a group. Replication Server 15.6 includes the **dsi_compile_retry_threshold** parameter as part of enhancements to the retry mechanism.

RTL automatically sets the Sybase-recommended default values for **dsi_compile_max_cmds**, **dsi_bulk_threshold**, **dsi_command_convert**, and **dsi_compile_retry_threshold**. However, you can specify your own values to tune performance in your replication environment:

- ```
  alter connection to IQSRVR.iqdb
  set dsi_compile_max_cmds to ‘50000’
  go
  ```
- ```
  alter connection to IQSRVR.iqdb
  set dsi_bulk_threshold to ‘15’
  go
  ```
- ```
  alter connection to IQSRVR.iqdb
  set dsi_command_convert to ‘i2di,u2di’
  go
  ```
- ```
  alter connection to IQSRVR.iqdb
  set dsi_compile_retry_threshold to '200'
  go
  ```

Note: You must execute a separate **alter connection** command for each parameter you want to change. Do not enter more than one parameter after entering **alter connection**.

See *Replication Server Reference Manual > Replication Server Commands >* ***alter connection*** for full descriptions of the parameters.

See also

- *Enhanced Retry Mechanism in HVAR and RTL* on page 70
- *Controlling Row Count Validation* on page 92

System Table Support in Replication Server

Replication Server uses the `rs_tbconfig` table to store support table-level configuration parameters, and the `ref_objowner` and `ref_objname` columns in the `rs_columns` table to support referential constraints.

See *Replication Server Reference Manual > Replication Server System Tables* for full table descriptions.

Scenario for Replication to Sybase IQ

Use this scenario to learn how to set up and test replication to Sybase IQ using RTL.

The Oracle database administrator (Oracle DBA), the Sybase IQ database administrator (IQ DBA), and you, as the replication system administrator (RSA), must prepare Oracle, Replication Server, and Sybase IQ for replication and set up the connection to the Sybase IQ database.

In this scenario, `dbo` is the table owner of the *testtab* table in the *pdb1* database of the ORA_DS primary Oracle server. *c1*, *c2*, and *c3* are columns in *testtab* with *int*, *int*, and *char(10)* datatypes respectively, and IQSRVR is the replicate Sybase IQ data server containing the *iqdb* database.

Creating Interfaces File Entries

Create an entry in the `interfaces` files of the replicate Replication Server and the Sybase IQ data server for each other.

1. Create an entry for the replicate Replication Server in the `interfaces` file (`sql.ini` file in Windows) of the Sybase IQ data server.

   Note: Create an `interfaces` file for the Sybase IQ data server if the file is not in the $SYBASE directory (%SYBASE% directory in Windows) that Sybase IQ is using.

2. Create an entry for the Sybase IQ data server in the `interfaces` file of the replicate Replication Server.

   If you are creating connections to different Sybase IQ multiplex nodes, create entries for each of the affected nodes in the `interfaces` file of the replicate Replication Server.

See also

- *Replicate Database Connectivity for Sybase IQ* on page 56

Creating Test Tables

Create a test table in the primary and replicate databases, and grant maintenance user permissions to it to test that replication works.

1. In the Oracle primary database `pdb1` in the Oracle data server, create a table named `testtab` with three columns: `c1 integer`, `c2 integer` and `c3 char(10)`. See Oracle documentation for syntax.

2. In the replicate database `iqdb` in the Sybase IQ IQSRVR data server, enter:

```
use iqdb
go
create table dbo.testtab(c1 int primary key, c2 int,
c3 char(10))
go
```

```
grant all on dbo.testtab to public
go
```

Creating the Connection to the Primary and Replicate Databases

Create the primary and replicate database connections.

1. Create the connection to the primary Oracle database. See the *Heterogeneous Replication Guide* and the Replication Server Options product documentation.
2. Create the connection to the replicate Sybase IQ database.

   Note: You cannot use **rs_init** to create the connection from Oracle to Sybase IQ.

   This example uses the *iqdb* database in the IQSRVR data server, and the default dbmaint Sybase IQ maintenance user.

   ```
   create connection to IQSRVR.iqdb
   using profile rs_oracle_to_iq;standard
   set username to dbmaint
   set password to dbmaint
   go
   ```

   If the command is successful, you see:

   ```
   Connection to ‘IQSRVR.iqdb’ is created.
   ```

   See "**create connection with using profile clause**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual*.
3. Verify that the connection is running:

   ```
   admin who
   go
   ```

   If the connection is running, you see:

   ```
   Spid   Name       State               Info
   ----   --------   ----------------    -----------------
    63    DSI EXEC   Awaiting Command    103(1)IQSRVR.iqdb
    62    DSI        Awaiting Message    103 IQSRVR.iqdb
    35    SQM        Awaiting Message    103:0 IQSRVR.iqdb
   ```

Enabling RTL

Enable RTL at the database level.

1. To enable and configure RTL at the database level to affect only the specified database, enter:

   ```
   alter connection to IQSRVR.iqdb
   set dsi_compile_enable to ‘on’
   go
   ```
2. Suspend and resume the connection to the replicate Sybase IQ database to enable the change to the connection:

   ```
   suspend connection to IQSRVR.iqdb
   go
   ```

```
resume connection to IQSRVR.iqdb
go
```

Marking Tables to Prepare for Replication Testing

Mark tables in the primary Oracle database that you want to replicate to the Sybase IQ database.

In these examples, `dbo` is the table owner of `testtab` in the `pdb1` database of the ORA_DS primary Oracle data server. `c1`, `c2`, and `c3` are columns in `testtab` with `int`, `int`, and `char(10)` datatypes, respectively.

1. Insert data rows into `testtab` in Oracle for testing replication and verify the inserts are successful.
2. Mark `testtab` for replication with the **pdb_setreptable** Replication Agent command.

   See "Marking tables in the primary database," in Chapter 2, "Setting Up and Configuring Replication Agent" in the *Replication Agent Administration Guide* in the Replication Server Options 15.5 documentation.

Creating Replication Definitions and Subscriptions

Create replication definitions and subscriptions for the tables marked for replication to Sybase IQ after you enable and configure RTL.

1. Create the *repdef_testtab* replication definition and add any required referential constraint clauses to the replication definition to support RTL:

   ```
   create replication definition repdef_testtab
   with primary at ORA_DS.pdb1
   with primary table named ‘TESTTAB’
   with replicate table named dbo.‘testtab’
   (C1 as c1 int, C2 as c2 int, C3 as c3 char(10))
   primary key(C1)
   go
   ```

   Note: The default character case of Oracle is all upper case for object names. You can convert object names from upper to lower case in the replication definition, as shown in the example in step 1, or by using the **ltl_character_case** Replication Agent for Oracle configuration parameter. See "**ltl_character_case**" in Chapter 2, "Configuration Parameters" in the *Replication Agent Reference Manual* in Replication Server Options.

2. Create subscriptions to match each of the table and stored procedure replication definitions:

   ```
   create subscription sub_testtab for repdef_testtab
   with replicate at IQSRVR.iqdb
   go
   ```

3. Verify that `testtab` is materialized by logging in to Sybase IQ and executing:

   ```
   select * from dbo.testtab
   go
   ```

   If materialization is successful, you see:

```
c1          c2          c3
---------   ---------   ---------
1           1           testrow 1
2           2           testrow 2
3           3           testrow 3
(3 rows affected)
```

Verifying That RTL Works

Learn how to check that RTL works.

1. Log in to the primary Oracle data server and execute some operations, such as inserting new rows into `testtab`.
2. Log in to Sybase IQ and verify that the changes to `testtab` have replicated to the Sybase IQ database:

   ```
   select * from dbo.testtab
   go
   ```

   If replication is successful, you see:

   ```
   c1          c2          c3
   ---------   ---------   ---------
   1           1           testrow 1
   2           2           testrow 2
   3           3           testrow 3
   4           4           testrow 4
   5           5           testrow 5
   6           6           testrow 6
   (6 rows affected)
   ```

Tables with Referential Constraints

You can use a replication definition to specify tables that have referential constraints, such as a foreign key and other check constraints, so that RTL is aware of these tables.

Usually, the referencing table contains referential constraints for a referenced table within the same primary database. However, RTL extends referential constraints support to referenced tables from multiple primary databases.

You can specify the referencing table in a replication definition for each primary database. However, if multiple referential constraints conflict with each other, Replication Server randomly selects one.

See also

- *RTL Processing and Limitations* on page 54

Replication Definitions Creation and Alteration

Use the **create replication definition** command with the **references** parameter to specify the table with referential constraints.

```
create replication definition
    ...
```

```
   (column_name [as replicate_column_name]
       ...
       [map to published_datatype]] [quoted]
       [references [table_owner.]table_name [(column_name)]] …)
   ....]
```

Use the **alter replication definition** command with the **references** parameter to add or change a referencing table. Use the **null** option to drop a reference.

alter replication definition

```
   .....
add column_name [as replicate_column_name]
[map to published_datatype] [quoted]
[references [table_owner.]table_name [(column_name)]
...
| alter columns with column_name references
{[table_owner.]table_name [(column_name)] | NULL}
[, column_name references {[table_owner.]table_name [(column_name)]
| NULL}
     ...
```

For both **alter replication definition** and **create replication definition** with the **reference** clause, Replication Server:

- Treats the **reference** clause as a column property. Each column can reference only one table.
- Does not process the column name you provide in the **column_name** parameter within the **reference** clause.
- Does not allow referential constraints with cyclical references. For example, the original referenced table cannot have a referential constraint to the original referencing table.

During replication processing, RTL loads:

- Inserts to the referenced tables before the referencing table you specify in the replication definition.
- Deletes to the referenced tables after the table you specify in the replication definition.

In some cases, updates to both tables fail because of conflicts. To prevent RTL from retrying replication processing, and thus decreasing performance, you can:

- Stop replication updates by setting **dsi_command_convert** to "**u2di,**" which converts updates to deletes and inserts.
- Turn off **dsi_compile_enable** to avoid compiling the affected tables.

RTL cannot compile tables with customized function strings, and tables that have referential constraints to an existing table that it cannot compile. By marking out these tables, RTL optimizes replication processing by avoiding transaction retries due to referential constraint errors.

Display RTL Information

You can display information on configuration parameter properties and table references.

Display Configuration Parameter Properties

Use **admin config** to view information about database-level and table-level configuration parameters as shown in the examples.

- Database-level:
  - To display all database-level configuration parameters for the connection to the `nydb1` database of the NY_DS data server (`NY_DS.nydb1`), enter:

    ```
    admin config, “connection”, NY_DS, nydb1
    ```
  - To verify that **dsi_compile_enable** is **on** for the connection to `NY_DS.nydb1`, enter:

    ```
    admin config, “connection”, NY_DS, nydb1,dsi_compile_enable
    ```
  - To display all the database-level configuration parameters that have "enable" as part of the name, such as **dsi_compile_enable**, enter:

    ```
    admin config, “connection”, NY_DS, nydb1,"enable"
    ```

    Note: You must enclose "enable" in quotes because it is a reserved word in Replication Server. See *Replication Server Reference Manual > Topics > Reserved Words*.
- Table-level:
  To display all configuration parameters after using **dsi_command_convert** to set **d2none** on the `tb1` table in the `nydb1` database of the NY_DS data server, enter:

  ```
  admin config, “table”, NY_DS, nydb1
  ```

See *Replication Server Reference Manual > Replication Server Commands >* ***admin config***.

Display Table References

Use **rs_helprep**, which you can execute on the Replication Server System Database (RSSD), to view information about table references and RTL information.

To display information about the **authors_repdef** replication definition created using **create replication definition**, enter:

```
rs_helprep authors_repdef
```

See *Replication Server Reference Manual > RSSD Stored Procedures >* ***rs_helprep***.

Net-Change Database

Replication Server has a net-change database that acts as an in-memory repository for storing the net-row changes of a transaction, that is, the compiled transaction.

There is one net-change database instance for each transaction. Each replicate table can have up to three tracking tables within a net-change database. You can inspect the net-change database and the tables within the database to monitor RTL replication and troubleshoot problems.

Monitoring the Net-Change Database

Access net-change database instances and monitor a net-change database.

Use the **sysadmin cdb** command to monitor a net-change database.

See *Replication Server Reference Manual > Replication Server Commands >* ***sysadmin cdb***.

Mixed-Version Support and Backward Compatibility

RTL can replicate referential constraints specified in replication definitions only if the outbound route version is later than 15.5.

RTL works if the outbound route version is earlier than 15.5. However, no referential constraint information is available to a Replication Server with version 15.5 or later

Continuous replication is the default replication mode available to all supported versions of Replication Server. RTL is available only with Replication Server 15.5 and later.

Migrating from the Staging Solution to RTL

Migrate to the real-time loading solution if you are currently using the staging solution for replication to Sybase IQ.

The scenario assumes a replication topology where `pdb` is the primary Oracle database, PRS is the primary Replication Server, RRS is the replicate Replication Server, `staging_db` is the Oracle staging database, and `iqdb` is the replicate Sybase IQ database. The data flow in this scenario is:

```
pdb -----> PRS -----> RRS -----> staging_db -----> iqdb
```

Prerequisites

Before you migrate from the staging solution, you need to perform some tasks.

The tasks include:

1. You must upgrade both the primary and replicate Replication Servers to version 15.5 or later. See the *Replication Server Installation Guide* and *Replication Server Configuration Guide*.
2. Verify that no transactions flow into `pdb` and that the replication system is quiesced during migration:
   a. Stop Replication Agent for all primary databases and system databases by executing on Replication Server:
      ```
      suspend log transfer from all
      ```
   b. Stop RepAgent for the RSSD if you are using Adaptive Server as the RSSD:
      ```
      sp_stop_rep_agent rssd_name
      ```
   c. Verify that the Replication Server queues have drained and that Replication Server has been quiesced by executing:

```
admin quiesce_check
```

Retry with **admin quiesce_force_rsi** if Replication Server is not quiesced yet. If Replication Server is not quiesced, you may lose data.

3. Verify that `pdb` and `iqdb` are synchronized. You can resynchronize the databases by loading data to `iqdb` from the staging database after all the data is replicated to the staging database. If you do not a resynchronize the databases, you must purge and materialize `iqdb`.
4. Add an entry for the replicate Replication Server to the Sybase IQ interface file to allow the Sybase IQ server to connect to the replicate Replication Server and pull data.

Migrating to the Real-Time Loading Solution

Migrate from the staging solution to RTL.

1. Create a maintenance user in the replicate Sybase IQ data server, or you can use the existing maintenance user.
2. Create the connection to the replicate Sybase IQ database from the replicate Replication Server using the **rs_oracle_to_iq** connection profile and the maintenance user from step 1, such as *dbmaint*:

```
create connection to IQSRVR.iqdb
using profile rs_oracle_to_iq;standard
set username to dbmaint
set password to dbmaint
go
```

3. At the primary database, if a table owned by `dbo` is not marked as **owner_on**, you must enable **owner** for the table so that Sybase IQ can find the table since `dbo` does not exist in Sybase IQ:

```
pdb_setreptable testtab, mark, owner
go
```

4. Re-create the replication definition to include owner information since you have enabled **owner**.
5. If there are referential constraints between tables, you must alter the replication definition to define referential constraints so that Replication Server is aware of the referential constraints and can perform bulk apply in the proper order.
6. Enable RTL for the connection to the replicate database:

```
alter connection to iqserver_name.iqdb
set dsi_compile_enable to 'on'
```

After suspending and resuming the connection, the change in the connection takes effect.

7. Create subscriptions for each table. If the primary and replicate database are synchronized, include the **without materialization** clause in the subscription. Otherwise, you must enable autocorrection during materialization.

You can now replicate from Oracle directly to Sybase IQ.

Cleaning Up After Migration

Clean up the systems in the staging solution after enabling and configuring replication using RTL.

1. Drop subscriptions of the staging database.
2. Drop the replication definition that you are not using.
3. Drop connections to the staging database from the replicate Replication Server.
4. Terminate the environment for pulling data from the staging database to Sybase IQ.

Performance Enhancements

Replication Server 15.6 includes several performance enhancements.

Enhanced Retry Mechanism in HVAR and RTL

The enhanced retry mechanism improves replication performance for high-volume adaptive replication (HVAR) and real-time loading (RTL).

HVAR and RTL try to group as many compilable transactions as possible together, compile the transactions in the group into a net change, and then use the bulk interface in the replicate database to apply the net changes to the replicate database. HVAR and RTL invoke the retry mechanism when a replicate transaction resulting from HVAR and RTL processing fails. If transactions in a group fail, RTL and HVAR split the group into two smaller groups of equal size, and retry the compilation and bulk application on each group. The retry mechanism identifies the failed transaction, allows Replication Server to execute error action mapping, and applies all transactions preceding the failed transaction in case DSI shuts down.

The net-change database in HVAR and RTL acts as an in-memory repository for storing the net row changes of a transaction, that is, the compiled transaction. The content of the net-change database is an aggregation of commands from different primary transactions that HVAR and RTL are not applying in log order. Therefore, there is no means to identify a failed transaction without a retry mechanism. The retry mechanism splits a group and retries compilation and bulk application continuously as long as a transaction in the group fails. This continuous retry process can degrade performance.

The enhanced retry mechanism splits the group into three groups of equal size when HVAR or RTL encounter a group containing transactions that fail, enabling the mechanism to more efficiently identify the group containing the failed transaction.

In addition, with Replication Server 15.6, you can use the **dsi_compile_retry_threshold** parameter to specify a threshold value for the number of commands in a group. If the number of commands in a group containing failed transactions is smaller than the value of **dsi_compile_retry_threshold**, Replication Server does not retry processing the group, and saves processing time, thus improving performance. Instead, Replication Server switches to

continuous replication mode for the group. Continuous replication mode sends each logged change to the replicate database according to the primary database log order.

Use **configure replication server** to set **dsi_compile_retry_threshold** at the server level to affect all replicate database connections, or use **alter connection** to set **dsi_compile_retry_threshold** for a connection to a database and data server that you specify.

Note: You must enable RTL or HVAR with **dsi_compile_enable** to use **dsi_compile_retry_threshold**. You must execute a separate **configure replication server** or **alter connection** command for each parameter you want to change. Do not enter more than one parameter after entering **configure replication server** or **alter connection**.

- Server level:

```
configure replication server
set dsi_compile_enable to 'on'
go
...
configure replication server
set dsi_compile_retry_threshold to 'value'
go
```

- Database connection level:

```
alter connection to data server.database
set dsi_compile_enable to 'on'
go
...
alter connection to data server.database
set dsi_compile_retry_threshold to 'value'
go
```

Valid values for **dsi_compile_retry_threshold** are integers between 0 – 2,147,483,647. The default value is 100.

You need not suspend and resume database connections when you set **dsi_compile_retry_threshold**. The parameter takes effect immediately after you execute the command.

See also

- *RTL Configuration* on page 60

Increased Queue Block Size Enhancement

You can change the queue block size without restarting Replication Server .

The queue block size is the number of bytes in a contiguous block of memory used by stable queue structures. In Replication Server 15.5, you can increase the queue block size from the default of 16KB to 32KB, 64KB, 128KB, or 256KB to enhance replication performance. Performance improvement is also dependent on the transaction profile and the environment. However, in version 15.5 you must restart Replication Server for the change in queue block size to take effect. In version 15.6, you need not restart Replication Server for the change in in queue block size to take effect.

Note: You must have the Advanced Services Options license, named REP_HVAR_ASE, to use the increased queue block size feature.

Recommendations

Sybase strongly recommends that you:

- Verify you have sufficient memory before you increase the queue block size.
- Experiment with different queue block sizes to determine the optimum value for your replication system.

Restrictions

- Make sure that there is no data flowing into Replication Server while the queue block size change is in progress.
- You cannot change the queue block size while a subscription is being materialized, if dematerialization is in progress, or if routes are being created or destroyed. The queue block size change terminates with an error message while Replication Server continues operating.
- Once you start the procedure to change the queue block size, Replication Server does not accept another command to change the queue block size until the first change is completed.
- Do not use any other procedures to change the queue block size in the RSSD directly, as these procedures may result in inconsistencies in the queue block size configuration and cause Replication Server to shut down.

Note: All queues are drained after the block size changes.

See also

- *Replication Server 15.6 Product Editions and Licenses* on page 49

Changing the Queue Block Size

Modifying the queue block size is a major change to the Replication Server configuration and affects all connections to the Replication Server. You must suspend log transfer and quiesce Replication Server.

In the queue block size change procedure, "upstream" refers to all replication system components that feed data to the Replication Server where you want to change the queue block size and "downstream" refers to the components that receive data from the affected Replication Server.

1. To maintain data integrity, you must stop data flowing into the Replication Server you want to configure before you change the queue block size.:
   a) Suspend log transfer from all Replication Agents to the Replication Server you want to configure.

b) Suspend all upstream log transfer from Replication Agents.
c) Quiesce all upstream Replication Servers.
d) Suspend all incoming routes to the Replication Server you want to configure.
e) Quiesce the Replication Server you want to configure.

2. Use **configure replication server** with the **set block_size to** '*value*' clause to set the queue block size on the Replication Server you want to configure. This command:
   1. Verifies that there is no subscription materialization in progress.
   2. Verifies that all log transfer is suspended.
   3. Verifies that all incoming routes are suspended.
   4. Verifies that the Replication Server is quiesced.
   5. Purges queues.
   6. Zeros the values in the `rs_locater` RSSD system table to allow Replication Agents to resend transactions that may have not been applied to the replicate database when you started the queue block size change procedure.
   7. Sets the queue block size to the value you entered.
   8. (Optional) If you include the **with shutdown** option, Replication Server shuts down. The queue block size change takes effect when you restart the Replication Server. Shutting down ensures that Replication Server clears all memory.
3. After changing the queue block size, resume data flow:
   a) If you used the **with shutdown** option, restart the Replication Server.
   b) Resume log transfer from Replication Agents.
   c) Resume all incoming routes.
4. Check for data loss at all downstream Replication Server RSSDs and data servers. Usually, there is data loss from the RSSD of the Replication Server you configured. Ignore the data loss from a replicate RSSD that receives data from the RSSD of the configured Replication Server.

   Follow the procedures to fix data loss at the data servers. If there is data loss at a RSSD, you see a message similar to this in the log of the affected Replication Server:

   ```
   E. 2010/02/12 14:12:58. ERROR #6067 SQM(102:0 primaryDS.rssd) - /
   sqmoqid.c(1071)
   Loss detected for replicateDS.rssd from primaryDS.RSSD
   ```

   replicateDS is the replicate data server name and *primaryDS* is the primary data server name.

Increasing Queue Block Size in a Simple Replication System

Learn to set the queue block size of the primary and replicate Replication Servers in this example of a simple replication system.

The replication system consists of:

- primary database – `pdb`

- replicate database – `rdb`
- primary Replication Server – PRS
- RSSD of primary Replication Server – `pRSSD`
- replicate Replication Server – RRS
- RSSD of replicate Replication Server – `rRSSD`

```
          pRSSD       rRSSD

            |           |

pdb -----> PRS -----> RRS -----> rdb
```

In this example, RSSD refers to both Adaptive Server as the Replication Server System Database (RSSD) and SQL Anywhere® as the Embedded Replication Server System Database (ERSSD). See the *Replication Server Reference Manual* for the full syntax, examples, and usage information for all commands.

1. Configure the primary Replication Server:

a) Suspend log transfer from all Replication Agents. At the primary Replication Server, execute:

```
suspend log transfer from all
```

b) Quiesce the primary Replication Server:

```
admin quiesce_force_rsi
```

c) Set the queue block size at the primary Replication Server to 64KB:

```
configure replication server
set block_size to ‘64’
```

(Optional) Use the **with shutdown** option to set the block size and shut down the primary Replication Server. For example:

```
configure replication server
set block_size to ‘64’ with shutdown
```

d) Look at the transaction log to verify that the primary Replication Server is not materializing, that log transfer and routes are suspended, and that the primary Replication Server is quiesced.

e) Restart the primary Replication Server if you have shut it down. See “Starting Replication Server,” in Chapter 4, “Managing a Replication System” in the *Replication Server Administration Guide Volume 1*.

f) Look at the primary Replication Server transaction log to verify that the block size is changed.

g) Resume log transfer to allow Replication Agents to connect to the primary Replication Server. At the primary Replication Server execute:

```
resume log transfer from all
```

h) Check the replicate Replication Server log file for information about data losses. Ignore data loss occurring from the primary Replication Server RSSD to the replicate Replication Server RSSD by executing the **ignore loss** command on the replicate Replication Server:

```
ignore loss from PRS.pRSSD to RRS.rRSSD
```

See "Ignoring a loss," in Chapter 7, "Replication System Recovery" in the *Replication Server Administration Guide Volume 2*.

2. Configure the replicate Replication Server:
   a) Suspend log transfer from all Replication Agents. At the primary Replication Server and at the replicate Replication Server, execute:

      ```
      suspend log transfer from all
      ```

   b) Quiesce the primary Replication Server:

      ```
      admin quiesce_force_rsi
      ```

   c) At all Replication Servers that originate routes to the replicate Replication Server, suspend the routes:

      ```
      suspend route to RRS
      ```

   d) Quiesce the replicate Replication Server:

      ```
      admin quiesce_force_rsi
      ```

   e) Set the block size at the replicate Replication Server to 64KB:

      ```
      configure replication server
      set block_size to ‘64’
      ```

      (Optional) Use the **with shutdown** option to shut down the replicate Replication Server. For example:

      ```
      configure replication server
      set block_size to ‘64’ with shutdown
      ```

   f) Look at the transaction log to verify that the replicate Replication Server is not materializing, that log transfer and routes are suspended, and that the replicate Replication Server is quiesced.
   g) Restart the replicate Replication Server if you have shut it down.
   h) Look at the the replicate Replication Server transaction log to verify that the block size is changed.
   i) Resume log transfer to allow Replication Agents to connect to the replicate Replication Server. At the replicate Replication Server, execute:

      ```
      resume log transfer from all
      ```

   j) Resume log transfer to allow Replication Agents to connect to the primary Replication Server. At the primary Replication Server execute:

      ```
      resume log transfer from all
      ```

   k) Resume the routes you suspended:

      ```
      resume route to RRS
      ```

   l) Check the the primary and replicate Replication Server log files for information about data losses. Ignore data loss occurring between the primary RSSD and the replicate RSSD if the replicate RSSD is replicated to the primary RSSD by executing the **ignore loss** command on the primary Replication Server.

      ```
      ignore loss from RRS.rRSSD to PRS.pRSSD
      ```

Increasing Queue Block Size in a Replication System with an Intermediate Route

Learn to set the queue block size of the primary Replication Server in this example of a replication system with an intermediate route.

The replication system consists of:

- primary database – `pdb`
- replicate database – `rdb`
- primary Replication Server – PRS
- RSSD of primary Replication Server – `pRSSD`
- replicate Replication Server – RRS
- RSSD of replicate Replication Server – `rRSSD`
- intermediate Replication Server – IRS
- RSSD of intermediate Replication Server – `iRSSD`

```
          pRSSD       iRSSD       rRSSD
            |           |           |
pdb -----> PRS -----> IRS -----> RRS -----> rdb
```

In this example, RSSD refers to both Adaptive Server as the Replication Server System Database (RSSD) and SQL Anywhere as the Embedded Replication Server System Database (ERSSD). See the *Replication Server Reference Manual* for the full syntax, examples, and usage information for all commands.

1. Suspend log transfer from all Replication Agents. At the primary Replication Server, execute:
   ```
   suspend log transfer from all
   ```
2. Quiesce PRS:
   ```
   admin quiesce_force_rsi
   ```
3. Set the block size at the primary Replication Server to 64KB:
   ```
   configure replication server
   set block_size to ‘64’
   ```
   (Optional) Use the **with shutdown** option to set the block size and shut down the primary Replication Server. For example:
   ```
   configure replication server
   set block_size to ‘64’ with shutdown
   ```
4. Look at the transaction log to verify that the primary Replication Server is not materializing, that log transfer and routes are suspended, and that the primary Replication Server is quiesced.

5. Restart the primary Replication Server if you have shut it down. See “Starting Replication Server,” in Chapter 4, “Managing a Replication System” in the *Replication Server Administration Guide Volume 1*.
6. Look at the primary Replication Server transaction log to verify that the block size is changed.
7. Resume log transfer to allow Replication Agents to connect to the primary Replication Server. At the primary Replication Server execute:

   ```
   resume log transfer from all
   ```

8. Check the intermediate and replicate Replication Server log files for information about data losses. Ignore data loss occurring from the primary Replication Server RSSD to the replicate Replication Server and from the primary RSSD to the intermediate RSSD by executing the **ignore loss** command twice on the intermediate Replication Serve:

   ```
   ignore loss from PRS.pRSSD to RRS
   go
   ignore loss from PRS.pRSSD to IRS.iRSSD
   ```

   See "Ignoring a loss," in Chapter 7, "Replication System Recovery" in the *Replication Server Administration Guide Volume 2*.

Usability and Process Improvements

Replication Server 15.6 introduces several usability and process enhancements.

Replicate Database Resynchronization for Adaptive Server

Replication Server allows you to resynchronize and materialize the replicate database, and resume further replication without losing or risking inconsistency of data, and without forcing a quiesce of your primary database.

Database resynchronization is based on obtaining a dump of data from a trusted source and applying the dump to the target database you want to resynchronize.

To resynchronize Oracle databases, see *Replication Server Heterogeneous Replication Guide > Oracle Replicate Databases Resynchronization*.

Configuring Database Resynchronization

Use commands and parameters from both Replication Server and RepAgent to configure database resynchronization.

1. Stop replication processing by suspending RepAgent.
2. Place Replication Server in resync mode.
   In resync mode, Replication Server skips transactions and purges replication data from replication queues in anticipation of the replicate database being repopulated from a dump taken from the primary database or trusted source.

3. Restart RepAgent and send a resync database marker to Replication Server to indicate that a resynchronization effort is in progress.
4. Verify that DSI receives the resync database marker.
5. Obtain a dump from the primary database.

   When Replication Server detects a dump marker that indicates the completion of the primary database dump, Replication Server stops skipping transactions and can determine which transactions to apply to the replicate database.
6. Verify that DSI receives the dump database marker.

   Note: Sending a dump database marker does not apply in cases where you send the resync marker with the **init** instruction.
7. Apply the dump to the replicate database.
8. Resume replication.

Instructing Replication Server to Skip Transactions

Use the **skip to resync** parameter with the **resume connection** command to instruct Replication Server to skip transactions in the DSI outbound queue for the specified replicate database until Replication Server receives and acknowledges a dump database marker sent by RepAgent.

Replication Server does not process records in the outbound queue, since the data in the replicate database is expected to be replaced with the dump contents.

See *Replication Server Reference Manual > Replication Server Commands >* ***resume connection***.

Run:

```
resume connection to data_server.database skip to resync marker
```

Warning! If you execute **resume connection** with the **skip to resync marker** option on the wrong connection, data on the replicate database becomes unsynchronized.

When you set **skip to resync marker**, Replication Server does not log the transactions that are skipped in the Replication Server log or in the database exceptions log. Replication Server logs transactions that are skipped when you set **skip [n] transaction**.

Send the Resync Database Marker to Replication Server

Instruct RepAgent to send a resync database marker to Replication Server to indicate that a resynchronization effort is in progress.

When you restart RepAgent in resync mode, RepAgent sends the resync database marker to Replication Server as the first message before it sends any SQL data definition language (DDL) or data manipulation language (DML) transactions. Multiple replicate databases for the same primary database all receive the same resync marker since they each have a DSI outbound queue.

For each DSI that resumes with the **skip to resync marker** parameter, the DSI outbound queue records in the Replication Server system log that DSI has received the resync marker and also records that from that point forward, DSI rejects committed transactions until it receives the dump database marker.

In Adaptive Server, use **sp_start_rep_agent** with the **resync**, **resync purge,** or **resync init** parameters to support the corresponding options for sending the resync database marker.

Sending a Resync Marker Without Any Option

Send a resync marker using **sp_start_rep_agent** without any option when there is no change to the truncation point and the expectation is that the RepAgent should continue processing the transaction log from the last point that it processed.

Syntax: **sp_start_rep_agent** *database_name*, **'resync'**

Each outbound DSI thread and queue receives and processes the resync database marker. DSI reports to the Replication Server system log when a resync marker has been received, satisfying the skip to resync marker request of DSI. Subsequently, DSI rejects committed transactions while it waits for a dump database marker. With this message and the change of behavior to one of waiting for the dump database marker, you can apply any dump to the replicate database.

Sending a Resync Marker with a purge Instruction

Send a resync marker using **sp_start_rep_agent** with the **purge** option to instruct Replication Server to purge all open transactions from the inbound queue, and reset duplicates detection, before receiving any new inbound transactions.

Syntax: **sp_start_rep_agent** *database_name*, **'resync purge'**

Use the **purge** option when the truncation point of the primary database has been moved, which occurs when you:

- Manually change the truncation point.
- Disable RepAgent.
- Execute Adaptive Server commands such as, **dbcc dbrepair**.

Since the truncation point has changed, open transactions in the Replication Server inbound queue must be purged because these transactions do not match new activity sent from the new secondary truncation point. Replication Server resets checking for duplicates since the changed truncation point could send a record with a previous origin queue ID (OQID). Since the prior data is purged from the queues, Replication Server does not treat any new activity from the RepAgent as duplicate activity, and consequently does not reject the new activity. The purge option does not change DSI processing because Replication Server continues to reject outbound queue commands while waiting for the dump database marker.

Sending a Resync Marker with the init Command

Send a resync marker with an **init** command using **sp_start_rep_agent** with the init option to instruct Replication Server to purge all open transactions from the inbound queue, reset duplicate detection, and suspend the outbound DSI.

Syntax: **sp_start_rep_agent** *database_name*, **'resync init'**

Use this option to reload the primary database from the same dump as the replicate database. Since there is no dump taken from the primary database, RepAgent does not send a dump database marker. Instead of waiting for a dump database marker after the resync marker, the **init** option suspends the DSI connection immediately after Replication Server processes the resync marker.

After DSI is suspended, all subsequent activity through DSI consists of new transactions. You can resume DSI once you reload the replicate database from the same dump you used for the primary.

See also

- *Send the Dump Database Marker to Replication Server* on page 80
- *Resynchronizing Both the Primary and Replicate Databases from the Same Dump* on page 85

Obtain a Dump of the Database

Use the **dump database** Adaptive Server command.

See *Adaptive Server Enterprise > System Administration Guide: Volume 2 > Developing a Backup and Recovery Plan > Using the* **dump** *and* **load** *Commands*.

Send the Dump Database Marker to Replication Server

RepAgent automatically generates and sends a dump database marker to Replication Server when you obtain a dump of the primary database.

Note: Sending a dump database marker does not apply when you send the resync marker with the **init** instruction.

You can manually resume DSI after you apply the dump to the replicate database. shuyingy, 29Sep2011: OK Transactions that commit after the dump point, which is indicated by the dump database marker, are replicated.

Monitor DSI Thread Information

Use the **admin who** command to provide information on DSI during database resynchronization.

State	Description
SkipUntil Resync	DSI resumes after you execute **skip to resync**. This state remains until DSI receives a resync database marker.
SkipUntil Dump	DSI has received a resync database marker. This state remains until DSI has processed a subsequent dump database marker.

Apply the Dump to a Database to be Resynchronized

You can load the primary database dump to the replicate database only after you see the relevant messages in the system log.

- When Replication Server receives the resync database marker with or without the **purge** option, and the dump database marker:

  ```
  DSI for data_server.database received and processed
  Resync Database Marker. Waiting for Dump Marker.
  ```

  ```
  DSI for data_server.database received and processed
  Dump Marker. DSI is now suspended. Resume after database has been
  reloaded.
  ```

- When Replication Server receives the resync database with **init** marker:

  ```
  DSI for data_server.database received and processed
  Resync Database Marker. DSI is now suspended. Resume after
  database has been reloaded.
  ```

See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* ***load database*** for instructions about loading the dump to the database you want to resynchronize.

Database Resynchronization Scenarios

Follow the procedure to resynchronize databases in different scenarios. After completing a procedure, the primary and replicate databases are transactionally consistent.

To execute a procedure, you must:

- Be a replication system administrator
- Have an existing replication environment that is running successfully
- Have methods and processes available to copy data from the primary database to the replicate database

For commands and syntax for RepAgent for Adaptive Server and Replication Server, see the *Replication Server Reference Manual* and *Replication Server Administration Guide Volume 1 > Manage RepAgent and Support Adaptive Server*.

Resynchronize One or More Replicate Databases Directly from a Primary Database

Resynchronize one or multiple replicate databases from a single primary database.

This procedure with minor variations, allows you to:

- Repopulate the replicate database when the replication latency between primary and replicate databases is such that to recover a database using replication is not feasible, and reporting based on the replicate data may not be practical because of the latency.
- Repopulate the replicate database with trusted data from the primary database.
- Coordinate resynchronization when the primary database is the source for multiple replicate databases.
- Coordinate resynchronization if the primary site is a logical database that consists of a warm standby pair of databases that you want to resynchronize with one or more replicate databases. In a warm standby pair, the active database acts as the primary database, and the standby acts as the replicate database. Therefore, the active database of a warm standby pair at a primary site also appears as a primary database to one or multiple replicate databases.

See also

- *Resynchronizing the Active and Standby Databases in a Warm Standby Application* on page 87

Resynchronizing Directly from a Primary Database

Resynchronize a replicate database from a primary database.

1. Stop replication processing by RepAgent. In Adaptive Server, execute:
   ```
   sp_stop_rep_agent database
   ```
2. Suspend the Replication Server DSI connection to the replicate database:
   ```
   suspend connection to dataserver.database
   ```
3. Instruct Replication Server to remove data from the replicate database outbound queue and wait for a resync marker from the primary database RepAgent:
   ```
   resume connection to data_server.database skip to
   resync marker
   ```
4. Instruct RepAgent to start in resync mode and send a resync marker to Replication Server:
   - If the truncation point has not been moved from its original position, in Adaptive Server execute:
     ```
     sp_start_rep_agent database, 'resync'
     ```
   - If the truncation point has been moved from its original position, in Adaptive Server execute:
     ```
     sp_start_rep_agent database, 'resync purge'
     ```
5. In the Replication Server system log, verify that DSI has received and accepted the resync marker from RepAgent by looking for this message:

```
DSI for data_server.database received and processed
Resync Database Marker. Waiting for Dump Marker.
```

Note: If you are resynchronizing multiple databases, verify that the DSI connection for each of the databases you want to resynchronize has accepted the resync marker.

6. Obtain a dump of the primary database contents. See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* **dump database**. Adaptive Server automatically generates a dump database marker.
7. Verify that Replication Server has processed the dump database marker by looking for this message in the Replication Server system log:

   ```
   DSI for data_server.database received and processed
   Dump Marker. DSI is now suspended. Resume after database has been
   reloaded.
   ```

   When Replication Server receives the dump marker, the DSI connection automatically suspends.
8. Apply the dump of the primary database to the replicate database. See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* **load database**.
9. After you apply the dump to the replicate database, resume DSI:

   ```
   resume connection to data_server.database
   ```

Resynchronizing Using a Third-Party Dump Utility

Coordinate resynchronization after you dump the primary database using a third-party dump utility, such as a disk snapshot.

Third-party tools do not interact as closely with the primary database as native database dump utilities. If your third-party tool does not record anything in the primary database transaction log that RepAgent can use to generate a dump database marker, generate your own dump database markers to complete the resynchronization process. See your third-party tool documentation.

1. Stop replication processing by RepAgent. In Adaptive Server, execute:

   ```
   sp_stop_rep_agent database
   ```
2. Suspend the Replication Server DSI connection to the replicate database:

   ```
   suspend connection to dataserver.database
   ```
3. Instruct Replication Server to remove data from the replicate database outbound queue and wait for a resync marker from the primary database RepAgent:

   ```
   resume connection to data_server.database skip to
   resync marker
   ```
4. Obtain a dump of the primary database contents using the third-party dump utility.
5. Determine the dump point based on information from the primary database when you took the dump, or information from the third-party tool. With a third-party tool, you are responsible for determining the dump point. For example, if you are using a disk replication tool, you can temporarily halt activity at the primary database to eliminate

transactions in progress from the disk snapshot, and then use the “end of transaction log” point as the dump database marker.

6. Execute the **rs_marker** stored procedure on the primary database for RepAgent to mark the end of the dump position that you obtained in step 5:

```
rs_marker “dump database database_name 'current date' oqid"
```

where *current date* is any value in `datetime` format and *oqid* is any valid hexadecimal value. See *Replication Server Reference Manual > Topics > Datatypes > Date/time, and Date and Time Datatypes > Entry Format for Date/Time Values*.

For example, you can mark the end of the dump position on the `rdb1` database with a date and time value of "20110915 14:10:10" and a value of 0x0003 for *oqid*:

```
rs_marker “dump database rdb1 '20110915 14:10:10' 0x0003"
```

RepAgent automatically generates a dump database marker for the point you marked in step 6, and sends the dump database marker to Replication Server.

7. Instruct RepAgent to start in resync mode and send a resync marker to Replication Server:
   - If the truncation point has not been moved from its original position, execute this command in Adaptive Server:

     ```
     sp_start_rep_agent database, 'resync'
     ```
   - If the truncation point has been moved from its original position, execute this command in Adaptive Server:

     ```
     sp_start_rep_agent database, 'resync purge'
     ```
8. Verify that DSI has received and accepted the resync marker from Replication Agent by looking for this message in the Replication Server system log:

```
DSI for data_server.database received and processed
Resync Database Marker. Waiting for Dump Marker.
```

9. Verify that Replication Server has processed the dump database marker by looking for this message in the Replication Server system log:

```
DSI for data_server.database received and processed
Dump Marker. DSI is now suspended. Resume after
database has been reloaded.
```

When Replication Server receives the dump marker, the DSI connection automatically suspends.

10. Apply the dump of the primary database from the third-party tool to the replicate database. See your Adaptive Server and third-party tool documentation.
11. After you apply the dump to the replicate database, resume DSI:

```
resume connection to data_server.database
```

Resynchronizing if There is No Support for the Resync Database Marker

Coordinate resynchronization if the RepAgent or the primary database have not been updated to support automatic generation of a resync marker.

Note: You can use this procedure for Adaptive Server only.

1. Suspend the Replication Server DSI connection to the replicate database:

   ```
   suspend connection to dataserver.database
   ```

2. Instruct Replication Server to remove data from the replicate database outbound queue and wait for a resync marker from the primary database RepAgent:

   ```
   resume connection to data_server.database skip to
   resync marker
   ```

3. Ensure that there are no open transactions in system log, and then in the primary database, manually generate the **resync marker**:

   ```
   execute rs_marker 'resync database'
   ```

4. In the Replication Server system log, verify that DSI has received and accepted the resync marker from RepAgent by looking for this message:

   ```
   DSI for data_server.database received and processed
   Resync Database Marker. Waiting for Dump Marker.
   ```

5. Obtain a dump of the primary database contents.

   Adaptive Server automatically generates a dump database marker.See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* ***dump database***.

6. Verify that Replication Server has processed the dump database marker by looking for this message in the Replication Server system log:

   ```
   DSI for data_server.database received and processed
   Dump Marker. DSI is now suspended. Resume after database has been
   reloaded.
   ```

   When Replication Server receives the dump marker, the DSI connection automatically suspends.

7. Apply the dump of the primary database to the replicate database. See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* ***load database***.

8. After you apply the dump to the replicate database, resume DSI:

   ```
   resume connection to data_server.database
   ```

Resynchronizing Both the Primary and Replicate Databases from the Same Dump

Coordinate resynchronization to reload both the primary database and replicate database from the same dump or copy of data. No dump database marker is needed, since you are not obtaining a dump from the primary database.

1. Stop replication processing by RepAgent. Do not alter the truncation point.

   In Adaptive Server, execute:

```
sp_stop_rep_agent database
```

2. Suspend the Replication Server DSI connection to the replicate database:

```
suspend connection to data_server.database
```

3. Instruct Replication Server to remove data from the replicate database outbound queue and wait for a resync marker from the primary database RepAgent:

```
resume connection to data_server.database skip to
resync marker
```

4. Obtain the RepAgent settings before you apply the dump.

Note: Adaptive Server stores, within the database, the connectivity settings and other configurations that RepAgent uses. If you load the primary database from a dump that you took from a different database, RepAgent loses its configuration settings, or the settings change to match the settings of the database from which you took the dump.

5. Apply the dump of the data from the external source to the primary database.

   After you apply the dump, reset the RepAgent configurations to the settings that existed before you applied the dump.

6. Make sure that the last primary database transaction log page does not contain any operation that can affect replicate database tables by executing at the primary Adaptive Server database:

```
rs_update_lastcommit 0, 0, 0, ""
go 100
```

7. Move the truncation point to the end of the transaction log for the primary database. In Adaptive Server, execute:

```
dbcc settrunc(‘ltm’, ‘end’)
go
```

8. Instruct RepAgent to start in resync mode with the **init** instruction. In Adaptive Server, execute:

```
sp_start_rep_agent database, 'resync init'
```

9. Verify that DSI has received and accepted the resync marker from the RepAgent by looking for this message in the Replication Server system log:

```
DSI for data_server.database received and processed
Resync Database Marker. DSI is now suspended. Resume
after database has been reloaded.
```

   When Replication Server receives and processes the resync database with **init** marker, the DSI connection suspends.

10. Apply the dump of the data from the external source to the replicate database.
11. After you apply the dump to the replicate database, resume DSI to the replicate database to allow Replication Server to apply transactions from the primary database:

```
resume connection to data_server.database
```

Resynchronizing the Active and Standby Databases in a Warm Standby Application

Resynchronize the active and standby databases in a warm standby environment, when the warm standby pair is the replicate site for a single primary database.

In this scenario, the replicate site is a warm standby pair that consists of the active and standby databases that act as a single logical database.

```
Primary ---> Replication ---> Replicate logical database
database     Server           [Active+Standby warm standby
                               pair]
```

The resynchronization scenario procedure is a two-step process—resynchronize the replicate active database of the warm standby pair with a dump from the primary database, and then resynchronize the replicate standby database of the warm standby pair with a dump from the active database or the existing dump from the primary database.

1. Stop replication processing by both the primary database RepAgent and the warm standby active database RepAgent.

   In Adaptive Server, execute:

   ```
   sp_stop_rep_agent database
   ```

2. Suspend the Replication Server DSI connection to the active and standby databases:

   ```
   suspend connection to dataserver.database
   ```

3. Instruct Replication Server to remove data from the outbound queue of the active and standby databases, and wait for a resync marker from the primary database RepAgent:

   ```
   resume connection to data_server.database skip to
   resync marker
   ```

4. Instruct the primary database RepAgent to start in resync mode and send a resync marker to Replication Server.

   - If the truncation point has not been moved from its original position, execute this command in Adaptive Server:

     ```
     sp_start_rep_agent database, 'resync'
     ```

   - If the truncation point has been moved from its original position, execute this command in Adaptive Server:

     ```
     sp_start_rep_agent database, 'resync purge'
     ```

5. Verify that DSI for the active database has received and accepted the resync marker from the primary database RepAgent by looking for this message in the Replication Server system log:

   ```
   DSI for data_server.database received and processed
   Resync Database Marker. Waiting for Dump Marker.
   ```

6. Obtain a dump of the primary database contents. See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* **dump database**. Adaptive Server automatically generates a dump database marker.
7. Obtain the RepAgent settings before you apply the dump.

   Note: Adaptive Server stores, within the database, the connectivity settings and other configurations that RepAgent uses. If you load the primary database from a dump that you took from a different database, RepAgent loses its configuration settings, or the settings change to match the settings of the database that you took the dump from.

8. Verify that the Replication Server DSI for the active database has processed the dump database marker by looking for this message from the active database in the Replication Server system log:

   ```
   DSI for data_server.database received and processed
   Dump Marker. DSI is now suspended. Resume after database has been
   reloaded.
   ```

9. Apply the dump of the primary database to the active database. See *Adaptive Server Enterprise Reference Manual: Commands > Commands >* **load database**.

   After you apply the dump, reset the RepAgent configurations to the settings that existed before you applied the dump.

10. Make sure that the last primary database transaction log page does not contain any operation that can affect replicate database tables by executing at the primary Adaptive Server database:

    ```
    rs_update_lastcommit 0, 0, 0, ""
    go 100
    ```

11. Move the truncation point to the end of the transaction log for the active database. In Adaptive Server, execute:

    ```
    dbcc settrunc(‘ltm’, ‘end’)
    go
    ```

12. Instruct RepAgent to start in resync mode with the **init** instruction. In Adaptive Server, execute:

    ```
    sp_start_rep_agent database, 'resync init'
    ```

13. Verify that DSI for the standby database has received and accepted the resync marker from the active database RepAgent by looking for this message in the Replication Server system log:

    ```
    DSI for data_server.database received and processed
    Resync Database Marker. DSI is now suspended. Resume
    after database has been reloaded.
    ```

    When Replication Server receives and processes the resync database with **init** marker, the DSI connection suspends.

14. Obtain a dump of the active database contents and apply the dump to the standby database. You can also apply the dump of the primary database from step 6 if the dump does not include database configuration information.

15. Resume DSI to the active and standby databases:

```
resume connection to data_server.database
```

See also

- *Resynchronize One or More Replicate Databases Directly from a Primary Database* on page 82

Delete Exceptions by Range

Use stored procedures to delete transactions in the RSSD exceptions log by range instead of individually.

With the **rs_delexception** stored procedure, you can specify only one transaction to delete from the exceptions log. With Replication Server 15.6, you can use **rs_delexception_id**, **rs_delexception_date**, and **rs_delexception_range** to specify a range of transactions to delete.

When you specify a range of transactions, Replication Server conserves resources and improves performance by creating only one temporary table for the range of transactions instead of a table for each transaction.

rs_delexception_id

Deletes a range of transactions identified by transaction ID in the exceptions log in the `rs_exceptscmd`, `rs_exceptshdr`, and `rs_systext` system tables.

Syntax

```
rs_delexception_id transaction_id_start [,transaction_id_end]
```

Parameters

- **transaction_id_start –** ID number of the first transaction in the range that you want to delete.
- **transaction_id_end –** ID number of the last transaction in the range that you want to delete. Specifying the last transaction in a range is optional.

Examples

- **Example 1 –** deletes from the exceptions log the transaction with ID number 1234. You can also use **rs_delexception_id** to delete a single transaction.

  ```
  rs_delexception_id 1234
  ```

- **Example 2 –** deletes from the exceptions log all transactions with ID numbers between 1234 and 9800, inclusive.

  ```
  rs_delexception_id 1234, 9800
  ```

Usage

- **rs_delexception_id** deletes the range of transactions between *transaction_id_start* and *transaction_id_end*, inclusive of *transaction_id_start* and *transaction_id_end* from the exception tables.
- If you do not specify any parameter, **rs_delexception_id** displays an error message. Use **rs_helpexception** or **rs_delexception** with no parameters to obtain a current list of valid transactions in the exceptions log.
- If you specify a single valid value for a transaction ID in *transaction_id_start*, and do not specify a second transaction ID number in *transaction_id_end*, **rs_delexception_id** deletes only the transaction you specify in *transaction_id_start*.
- If you enter 0 (zero) as a transaction ID number and do not enter a second transaction ID number, **rs_delexception_id** deletes all transactions in the exceptions log.
- If you enter a floating point number, such as 123.456, and you are using:
  - **ERSSD – rs_delexception_id** only processes the integer—123, and ignores the numerals after the decimal point
  - **RSSD – rs_delexception_id** returns with an error message and you can reenter the command
- **rs_delexception_id** displays an error message if the command you enter does not result in any transactions being deleted.

rs_delexception_date

Deletes a range of transactions identified by transaction date in the exceptions log in the `rs_exceptscmd`, `rs_exceptshdr`, and `rs_systext` system tables.

Syntax

```
rs_delexception_date transaction_date_start [,transaction_date_end]
```

Parameters

- **transaction_date_start –** the originating date of the earliest transactions in the range that you want to delete. Enclose the date in double quotation marks.
- **transaction_date_end –** the originating date of the latest transactions in the range that you want to delete. Specifying the latest transaction originating date in a range of dates is optional. Enclose the date in double quotation marks.

Examples

- **Example 1 –** deletes from the exceptions log the transactions with an originating date of 1st October 2010.

  ```
  rs_delexception_date "10/01/2010"
  ```
- **Example 2 –** deletes from the exceptions log all transactions that have originating dates between 1st October 2010 and 31st October 2010, inclusive.

```
rs_delexception_date "10/01/2010", "10/31/2010"
```

Usage

- You can enter the dates for *transaction_date_end* and *transaction_date_end* in the different formats supported by the Adaptive Server hosting the RSSD or the SQL Anywhere database that is the ERSSD. For information about acceptable date and time formats, see:
  - *Adaptive Server Enterprise Reference Manual: Building Blocks > System and User-Defined Datatypes > Date and time datatypes > Entering date and time data*
  - *SQL Anywhere Server - SQL Reference > SQL Data Types > Date and Time Data Types > Sending Dates and Times to the Database*.
- **rs_delexception_date** deletes the range of transactions between *transaction_date_start* and *transaction_date_end*, inclusive of *transaction_date_start* and *transaction_date_end* from the exception tables.
- If you do not specify any parameter, **rs_delexception_date** displays an error message. See the `"org date"`column when you execute **rs_helpexception** or **rs_delexception** with no parameters to obtain a current of valid transactions and originating dates in the exceptions log.
- If you specify a valid date only for *transaction_date_start*, and do not specify a second valid date in *transaction_date_end*, **rs_delexception_date** deletes only the transactions you specify in *transaction_date_start*.
- **rs_delexception_date** displays an error message if the command you enter does not result in any transactions being deleted.

rs_delexception_range

Deletes a range of transactions identified by originating site or user , or destination site in the exceptions log in the `rs_exceptscmd`, `rs_exceptshdr`, and `rs_systext` system tables.

Syntax

```
rs_delexception_range
{{"origin"|"org"}, "origin_data_server.origin_database" |
, {"destination"|"dest"},
"destination_data_server.destination_database" |
, "user", "origin_user"}
```

Parameters

- **"origin"/"org", "*origin_data_server.origin_database*" –** enter **"origin"** or the short form—**"org"** and specify the data server and database that originated the transactions you want to delete from the exceptions log. Enclose these parameters in double quotation marks, and use commas to separate the parameters from each other.
- **"destination"/"dest", "*destination_data_server.destination_database*" –** enter **destination** or the short form—**"dest"** and specify the data server and database that received the transactions you want to delete from the exceptions log. Enclose these

parameters in double quotation marks, and use commas to separate the parameters from each other.

- **"user", "*origin_user*" –** enter **"user"** and specify the user that originated the transactions you want to delete from the exceptions log. Enclose these parameters in double quotation marks, and use commas to separate the parameters from each other.

Examples

- **Example 1 –** deletes from the exceptions log the transactions that originated from the `south_db` database of the SYDNEY_DS data server.

```
rs_delexception_range "org", "SYDNEY_DS.south_db"
```

- **Example 2 –** deletes from the exceptions log the transactions that were received by the `east_db` database of the TOKYO_DS data server.

```
rs_delexception_range "destination", "TOKYO_DS.east_db"
```

- **Example 3 –** deletes from the exceptions log the transactions that originated from the rsuser1 user.

```
rs_delexception_range "user", "rsuser1"
```

Usage

- You can enter only one parameter and the corresponding value at a time. For example, you cannot enter "**org**", "*origin_dataserver.origin_database*" followed by "**user**", "*origin_user*".
- You must enter a parameter and specify a value. If you do not specify any parameter, **rs_delexception_range** displays an error message. See the `Origin Site`, `Dest. Site`, and `Dest. User` columns when you execute **rs_helpexception** or **rs_delexception** with no parameters, to obtain a current list of values for the respective columns for valid transaction in the exceptions log.
- If you enter only **"origin"**, **"destination"**, or **"user"** with **rs_delexception_range**, and do not specify the corresponding values, **rs_delexception_range** displays an error message.
- **rs_delexception_range** displays an error message if the command you enter does not result in any transactions being deleted.

Controlling Row Count Validation

Use **dsi_row_count_validation** to disable row count validation.

In version 15.2 and later, Replication Server enables row count validation by default and automatically displays error messages and performs default error actions in reaction to different row count validation errors such as row count mismatch. You can configure the Replication Server error class to enable different error actions.

With Replication Server 15.6, if you have table rows that are not synchronized, and you want to bypass the default error actions and messages, you can set **dsi_row_count_validation** to **off** to disable row count validation.

dsi_row_count_validation is set to **on**, by default, to enable row count validation.

Use **configure replication server** to set **dsi_row_count_validation** at the server level to affect all replicate database connections, or use **alter connection** to set the parameter for a connection to a database and data server that you specify. For example, to:

- Disable row count validation for all database connections:

  ```
  configure replication server
  set dsi_row_count_validation to 'off'
  ```

  You must suspend and resume all database connections to Replication Server after you execute **configure replication server** with **dsi_row_count_validation**. The change in setting takes effect after you resume database connections.
- Enable row count validation for a specific connection — `pubs2` database in SYDNEY_DS data server:

  ```
  alter connection to SYDNEY_DS.pubs2
  set dsi_row_count_validation to 'on'
  ```

  You need not suspend and resume a database connection when you set **dsi_row_count_validation** for the connection; the parameter takes effect immediately. However, the new setting affects the batch of replicated objects that Replication Server processes after you execute the command. Changing the setting does not affect the batch of replicated objects that Replication Server is currently processing.

See "Data server error handling," in Chapter 6, "Handling Errors and Exceptions" in the *Replication Server Administration Guide Volume 2*.

Display Table Names in Row Count Validation Error Message

Row count validation error messages display table names with Replication Server 15.6.

If you are using:

- Continuous mode log-order row-by-row replication – Replication Server logs and displays the table name, table owner name, and the number that identifies the output command that caused the transaction to fail. Replication Server logs only the first 30 bytes of the table name. You can enable the DSI_CHECK_ROW_COUNT_FULL_NAME trace to expand the maximum length of the table name that displays to 255 bytes.
- High volume adaptive replication (HVAR) or real-time loading (RTL) – Replication Server logs and displays the internal **join-update** and **join-delete** statements that result from HVAR and RTL compilation. You cannot obtain the specific command that caused the failed transaction since HVAR or RTL have already compiled the command as part of HVAR and RTL processing. The maximum length of the **join-update** and **join-delete** statements that can display is 128 bytes including the `"...\0"` tail string.

This example consists of:

- Primary site – `pdb1` primary database with a table named `ThisTableHasANameLongerThan30Characters` that has three columns and three rows.

id	name	age
1	John	40
2	Paul	38
3	George	37

- Replicate site – `rdb1` primary database with a table with the same name `ThisTableHasANameLongerThan30Characters` that has two rows with values of 1 and 3 for the `id` column.

If you execute this command against `pdb1`:

```
update ThisTableHasANameLongerThan30Characters set age = 20
```

the error messages appear differently for each type of replication mode. In:

- Continuous mode log-order row-by-row replication:
  ```
  I. 2010/06/07 01:30:21. DSI received Replication Server
  error #5185 which is mapped to WARN by error action mapping.
  W. 2010/06/07 01:30:21. WARNING #5185 DSI EXEC(103(1)
  ost_replnx6_61.rdb1) - /dsiexec.c(11941)
  Row count mismatch for the command executed on
  'ost_replnx6_61.rdb1'. The command impacted 0 rows but it
  should impact 1 rows.
  I. 2010/06/07 01:30:21. The error was caused by output
  command #3 of the failed transaction on table
  'dbo.ThisTableHasANameLongerThan30C'.
  ```

  Note: The table name is truncated to the default of 30 bytes.

  If you turn on the **DSI_CHECK_ROW_COUNT_FULL_NAME** trace to enable the maximum table name length of 255 bytes that the error message can display, the last line of the error message displays the full table name:
  ```
  I. 2010/06/07 02:22:55. The error was caused by output
  command #3 of the failed transaction on table
  'dbo.ThisTableHasANameLongerThan30Characters'.
  ```
- HVAR or RTL replication:
  ```
  W. 2010/06/07 02:06:56. WARNING #5185 DSI EXEC(103(1)
  ost_replnx6_61.rdb1) - i/hqexec.c(4047)
  Row count mismatch for the command executed on
  'ost_replnx6_61.rdb1'. The command impacted 1 rows but it
  should impact 2 rows.
  I. 2010/06/07 02:06:56. (HQ Error): update
  ThisTableHasANameLongerThan30Characters set age = w.age
  ```

```
from ThisTableHasANameLongerThan30Characters
t,#rs_uThisTab...
I. 2010/06/07 02:06:57. The DSI thread for database
'ost_replnx6_61.rdb1' is shutdown.
```

Seamless Upgrade

Replication Server version 15.6 offers a simplified process for upgrading routes.

With the simplified route upgrade process, you need not use the Replication Manager plug-in in Sybase Central™. See *Replication Server Configuration Guide > Upgrade or Downgrade Replication Server > Upgrading Replication Server > Commit a Local Site to a New Version Level > Replication Server Route Version > Upgrading Routes*.

Enhancements to Adaptive Server Replication Support

Replication Server 15.6 includes enhancements to support Adaptive Server replication.

In-Memory and Relaxed-Durability Databases

In Replication Server 15.5, you can use in-memory and relaxed-durability databases as the replicate database. With Replication Server 15.6, you can use in-memory and relaxed-durability databases as the primary database, and use database resynchronization and bulk materialization to restore in-memory and relaxed-durability databases.

Since an in-memory database exists only in cache, the data and database objects are not saved if the supporting host is shut down or the database fails. To restore in-memory or relaxed-durability replicate databases, you can use:

- Database resynchronization – follow the procedures in "Resynchronize One or More Replicate Databases Directly from a Primary Database." Ensure that you have enough disk space and time to perform a database dump and load, and that the period of time during which Replication Server skips transactions is acceptable. You can estimate the acceptable period of time by monitoring the segments in the outbound queue with **admin who, sqm**. See "**admin who**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual*.
- Bulk materialization – follow the instructions in "Resynchronizing a Replicate In-Memory or Relaxed-Durability Database with Bulk Materialization."

See "Support for in-memory and relaxed-durability databases," in Chapter 5, "Managing RepAgent and Supporting Adaptive Server" in the *Replication Server Administration Guide Volume 1* to set up replication for the first time or restore in-memory and relaxed-durability databases, using either a template database or a database dump.

See also

- *Resynchronize One or More Replicate Databases Directly from a Primary Database* on page 82

Resynchronizing a Replicate In-Memory or Relaxed-Durability Database with Bulk Materialization

You can use one of two bulk materialization methods to restore an in-memory or relaxed-durability database.

Prerequisites

Before you start bulk materialization, verify that the replication definitions and subscriptions exist.

Task

1. To quickly empty the inbound and outbound queues, deactivate the subscriptions that have the in-memory or relaxed-durability database:

   ```
   deactivate subscription subscription_name
   for {table_repdef_name | func_repdef_name |{publication pub_name |
   database replication definition db_repdef_name}
   with primary at dataserver.database}
   with replicate at dataserver.database
   go
   ```

   After you deactivate the subscriptions, Replication Server does not propagate all the transactions in the inbound queue to the outbound queue of the in-memory or relaxed-durability database.

   In contrast, when you drop a subscription, all the committed transactions that have been written into the inbound queue are distributed downstream of Replication Server. You can deactivate a subscription even if the DSI is not running because the deactivation only happens at the primary site. When the deactivate marker arrives at the outbound queue, you can see this entry in the Replication Server log:

   ```
   The deactivate marker for subscription subscription_name
   arrives at outbound queue: data_server_name.database_name.
   ```

   After the deactivate marker arrives at the outbound queue, use **sysadmin sqm_purge_queue** to purge the outbound queue at the replicate site to quickly empty the outbound queue. See *Replication Server Reference Manual > Replication Server Commands >* ***sysadmin sqm_purge_queue***.

2. Execute **check subscription** at both the primary and replicate Replication Servers to verify that the subscription status is DEFINED at the primary Replication Server and VALID at the Replication Server.
3. Use the "Simulate Atomic Materialization" or "Simulate Nonatomic Materialization" bulk materialization methods described in *Replication Server Administration Guide Volume 1 > Manage Subscriptions > Subscription Materialization Methods > Bulk Materialization*, to build the in-memory or relaxed-durability database. If you use:

- Simulate atomic materialization — execute steps 4 to 9
- Simulate nonatomic materialization — execute steps 4 to 13

Bulk Copy-in of image and Java Datatypes

With Replication Server 15.6 and Adaptive Server 15.0.3 ESD #1, you can replicate `image` and Java datatype columns in Adaptive Server tables using bulk copy-in. Replicate these datatypes to replicate databases and warm standby databases by specifying the datatypes in replication definitions, function replication definitions, and subscriptions.

Note: You must upgrade to Adaptive Server 15.0.3 ESD #1 or later to use bulk copy-in of `image` and Java datatypes.

See "DSI bulk copy-in," in Chapter 4, "Performance Tuning" in the *Replication Server Administration Guide Volume 2* to configure bulk copy-in.

New Features in Replication Server Version 15.5

Replication Server 15.5 introduces performance, usability, process, and database support enhancements.

Replication Server 15.5 Product Editions and Licenses

Replication Server 15.5 is released as two separate product editions—Enterprise Edition and Real-time Loading Edition—that bundle different base and optional features, and which require separate licences.

Note: You cannot use the "Replication Server – Real-Time Loading Edition" to replicate to Adaptive Server.

Table 8. Replication Server Product Edition Features and Licenses

Edition	Feature Type	Features	Description	License
Enterprise Edition	Base	Replication Server	Replication Server features, excluding Advanced Services Option, ExpressConnect for Oracle, and Real-time Loading.	REP_SERVER
	Optional	Advanced Services Option	Replication Server performance enhancements.	REP_HVAR_ASE
		ExpressConnect for Oracle	Provides Replication Server with the capability to connect directly to Oracle. See the Replication Server Options 15.5 documentation.	REP_EC_ORA
Real-Time Loading Edition	Base	Replication Server	Replication Server features, excluding Advanced Services Option, ExpressConnect for Oracle, and Real-time Loading.	REP_SERVER

Edition	Feature Type	Features	Description	License
		Real-time Loading (RTL).	Allows replication to Sybase® IQ.	REP_RTL_IQ
		Advanced Services Option	Replication Server performance enhancements.	REP_HVAR_ASE
	Optional	None		

Obtain valid SySAM licenses before you install Replication Server. Sybase Software Asset Management (SySAM) performs license administration and asset management tasks for Sybase products.See *Installation Guide > Before You Begin > Preinstallation Tasks > Obtaining a License*.

See also

- *Sybase IQ Replication Using Real-Time Loading* on page 100
- *Replication Server – Advanced Services Option* on page 104

Sybase IQ Replication Using Real-Time Loading

Sybase IQ replication using real-time loading (RTL) improves performance and is available as a Replication Server – Real-Time Loading Edition feature.

In versions earlier than 15.5, Replication Server sends each replication operation to the replicate database directly, row-by-row and in log order in a continuous replication mode.

Replication Server version 15.5 and later allows you to replicate to Sybase IQ from Adaptive Server using real-time loading. When replicating into Sybase IQ replicate databases with identical database schema, Replication Server achieves better performance than with the continuous replication mode. RTL uses these processes, which result in data reduction:

- Compilation – rearranges replicate data, by clustering it by each table, and each insert, update, and delete operation, and then compiling the operations into net-row operations.
- Bulk apply – applies the net result of the compilation operations in bulk using the most efficient bulk interface for the net result. Replication Server uses an in-memory net-change database to store the net row changes which it applies to the replicate database.

Instead of sending every logged operation, compilation removes all the intermediate operations and sends only the final states of a replicated transaction. Depending on the application, this generally means a much smaller amount of data is processed.

As Replication Server compiles and combines a larger number of transactions into a group, bulk operation processing improves; therefore, replication throughput and performance also improves.You can control the amount of data that is grouped together for bulk apply by adjusting group sizes.

See *Replication Server Heterogeneous Replication Guide > Sybase IQ as Replicate Data Server*.

License

Replication to Sybase IQ using RTL is available in the Real-Time Loading Edition product edition.

Database and Platform Support

You can use RTL to replicate into Sybase IQ 12.7 ESD #3 and later. You can achieve optimal performance using 64-bit hardware platforms.

Replication Server 15.5 supports replication to Sybase IQ only from Adaptive Server version 15.0.3 or version 15.5 and later as the primary database.

See also

- *Support for 64-bit Computing Platforms* on page 123
- *Replication Server 15.5 Product Editions and Licenses* on page 99

Enhancements to Heterogeneous Replication Support

Replication Server 15.5 extends support for heterogeneous databases.

Parallel DSI Support in a Heterogeneous Environment

You can configure Replication Server to apply transactions to the replicate data server in a heterogeneous environment using parallel Data Server Interface (DSI) threads. Applying transactions in parallel increases the speed of replication, yet maintains the serial order of the transactions as they are applied at the primary site.

Table 9. Support for Parallel DSI for Non-ASE Databases by Replication Server

Database	Internal Commit Control Method	External Commit Control Method
Oracle	Yes	No
Microsoft SQL Server	Yes	Yes
IBM DB2 UDB	Yes	Yes

See the *Replication Server Heterogeneous Guide* for detailed information about using parallel DSI for non-ASE databases.

New Serialization Method

wait_after_commit is a transaction serialization method that improves performance and data integrity with parallel DSI for heterogeneous replication.

In **wait_after_commit**, each thread waits to begin its first batch until the previous thread has completely committed. Sybase recommends that you use the **wait_after_commit** serialization method for databases that use multiversion concurrency control (MVCC) or optimistic concurrency control (OCC), such as an Oracle database. Otherwise, you can use **wait_for_commit** as the default method.

New and Updated Configuration Parameters

Replication Server 15.5 introduces **dsi_max_cmds_in_batch**, and updates **dsi_max_xacts_in_group** and **dsi_serialization_method** to support parallel Data Server Interface (DSI) processing.

Table 10. New Configuration Parameter

Parameter	Value	Default	Description
dsi_max_cmds_in_batch	Integer	100	Defines maximum number of source commands whose output commands can be batched. You must suspend and resume the connection for any change in the parameter to take effect. Range: 1 – 1000

Table 11. Updated Configuration Parameters

Parameter	Value	Default	Description
dsi_max_xacts_in_group	Integer	20	Specifies the maximum number of transactions in a group. Larger numbers may improve data latency at the replicate database. Range: 1 – 1000
dsi_serialization_method	**no_wait** **wait_for_start** **wait_for_commit** **wait_after_commit**	**wait_for_commit**	Specifies the method used to maintain serial consistency between parallel DSI threads when applying transactions to a replicate data server. In all cases, commit order is preserved.

Function-String Changes for Internal Commit Control for Adaptive Server

Replication Server uses the **rs_dsi_check_thread_lock** function to check whether the current DSI executor thread is blocking another replicate database process. In Replication Server 15.5, the **rs_dsi_check_thread_lock** function string has been modified to detect deadlocks.

See Chapter 4, "Replication Server System Functions" in the *Replication Server Reference Manual*.

Heterogeneous Warm Standby Support for Oracle

In Replication Server 15.5, you can create and maintain warm standby applications for Oracle database.

There are tasks that the Replication Server system administrator has to perform manually to create the warm standby setup for Oracle databases. See Chapter 12, "Managing Heterogeneous Warm Standby for Oracle" in the Replication Server Heterogeneous Replication Guide for complete information.

A new configuration parameter, **ra_standby**, has been added to support Oracle warm standby applications. This parameter identifies whether Replication Agent for Oracle works in a standby mode. See Chapter 2, "Configuration Parameters" in the *Replication Agent 15.5 Reference Manual*.

For Adaptive Server, administrators can continue using the **rs_init** utility to set warm standby environment. See the *Replication Server Administration Guide Volume 2*.

Product Compatibility

The table describes the additional replication components required for supporting the warm standby feature for Oracle.

Table 12. Product Compatibility for Oracle Warm Standby Support

Database Server Version	Replication Agent Version	ECDA Option Version	ExpressConnect Version
Oracle 10g, 11g	Replication Agent for Oracle 15.5	ECDA 15.0 ESD #3	ExpressConnect for Oracle 15.5

Trigger Control at the Oracle Replicate Database

You can control trigger firing at the session level or connection level each time a PL/SQL command is executed against an Oracle 10g or 11g replicate database. Controlling trigger execution at the replicate database eliminates duplication and inaccuracy that were caused in earlier versions due to the absence of trigger control at the replicate database.

The **RS_TRIGGER_CONTROL** package supports this feature and is automatically installed when a connection to the replicate Oracle database is created through connection profiles. The **rs_triggers_reset** system function has also been modified to support this feature, and you can

now set the **dsi_keep _triggers** connection parameter to off to disable triggers in Oracle. Re-create every trigger needed to be controlled at the replicate database, adding the trigger control statement at the beginning of your trigger action.

See “Settings for trigger firing,” in Chapter 10, “Oracle Replicate Data Server Issues” in the *Replication Server Heterogeneous Replication Guide*.

Performance Enhancements

Replication Server 15.5 includes several performance enhancements.

Replication Server – Advanced Services Option

Replication Server – Advanced Services Option is a separately licensed product option for Replication Server that contains Replication Server performance enhancements.

It is:

- Available as an option in the Replication Server – Enterprise Edition, as the separate REP_HVAR_ASE license.
  If you are using the Replication Server – Enterprise Edition, download the REP_HVAR_ASE license file from the *Sybase Product Download Center (SPDC)* to activate any of the enhancements in the Advanced Services Option.
- Bundled in the Replication Server – Real-Time Loading Edition.
  To activate Replication Server – Real-Time Loading Edition, download the Replication Server – Real-Time Loading Edition product edition license from the SPDC.

See also

- *Replication Server 15.5 Product Editions and Licenses* on page 99

High Volume Adaptive Replication

High-Volume Adaptive Replication (HVAR) uses compilation and bulk-apply processes that result in data reduction and achieve better performance compared to continuous replication mode, which sends each replication operation to the replicate database directly, row-by-row and in log order.

- Compilation – rearranges replicate data by clustering it by each table, and each insert, update, and delete operation, then compiling the operations into net-row operations.
- Bulk apply – applies the net result of the compilation operations in bulk using the most efficient bulk interface for the net result. Replication Server uses an in-memory net-change database to store the net row changes which it applies to the replicate database.

Instead of sending every logged operation, compilation removes all the intermediate operations and sends only the final states of a replicated transaction. Depending on the application, this generally means a much smaller amount of data is processed.

As Replication Server compiles and combines a larger number of transactions into a group, bulk operation processing improves; therefore, replication throughput and performance also

improves. You can control the amount of data that is grouped together for bulk apply by adjusting group sizes.

HVAR is especially useful for creating online transaction processing (OLTP) archiving and reporting systems where the replicate databases have the same schemas as the primary databases.

See "High Volume Adaptive Replication to Adaptive Server" in Chapter 4, "Performance Tuning" in the *Replication Server Administration Guide Volume 2*.

System Table Support

Replication Server uses the *rs_tbconfig* table to store support table-level configuration parameters, and the *ref_objowner* and *ref_objname* columns in the *rs_columns* table to support referential constraints.

See Chapter 8, "Replication Server System Tables" in the *Replication Server Reference Manual* for full table descriptions.

Database and Platform Support

HVAR supports replication into Adaptive Server 12.5 and later, and you can achieve optimal performance using 64-bit hardware platforms.

See also

- *Support for 64-bit Computing Platforms* on page 123

Enhanced DSI Efficiency

Enhanced Data Server Interface (DSI) efficiency improves performance by reducing data replication latency, which decreases the length of time that Replication Server waits for results from the replicate data server through the **ct_results** routine, and subsequently reduces the length of time the data server waits for Replication Server.

See "Enhanced DSI efficiency" in Chapter 4, "Performance Tuning" in the *Replication Server Administration Guide Volume 2* for configuration information.

Enhanced RepAgent Executor Thread Efficiency

Enhanced RepAgent executor thread efficiency improves performance by using the NRM thread to normalize and pack Log Transfer Language (LTL) commands in parallel with parsing by the RepAgent Executor thread.

Parallel processing by the NRM thread reduces the response time of the RepAgent executor thread. The NRM thread is a thread split from RepAgent executor thread.

See "Enhanced RepAgent Executor thread efficiency" in Chapter 4, "Performance Tuning" in the *Replication Server Administration Guide Volume 2* to enable the NRM thread and specify the memory available to the RepAgent Executor thread.

Enhanced Distributor Thread Read Efficiency

With Replication Server 15.5, the distributor (DIST) thread reads SQL statements directly from the Stable Queue Transaction (SQT) thread cache. This reduces the workload from SQT and the dependency between the two, and improves the efficiency of both SQT and DIST.

See “Enhanced distributor thread read efficiency” in Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2*.

Enhanced Memory Allocation

Enhanced memory allocation improves performance by allocating memory in larger chunks and therefore reducing the number of memory allocations.

See “Enhanced memory allocation” in Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2*.

Increased Queue Block Size

The queue block size has been increased allowing you to process more transactions in a single block.

The queue block size is the number of bytes in a contiguous block of memory used by stable queue structures. In earlier versions of Replication Server, the queue block size is fixed at 16KB.

Note: You must suspend incoming data flow and routes, and quiesce Replication Server before modifying the queue block size. After executing the command to set the block size, Replication Server automatically shuts down. Restart Replication Server for the new block size to take effect.

See “Increasing queue block size,” in Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2* for recommendations, prerequisites, and instructions for configuring the queue block size.

Dynamic SQL Enhancements

Replication Server 15.5 introduces several enhancements to dynamic SQL.

Dynamic SQL enhances performance by allowing the Replication Server Data Server Interface (DSI) module to prepare dynamic SQL statements at the replicate database and to run them repeatedly. See “Dynamic SQL for enhanced Replication Server performance,” in Chapter 4, “Peformance Tuning” in the *Replication Server Administration Guide Volume 2*.

Optimized Dynamic SQL Statement Execution

Replication Server 15.5 enhances the execution of dynamic SQL statements.

In versions earlier than 15.5, Replication Server generates the language command, the prepared statement, and the execute statement each time DSI executes dynamic SQL

statements, but Replication Server only uses the language command when the dynamic SQL command fails.

In version 15.5, Replication Server optimizes dynamic SQL statements by:

- Generating the language command only when the dynamic SQL command fails.
- Generating the prepared statement only once when the prepared statement is used for the first time.

replicate minimal columns clause with Dynamic SQL

In Replication Server 15.5, replication processing does not skip dynamic SQL even if the **replicate minimal columns** clause is enabled and Replication Server uses **replicate minimal columns** and dynamic SQL effectively at the same time.

In Replication Server versions earlier than 15.5, if the **replicate minimal columns** clause is used in a replication definition, the columns that are not changed are not available to the DSI, and dynamic SQL is skipped.

replicate_minimal_columns Extension

With version 15.5, Replication Server extends the **replicate_minimal_columns** parameter to connections in all situations, so that the Data Server Interface (DSI) can use the parameter to determine whether to use minimal columns when there is no replication definition, or when the replication definition does not contain the **replicate minimal columns** clause.

In Replication Server versions earlier than 15.5, you could only use **replicate_minimal_columns** in warm standby situations.

By default, **replicate_minimal_columns** is on for all connections. The **replicate_minimal_columns** setting for a connection overrides replication definitions set with the **replicate all columns** clause.

The behavior of the current replication environment could change when you set **replicate_minimal_columns** to on for a connection. Custom function strings at the replicate connection and trigger processes that rely on commands being sent to the replicate, even if no values are changed, could be affected. To restore the original behavior, set **replicate_minimal_columns** off for the connection.

For example, to enable **replicate_minimal_columns** for the connection to the *pubs2* database in the SYDNEY_DS data server, enter:

```
alter connection to SYDNEY_DS.pubs2
set replicate_minimal_columns to ‘on’
```

You can use **admin config** to display the **replicate_minimal_columns** setting.

Note: When you set **dsi_compile_enable** to on, Replication Server ignores what you set for **replicate_minimal_columns**.

Function-String Efficiency Improvements

Replication Server 15.5 includes enhancements to function-string processing commands, stored procedures, and a system table to allow you to identify specific function strings that do not need to be applied to replicate databases.

In versions earlier than 15.5, Replication Server executes all function strings for all replicate databases although many of these function strings, such as those without output commands, do not apply to non-ASE databases. Preventing these function strings from being executed reduces processing overhead and simplifies the replication environment.

Modification to Function-String Processing Commands

Replication Server 15.5 extends the scope of the **none** parameter to apply to all functions, and provides the flexibility to identify which function strings Replication Server can avoid executing on replicate databases.

In versions earlier than 15.5, the **none** parameter of the **alter function string** and **create function string** commands applies only to the **rs_writetext** function, and instructs Replication Server not to replicate a *text*, *unitext*, or *image* column value.

Use the **none** parameter to identify class-level and table-level function strings that do not have output commands. Replication Server does not execute these function strings on replicate databases.

See “Using output templates” in Chapter 2, “Customizing Database Operations” in the *Replication Server Administration Guide Volume 2*.

There is no change in the syntax of the **alter function string** and **create function string** commands. See Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual*.

Modification to Stored Procedures

Replication Server 15.5 extends the **rs_helpfstring** and **rs_helpclassfstring** stored procedures to support the enhancements to function string processing.

- **rs_helpfstring** – displays function strings for table-level functions, including those without output commands.
- **rs_helpclassfstring** – displays function strings for class-level functions, including those without output commands.

See Chapter 6, “Adaptive Server Stored Procedures” in the *Replication Server Reference Manual*.

Modification to rs_funcstrings System Table

Replication Server 15.5 adds the *0x08* bit to the *attributes* column in the *rs_funcstrings* table to support the function string enhancements.

See “rs_funcstrings,” in Chapter 8, “Replication Server System Tables” in the *Replication Server Reference Manual*.

Usability and Process Enhancements

Replication Server 15.5 introduces several usability and process enhancements.

Enhanced Replication Definition Change Request Process

Replication Server 15.5 includes enhancements for requesting changes to replication definitions that automatically coordinate the propagation of replication definition changes and data replication.

These enhancements make coordinating database schema changes and replication definition changes more convenient because there is no downtime for the primary database, and minimal or no downtime for the replicate database. When you issue a replication definition change request, Replication Server determines if there is a need to create a new replication definition version based on the type of change requested. If Replication Server creates a new replication definition version, primary updates before the replication definition change request automatically use the old replication definition version, while primary updates after the replication definition change request use the new replication definition version.

Without these enhancements, to coordinate schema changes and replication definition changes, you must quiesce primary updates, wait until all the data associated with a primary table or stored procedure is processed through the entire replication system, shut down Replication Agent, alter the primary schema, alter the replication definition, alter any customized function strings, wait for the changes to replicate, alter the replicate schema, and then restart Replication Agent and resume primary updates.

See “Replication definition change request process,” in Chapter 9, “Managing Replicated Tables” in the *Replication Server Administration Guide Volume 1* for the commands, procedures, and a user scenario for the enhanced replication definition change request process.

Product Compatibility

You can change replication definitions on primary databases for Adaptive Server, and for all versions of Microsoft SQL Server and Oracle that Replication Server supports. See the Replication Server Options documentation for supported versions.

Mixed-Version Support

If you execute the **alter replication definition** with the **drop** *column name* clause, and there is a subscription to the replication definition from a replicate site with a site version earlier than 1550, the primary Replication Server rejects the **alter replication definition** command.

Issuing any alter replication definition request with the **with DSI_suspended** parameter does not suspend any replicate DSI with site versions earlier than 1550.

List of Replication Definition Enhancements

With the enhanced replication definition, you can request replication definition changes directly at the primary database using the **alter replication definition**, **alter applied replication definition**, or **alter request function replication definition** commands, while making changes to the database schema.

You can:

- Issue a replication definition command directly from a primary database.
- Use an **alter replication definition** command to instruct Replication Server to suspend the target DSIs after Replication Server applies all data for the old replication definition version at the target database. This provides a window for you to alter the target schema and alter customized function strings before the data for the new replication definition version arrives.
- Verify that Replication Server can execute a replication definition request successfully by executing the request without changing any data.
- Drop columns from a replication definition using **alter replication definition**.
- Instruct Replication Server to skip a failed replication definition request sent by a Replication Agent. When a replication definition command fails at the primary Replication Server, Replication Agent shuts down. If you restart Replication Agent, the failed command executes again unless Replication Server skips the command.

System Table Changes

To support the enhanced replication definition change process, Replication Server includes changes to the *rs_columns*, *rs_locater*, and *rs_objects* system tables.

System Table	Description of Change
rs_columns	*version* column added.
rs_locater	*C*, *F*, and *S* values added to *type* column.
rs_objects	*active_inbound*, *attributes2*, and *version* columns added.

See Chapter 8, “Replication Server System Tables” in the *Replication Server Reference Manual*.

Replication Task Scheduling

Replication Server 15.5 lets you schedule replication tasks.

For example, you can report on a specific state of the replicate database while the replicate database is not receiving data from the primary database. You can schedule replication to happen only during specific night hours, so that the processing of the next day does not change the replicate database, and reporting can occur on the data from the previous day. You can do this by creating schedules to suspend and resume connections to the replicate database at specific times of the day. The schedules you create are stored in the *rs_schedule* and *rs_scheduletxt* system tables.

See "Scheduling replication tasks" in Chapter 13, "Scheduling Replication Tasks" in the *Replication Server Administration Guide Volume 1* and Chapter 8, "Replication Server System Tables" in the *Replication Server Reference Manual*.

Replication Delay

Replication Server 15.5 lets you delay replication by a fixed period of time.

You can use a replicate database as a failback system by delaying updates for a certain amount of time behind the primary database to recover from any human error committed on the primary database.

See "Delaying replication," in Chapter 13, "Scheduling Replication Tasks" in the *Replication Server Administration Guide Volume 1*.

Replicate Database Resynchronization

Database resynchronization allows you to rematerialize your replicate database and resume further replication without data loss or inconsistency, and without forcing a quiesce of your primary database.

Replication Agent Support

The full functionality of database resynchronization, such as automatic generation of the resync marker, requires Replication Agent support. Replication Agent 15.5 for Oracle supports the full functionality of database resynchronization. See Chapter 13, "Resynchronizing Oracle Replicate Databases" in the *Replication Server Heterogeneous Replication Guide* and the Replication Agent documentation.

RepAgent, the Replication Agent for Adaptive Server, is scheduled to support the full functionality of database resynchronization in a version later than Adaptive Server 15.5. See "Resynchronizing replicate databases," in Chapter 7 in "Replication System Recovery" in the *Replication Server Administration Guide Volume 2* to resynchronize Adaptive Server databases without support from RepAgent.

Product Compatibility

The following table lists the versions of Oracle, Replication Agent for Oracle, ECDA Option for Oracle, and ExpressConnect for Oracle that support the resynchronization of Oracle databases. With Replication Server Options 15.5, ExpressConnect for Oracle replaces ECDA Option for Oracle.

See the Replication Server Options documentation and the *Replication Server Heterogeneous Replication Guide*.

Table 13. Product Compatibility for Resynchronizing Oracle Databases

Database Server Version	Replication Agent Version	ECDA Option Version	ExpressConnect Version
Oracle 10g, 11g	Replication Agent for Oracle 15.5	ECDA 15.0 ESD #3	ExpressConnect for Oracle 15.5

System Table Support

In the *rs_databases* table, the datatype of *dist_status* and *src_status* columns has been changed from *tinyint* to *cs_int*, and the “0x100 – waiting for a resync marker” status has been added to *dist_status*.

Resynchronizing Replicate Databases

Obtain a datra dump from a trusted source and apply the dump to the target database you want to resynchronize.

1. Stop replication processing by suspending Replication Agent.
2. Place Replication Server in resync mode.

   In resync mode, Replication Server skips transactions and purges replication data from replication queues in anticipation of the replicate database being repopulated from a dump taken from the primary database or trusted source.
3. Restart Replication Agent and send a resync database marker to Replication Server to indicate that a resynchronization effort is in progress.
4. Obtain a dump from the primary database.

   When Replication Server detects a dump marker that indicates completion of the primary database dump, Replication Server stops skipping transactions and can determine which transactions to apply to the replicate database.
5. Apply the dump to the replicate database.
6. Resume replication.

Row Count Validation Changes

Replication Server 15.5 changes the default error actions for the 5185 and 5187 error numbers from “warn” to “stop replication” and adds 5203 for Replication Server error classes.

Table 14. New and Changed Errors

server_error	Error Message	Default Error Action	Description
5185	`Row count mismatch for the command executed on ‘dataserver.database’. The command impacted x rows, but it should impact y rows.`	**stop_replication**	This message appears if the affected number of rows is different from the expected number of rows, after a command that is not part of SQL statement replication, or a stored procedure, or a row change with autocorrection enabled is sent to the data server.
5187	`Row count mismatch for the autocorrection delete command executed on ‘dataserver.database’. The command deleted x rows, but it should delete y rows.`	**stop_replication**	This message appears if the affected number of rows is different from the expected number of rows, after a delete command is sent to the dataserver, and if autocorrection is enabled.
5203	`Row count mismatch on ‘dataserver.database’. The delete command generated by dsi_command_convert deleted x rows, whereas it should delete y rows.`	**stop_replication**	This message appears if the number of rows deleted is different from the expected number of rows to be deleted.

Use the **assign action** command at the primary site for the Replication Server error class to override the default error action.

See Chapter 6, “Handling Errors and Exceptions” in the *Replication Server Administration Guide Volume 2*.

For details about commands, parameters, stored procedures, and system tables, see the *Replication Server Reference Manual*.

See also

- *Row Count Validation for Non-SQL Statement Replication* on page 141

Enhanced alter error Class

You can use **alter error class** to modify a Replication Server error class.

Use the **replication server** parameter in the **alter error class** command syntax to specify a Replication Server error class. For details, see the *Replication Server Reference Manual* and Chapter 6, "Handling Errors and Exceptions" in the *Replication Server Administration Guide Volume 2*.

Toolset for Implementing a Reference Replication Environment

Replication Server 15.5 includes a toolset for quickly setting up a reference implementation of Adaptive Server to Adaptive Server and Oracle to Oracle replication using the products available in your environment. A reference replication environment lets you collect statistics to identify performance issues, and demonstrates Replication Server features and functionalities.

Use the toolset to:

1. Build Replication Server and the primary and replicate databases.
2. Configure the replication environment.
3. Perform simple transactions on the primary database and replicate the changes by database-level replication.
4. Collect statistics and monitors counters from the replication processing in step 3.
5. Clean up the reference replication environment.

See Appendix D, "Implementing a Reference Replication Environment" in the *Replication Server Administration Guide Volume 2* to build, configure, and use a reference replication environment.

Note: The reference implementation builds a replication environment containing a single Replication Server, primary database server, and replicate database server. You cannot configure the reference environment topology for multiple replication system components.

Platform Support

Reference implementation is available for all platforms that Replication Server 15.5 supports. However, to set up the reference environment on any Microsoft Windows platform that Replication Server supports, you must use Cygwin to run the reference implementation scripts. See the *http://www.cygwin.com/*.

Required Components for Adaptive Server

A reference implementation environment for Adaptive Server to Adaptive Server replication requires supported versions of Replication Server and Adaptive Server.

Table 15. Supported product component versions for Adaptive Server reference implementation

Replication Server	Adaptive Server
15.5	15.0.3, 15.5

Required Components for Oracle

A reference implementation environment for Oracle to Oracle replication requires supported versions of Replication Server, Oracle, Replication Agent for Oracle, and ECDA Option for Oracle.

Table 16. Supported Product Component Versions for Oracle Reference Implementation

Replication Server	Oracle	Replication Agent for Oracle	ECDA Option for Oracle
15.5	10.2	15.2	15.0 ESD #3

Enhanced admin who Command

Replication Server 15.5 lets you specify connection identifiers when you execute **admin who** on any thread module.

Instead of seeing information for all the connections of a thread module, you can show only the **admin who** execution results of a specific connection by specifying connection identifiers for these thread modules:

- DIST – Distributor
- DSI – Data Server Interface
- RSI – Replication Server Interface
- SQM – Stable Queue Manager
- SQT – Stable Queue Transaction

If you specify connection identifiers and Replication Server cannot find information that fulfills the criteria, the output does not display any record.

See “admin who,” in Chapter 3, in the *Replication Server Reference Manual* for the full syntax and examples.

Note: You cannot use the **no_trunc** option if you specify connection identifiers.

New Columns for the DIST and DSI Thread Modules

admin who displays additional columns for the DIST and DSI thread modules.

Table 17. Additional Columns for DIST and DSI Thread Modules

Thread	Column Name	Description	Value
DIST	*RSTicket*	The number of **rs_ticket** subcommands that have been processed by a DIST thread, if the Replication Server **stats_sampling** parameter is on.	Minimum: 0 Maximum: 2^{63}-1 Default: 0
DIST	*SqtMaxCache*	Maximum SQT cache memory for the database connection, in bytes.	The default, 0, means that the current setting of **sqt_max_cache_size** is used as the maximum cache size for the connection.Default: 0
DSI	*RSTicket*	The number of **rs_ticket** subcommands that have been processed by a DSI queue manager, if the Replication Server **stats_sampling** parameter is on.	Minimum: 0 Maximum: 2^{63}-1 Default: 0
DSI	*dsi_rs_ticket_report*	Determines whether to call function string **rs_ticket_report**. **rs_ticket_report** is invoked when **dsi_rs_ticket_report** is set to on.	on or off Default: off

See “admin who,” in Chapter 3, in the *Replication Server Reference Manual* for the full syntax and examples.

Database Generation Number Reset

Each primary database in a replication system includes a database generation number. This number is stored both in the database and in the RSSD of the Replication Server that manages the database.

Any time you load a primary database for recovery, you must change the database generation number, as instructed in the recovery procedure you are using.

The maximum value for the database generation number is 65,535. Sybase recommends that you do not increase the number to high values unless absolutely necessary. In Replication Server 15.5 and later, you can reset the database generation number to 0 before it reaches the maximum of 65,535. In Replication Server versions earlier than 15.5, you must rebuild the replication environment after resetting the database generation number.

See “Resetting database generation numbers,” in Chapter 7, “Replication System Recovery” in the *Replication Server Administration Guide Volume 2*.

Insertion of rs_ticket Markers into the Inbound Queue

Replication Server 15.5 introduces a system command to identify performance issues in data replication.

sysadmin issue_ticket injects an **rs_ticket** marker into the inbound queue, bypassing the need for the RepAgent on the primary database to process the ticket. **rs_ticket** processes from this point by appending the system time at the end of the marker as it passes through certain modules on the Replication Server. The information gathered by this marker is stored in the *rs_ticket_history* table in the replicated database.

See Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual*.

Changes to Default Settings and Reserved Words

Replication Server 15.5 includes changes to default settings and values to accommodate the enhancements in Replication Server and improve performance.

Changes to Parameter Default Values

Except for **memory_limit** and **smp_enable**, if you upgrade to Replication Server 15.5, the only values that are set to the new defaults are those that used the defaults in the earlier version.

Table 18. Changes to Parameter Default Values

Parameter	Old Value	New Value	Downgrading from Version 15.5
exec_cmds_per_timeslice	5	2,147,483,647	Downgrading does not change the value you have set.
init_sqm_write_delay	1000 milliseconds	100 milliseconds for all platforms	Downgrading does not change the value you have set.
init_sqm_write_max_delay	10,000 milliseconds	1000 milliseconds for all platforms	Downgrading does not change the value you have set.
memory_limit	80MB	2,047MB Upgrading increases the value to the new default if the earlier value was less than 2,047MB.	If the value you set is larger than 2,047MB, downgrading resets the value to 2,047MB to protect against overflow.

Parameter	Old Value	New Value	Downgrading from Version 15.5
smp_enable	off	on Upgrading does not change the setting to on if your original setting was off.	Downgrading does not change the value you have set.
sqt_max_cache_size	1,048,576 bytes (1MB)	• 32-bit platforms: 1,048,576 bytes (1MB) • 64-bit platforms: 20,971,520 bytes (20MB)	If the value set is larger than 2,147,483,647 bytes, downgrading resets the value to 2,147,483,647 bytes to protect against overflow.
sts_full_cache_*system_table_name* for these system tables: *rs_columns*, *rs_functions*, *rs_objects*	off – not fully cached	on – fully cached	Downgrading does not change your settings.

See Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual* for descriptions of the parameters, examples and usage information.

Changes to the RSSD Locking Schema

To reduce contention and improve performance, the default locking schema for the system tables in the Replication Server System Database (RSSD) is row-level locking when you install or upgrade to version 15.5.

The locking schema does not change when you downgrade from version 15.5.

There is no change in the default locking schema for the Embedded Replication Server System Database (ERSSD), which is row-level locking.

Reserved Words

Version 15.5 adds to the list of reserved Replication Server keywords.

See “Reserved words,” in Chapter 2, “Topics” in the *Replication Server Reference Manual* which is updated with new reserved Replication Server keywords.

Enhancements to Adaptive Server Replication Support

Replication Server 15.5 includes enhancements to support Adaptive Server replication.

bigdatetime and bigtime Replication

Replication Server 15.5 supports replication of the Adaptive Server 15.5 *bigdatetime* and *bigtime*. Replicate these datatypes to replicate databases and warm standby databases by specifying the datatypes in replication definitions, function replication definitions, and subscriptions.

bigdatetime and *bigtime* allow Adaptive Server to store data and time data up to microsecond precision. *bigdatetime* corresponds to the *TIMESTAMP* datatype, and *bigtime* corresponds to the *TIME* datatype in Sybase IQ and Sybase SQL Anywhere.

See "Support for *bigdatetime* and *bigtime* datatypes," in Chapter 5, "Managing RepAgent and Supporting Adaptive Server" in the *Replication Server Administration Guide Volume 1* to use *bigdatetime* and *bigtime*.

Mixed-Version Information

Only Adaptive Server versions 15.5 and later support *bigdatetime* and *bigtime*. If the primary data server is at least Adaptive Server 15.5, and:

- Primary and replicate Replication Server are version 15.5 or later, and the replicate Adaptive Server does not support the datatypes, create a replication definition that contains a mapping for each of the two datatypes to the *varchar* datatype. Alternatively, use the *varchar* datatype instead of the two datatypes in the replication definition.
- Primary Replication Server is version 15.5 or later, and the replicate Replication Server and Adaptive Server do not support the datatypes, use the *varchar* datatype instead of the two datatypes in the replication definition.
- Primary and replicate Replication Server, and the replicate Adaptive Server do not support the datatypes, RepAgent automatically sends the *varchar* datatype to Replication Server.

See also

- *Enabling bigdatetime and bigtime Replication* on page 125

Deferred Name Resolution

Replication Server 15.5 supports the Adaptive Server 15.5 deferred name resolution feature. Deferred name resolution lets you create stored procedures in Adaptive Server without resolving the objects used internally by these stored procedures.

Adaptive Server postpones the object resolution phase until you first execute the stored procedure in Adaptive Server. Stored procedures execute normally when you execute them after the first time. See "Deferred Name Resolution for User-Defined Stored Procedures" in the *Adaptive Server Enterprise 15.5 New Features Summary*.

Replication Server Issues

In Replication Server versions earlier than 15.5, you can set up a warm standby application and enable **sp_reptostandby** at the active database to allow replication of supported data definition language (DDL) commands to the standby database.

However, on a standby database or a replicate database in a non-warm standby environment, you cannot create a stored procedure that references a temporary table, because Replication Server does not replicate temporary tables. The create stored procedure process must resolve the objects used internally by the stored procedure. However, there is no temporary table in the replicate or standby database, therefore, Replicate Server does not create the stored procedure in the replicate or standby database.

With support for deferred name resolution, Replication Server 15.5 allows the replication of stored procedures that refer to temporary tables, tables that do not exist, and procedures that do not exist, to replicate or standby databases.

See "Deferred name resolution," in Chapter 5, "Managing RepAgent and Supporting Adaptive Server" in the *Replication Server Administration Guide Volume 1* to configure deferred name resolution in Replication Server.

SQL Statement Replication Threshold Setting

With Adaptive Server 15.0.3 ESD #1 and later, you can set the threshold at the database level or session level to trigger SQL statement replication without having to set the threshold on individual tables. Replication Server 15.5 supports this new threshold setting.

The threshold set at the session level overrides the threshold at the table level and database level, and the threshold set for any table overrides the threshold set at the database level. Earlier versions of Replication Server allowed you to set the threshold only at the table level.

See "Setting SQL statement replication threshold," in Chapter 4, "Performance Tuning" in the *Replication Server Administration Guide Volume 2*.

Incremental Data Transfer

With Adaptive Server 15.5, you can transfer data incrementally from a table instead of transferring an entire table from one Adaptive Server to another. Replication Server supports the data definition language related to the Adaptive Server 15.5 incremental data transfer feature, and replication proceeds normally for data modification operations performed on a replicate table marked for incremental transfer.

If you load a replicate table using the **transfer table** command, and the table has a unique index command and the data on the incremental transfer already exists on the table, Adaptive Server internally converts an **insert** command into an **update** command.

The **transfer table** command applies only to the data server and database where you initiated the transfer the first time.

If you mark tables for incremental transfer in the active database within a warm standby or multisite availability (MSA) environment, and then switch to the standby database if the active database terminates, incremental data transfer may not resume correctly at the standby database. Unlike the active database, the standby database does not have a record of the incremental data transfer activity. Therefore, you must also initialize the incremental data transfer on the standby database.

See Chapter 8, “Adding, Changing, Transferring, and Deleting Data,” in the *Adaptive Server Enterprise Transact-SQL Users Guide*.

In-Memory and Relaxed-Durability Databases

Adaptive Server 15.5 introduces in-memory and relaxed-durability databases.

In-memory databases reside entirely in cache and do not use disk storage for data or logs, and therefore do not require disk I/O. This results in potentially better performance than a traditional disk-resident database, as well as other advantages. However, since an in-memory database exists only in cache, you cannot recover the database if the supporting host is shut down or the database fails.

With relaxed-durability databases, Adaptive Server extends the performance benefits of an in-memory database to disk-resident databases. Disk-resident databases perform writes to disk, and ensure that the transactional ACID (atomicity, consistency, integrity, and durability) properties are maintained. A traditional disk-resident database operates at full durability to guarantee transactional recovery from a server failure. Relaxed-durability databases trade the full durability of committed transactions for enhanced runtime performance for transactional workloads. A relaxed-durability database created with the **no_recovery** level is similar to an in-memory database: you cannot recover data or logs if the server terminates or is shut down. You can also create a relaxed-durability database with the **at_shutdown** level, where transactions are written to disk if there is a proper shutdown of the database.

See the *Adaptive Server Enterprise In-Memory Database Users Guide*.

Replication Server Support

Replication Server supports as the replicate database the in-memory databases and relaxed-durability databases set with durability at **no_recovery**.

The primary database must be a traditional full-durability, disk-resident database. For convenience, this document refers to relaxed-durability databases with durability set to **non_recovery** as “relaxed-durability databases.”

You can initialize an in-memory and relaxed-durability database as a new replicate database by obtaining data, object schema, and configuration information from one of:

- A template database that retains basic information.
- A database dump from another database. Load the dump to the target in-memory database or relaxed-durability database.

  The dump source database can be another in-memory database, relaxed durability database, or a traditional disk-resident database.

In-memory and relaxed-durability databases lose their object definition, data, and RepAgent configuration once the host data server shuts down or restarts. You must reinitialize the in-memory or relaxed-durability database from the template or database dump from a source database.

See “Support for in-memory and relaxed-durability databases,” in Chapter 5, “Managing RepAgent and Supporting Adaptive Server” in the *Replication Server Administration Guide Volume 1* to configure an in-memory or a relaxed-durability database as a replicate database.

Minimal DML Logging and Replication

To optimize the log records that are flushed to the transaction log on disk, Adaptive Server can perform minimal to no logging when executing some data manipulation language (DML) commands—**insert**, **update**, **delete**, and **slow bcp**—on all types of low-durability databases, such as in-memory databases and relaxed-durability databases set with durability of **at_shutdown** or **no_recovery**.

You can perform minimal logging for DMLs on a per-database, per-table, and session-specific basis. See “Minimally-logged DML” in the *Adaptive Server Enterprise In-Memory Database Users Guide*.

Note: Minimal DML logging session-level settings take precedence over database-level settings and table-level settings.

Replication Server Support

As replication uses full logging, replication and the minimal DML logging feature in Adaptive Server 15.5 are incompatible at the same level, such as the database level or table level. However, you can take advantage of the performance enhancements from minimal DML logging on some tables while allowing replication on others, as minimal DML logging and replication can coexist at different levels. See “Minimal DMLlogging and replication,” in Chapter 5, “Managing RepAgent and Supporting Adaptive Server,” in the *Replication Server Administration Guide Volume 1* for scenarios that result in incompatibility between replication and minimal DML logging.

Mixed-Version Environments

If a replication system domain has Replication Server 15.5 and later, the system version, and all site and route versions in the replication system domain must be version 12.6 and later.

You must upgrade Replication Server to version 12.6 or later, set the site version to 12.6 or later, and upgrade routes to 12.6 or later, before you can upgrade to version 15.5.

See “Upgrading or Downgrading Replication Server” in the *Replication Server Configuration Guide*.

Newly Supported Operating Systems

Replication Server 15.5 introduces support to several operating systems.

- Microsoft Windows Server 2008 R2

- Microsoft Windows 7
- SuSe Linux Enterprise Server SLES 11

Support for 64-bit Computing Platforms

Replication Server 15.5 supports 64-bit computing platforms, which provide Replication Server with a large amount of virtual memory space and removes the maximum memory constraint of 2GB.

In addition, all the available Replication Server counters are now defined as 64-bit, which allows high-precision computations in Replication Server.

See Chapter 3, “Upgrading or Downgrading Replication Server” in the *Replication Server Configuration Guide* to migrate to 64-bit platforms and Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2* to configure support for 64-bit platforms.

Changes to Replication Server Configuration Parameters

Changes to certain configuration parameters impact performance on 32-bit Replication Server and 64-bit Replication Server.

Table 19. Replication Server Configuration Parameters

Parameter	Description	Valid Range for 32-Bit (in Bytes)	Valid Range for 64-Bit (in Bytes)
dsi_sqt_max_cache_size	The maximum Stable Queue Transaction (SQT) cache size for the database connection. The default, 0, means the current setting of the **sqt_max_cache_size** parameter is used as the maximum cache size for the connection.	Min: 0 Max: 2147483647	Min: 0 Max: 2251799813685247
dist_sqt_max_cache_size	The maximum Stable Queue Transaction (SQT) cache size for the DIST connection. The default, 0, means the current setting of the **sqt_max_cache_size** parameter is used as the maximum cache size for the connection.	Min: 0 Max: 2147483647	Min: 0 Max: 2251799813685247
sqt_max_cache_size	Maximum SQT interface cache memory, in bytes.	Min: 0 Max: 2147483647	Min: 0 Max: 2251799813685247

Changes to memory_limit Configuration Parameter

Changes to the **memory_limit** configuration parameter impacts performance on 32-bit Replication Server and 64-bit Replication Server.

Table 20. memory_limit Configuration Parameter

Parameter	Description	Valid Range for 32-Bit	Valid Range for 64-Bit
memory_limit	The maximum total memory the Replication Server can use, in megabytes.Values for several other configuration parameters are directly related to the amount of memory available from the memory pool indicated by **memory_limit**. These include **fstr_cachesize**, **md_source_memory_pool**, **queue_dump_buffer_size**, **sqt_max_cache_size**, **sre_reserve**, and **sts_cachesize**.	Min: 0 Max: 2047	Min: 0 Max: 2147483647

New Feature in Replication Manager 15.5

Replication Manager 15.5 supports bigdatetime and bigtime datatypes.

Enabling bigdatetime and bigtime Replication

Enable replication of *bigdatetime* and *bigtime* datatypes included with Adaptive Server 15.5 using Replication Manager 15.5.

You can replicate *bigdatetime* and *bigtime* datatypes to replicate databases and warm standby databases by specifying the datatypes in replication definitions, function replication definitions, and subscriptions.

On the Columns tab of the Add New Table Replication Definition dialog box, select *bigdatetime* or *bigtime* from the Replication Definition list in the Datatype area.

See also

- *bigdatetime and bigtime Replication* on page 119

New Features in Replication Server Version 15.2

Replication Server 15.2 introduces DSI bulk copy-in, non-blocking commit, quoted identifiers, Replication Server gateway, row count validation for non-SQL statement replication, SQL statement replication, and non-Adaptive Server error class support. Replication Server 15.2 also includes enhancements to non-Adaptive Server replication.

Support for DSI Bulk Copy-in

Replication Server version 15.2 introduces support for bulk copy-in to improve performance when replicating large batches of **insert** statements on the same table in Adaptive Server® Enterprise 12.0 and later.

In versions 15.1 and earlier, when Replication Server replicates data to Adaptive Server, Replication Server forms a SQL **insert** command, sends the command to Adaptive Server, and waits for Adaptive Server to process the row and send back the result of the operation. This process affects Replication Server performance when large batches of data are replicated, such as in end-of-day batch processing or trade consolidation.

Replication Server 15.2 implements bulk copy-in in Data Server Interface (DSI), the Replication Server module responsible for sending transactions to replicate databases, using the Open Client™ Open Server™ Bulk-Library.

Note: Bulk copy-in is supported only for Adaptive Server databases. If you turn on DSI bulk copy-in and the replicate database is not Adaptive Server, DSI shuts down with an error.

For information about the Open Client Open Server Bulk-Library, see the *Open Client and Open Server Common Libraries Reference Manual*.

Enhanced Subscription Materialization

Bulk copy-in also improves the performance of subscription materialization. When **dsi_bulk_copy** is on, Replication Server uses bulk copy-in to materialize the subscriptions if the number of **insert** commands in each transaction exceeds **dsi_bulk_threshold**.

Note: In normal replication, bulk operation is disabled for a table if **autocorrection** is on. However, in materialization, bulk operation is applied even when **autocorrection** is enabled, if **dsi_bulk_threshold** is reached and the materialization is not a nonatomic subscription recovering from failure.

For more information about subscription materialization, see *Replication Server Administration Guide Volume 1*.

New Connection Parameters

These new database connection parameters control bulk operations in DSI.

Parameter	Description
dsi_bulk_copy	Turns the bulk copy-in feature on or off for a connection. If **dynamic_sql** and **dsi_bulk_copy** are both on, DSI applies bulk copy-in. Dynamic SQL is used if bulk copy-in is not used. Default: off.
dsi_bulk_threshold	The number of **insert** commands that, when reached, triggers Replication Server to use bulk copy-in. When Stable Queue Transaction (SQT) encounters a large batch of **insert** commands, it retains in memory the number of **insert** commands specified to decide whether to apply bulk copy-in. Because these commands are held in memory, Sybase suggests that you do not configure this value much higher than the configuration value for **dsi_large_xact_size**. Minimum: 1 Default: 20

Usage

To set the values of **dsi_bulk_copy** and **dsi_bulk_threshold**, use:

- **alter connection** to change the bulk copy-in connection parameters at the connection level:

```
alter connection to dataserver.database
set {dsi_bulk_copy | dsi_bulk_threshold} to value
```

- **configure replication server** to change the server defaults:

```
configure replication server
set {dsi_bulk_copy | dsi_bulk_threshold} to value
```

To check the values of **dsi_bulk_copy** and **dsi_bulk_threshold**, use **admin config**.

When **dsi_bulk_copy** is on, SQT counts the number of consecutive **insert** statements on the same table that a transaction contains. If this number reaches the **dsi_bulk_threshold**, DSI:

1. Bulk-copies the data to Adaptive Server until DSI reaches a command that is not **insert** or that belongs to a different replicate table.
2. Continues with the rest of the commands in the transaction.

Adaptive Server sends the result of bulk copy-in at the end of the bulk operation, when it is successful, or at the point of failure.

Note: The DSI implementation of bulk copy-in supports multistatement transactions, allowing DSI to perform bulk copy-in even if a transaction contains commands that are not part of the bulk copy.

New Counters for Bulk Copy-in

New counters have been added for bulk copy-in.

Counter	Description
DSINoBulkDatatype	The number of bulk operations skipped due to the data containing datatype is incompatible with bulk copy-in.
DSINoBulkFstr	The number of bulk operations skipped due to tables that have customized function strings for **rs_insert** or **rs_writetext**.
DSINoBulkAutoc	The number of bulk operations skipped due to tables that have **autocorrection** enabled.
DSIEBFBulkNext	The number of batch flushes that executed because the next command is a bulk copy.
DSIEBulkSucceed	The number of times the Data Server Interface executor (DSI/E) invoked **blk_done(CS_BLK_ALL)** at the target database.
DSIEBulkCancel	The number of times DSI/E invoked **blk_done(CS_BLK_CANCEL)** at the target database.
DSIEBulkRows	The number of rows that DSI/E sent to the replicate data server using bulk copy-in.
BulkTime	The amount of time, in milliseconds, that DSI/E spent in sending data to the replicate data server using bulk copy-in.

Limitations

There are certain instances when DSI does not use bulk copy-in, or when the bulk-copy-in feature is not supported.

The Replication Server DSI does not use bulk copy-in when:

- Autocorrection is on and the data is not part of subscription materialization.
- **rs_insert** has a user-defined function string.
- *text* column has a user-defined function string for **rs_writetext** with output `none` or `rpc`.
- The data row contains *opaque* datatype or a user-defined datatype (UDD) that has an *rs_datatype.canonic_type* value of 255.
- The data row contains an *image* or a Java datatype.

The bulk-copy-in feature is not supported under the conditions listed below. In these instances, disable bulk copy-in.

- The replicate database is not Adaptive Server. In this case, if the DSI bulk copy-in is enabled, DSI terminates with an error message.

- The data size changes between Replication Server and the replicate Adaptive Server character sets, and the datarow contains text columns. In this case, if the DSI bulk copy-in is enabled, DSI terminates with this message:

  ```
  Bulk-Lib routine 'blk_textxfer' failed.
  Open Client Client-Library error: Error: 16843015,
  Severity 1 -- 'blk_textxfer(): blk layer: user
  error: The given buffer of xxx bytes exceeds the
  total length of  the value to be transferred.'
  ```
- The *owner.tablename* length is larger than 255 bytes and the replicate database is earlier than version Adaptive Server 15.0.3 Interim Release. If the DSI bulk copy-in is enabled, Replication Server terminates with this message:

  ```
  Bulk-Lib routine 'blk_init' failed.
  ```

  To specify not to use bulk copy-in when *owner.tablename* length is larger than 255 bytes:

  1. Turn trace on:

     ```
     trace "on", rsfeature, ase_cr543639
     ```
  2. Add this to the Replication Server configuration file:

     ```
     trace=rsfeature,ase_cr543639
     ```

Other limitations:

- Unlike the **insert** command, bulk copy-in does not generate timestamps; NULL values are inserted to the *timestamp* column if the *timestamp* column is not included in the replication. Either disable bulk copy-in, or set up your replication definition to include the *timestamp* column.
- *Text* and *image* columns are always logged, even if you change the **writetext** function string to **no log**.
- Bulk copy does not invoke **insert** trigger in Adaptive Server.
- The configuration parameter **send_timestamp_to_standby** has no effect on bulk copy-in. *timestamp* data is always replicated.

Non-blocking Commit

Replication Server 15.2 includes non-blocking commit, which uses the delayed commit feature in Adaptive Server to improve replication performance.

Note: To use non-blocking commit, you must use Sybase Enterprise Connect™ Data Access 15.0 ESD #3 (ECDA) or later.

Adaptive Server Delayed Commit Feature

Adaptive Server 15.0 and later includes the delayed commit feature designed to improve performance by delaying the commit phase of a transaction.

The commit phase includes writing log records of the transaction to disk and then notifying the client application of the transaction status. With delayed commit, Adaptive Server notifies the client application of a successful commit before writing the corresponding transaction log to

disk. This delay in writing to disk reduces contention on the last and active log page and thus improving performance.

However, the last page of a transaction log can be lost, if Adaptive Server terminates or if you shut down Adaptive Server using **shutdown with no wait**.

See "Using **delayed_commit** to determine when log records are committed," in Chapter 11, "Developing a Backup and Recovery Plan" in the *Adaptive Server Enterprise 15.0 System Administration Guide: Volume 2* and the **delayed_commit** parameter of the **set** command in "set," in Chapter 1, "Commands" in the *Adaptive Server Enterprise 15.0 Reference Manual: Commands*.

dsi_non_blocking_commit Configuration Parameter

The **dsi_non_blocking_commit** configuration parameter extends the period of time Replication Server saves messages after a commit.

Extending the save period requires a larger stable queue. See "Stable queues," in Chapter 2, "Replication Server Technical Overview" in the *Replication Server Administration Guide Volume 1*.

Use **alter connection** to configure **dsi_non_blocking_commit** for a database connection:

```
alter connection to data_server.database
   set dsi_non_blocking_commit to 'value'
```

Note: You cannot use this parameter with **alter connection** to configure an active database connection in a standby environment.

Use **configure replication server** to configure **dsi_non_blocking_commit** as a server default:

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

where *value* is the number of minutes, to a maximum of 60, to extend the save period. The default is zero, which disables non-blocking commit.

Use **admin config** to check the current *value* of **dsi_non_blocking_commit**.

For details about the commands discussed, see the *Replication Server Reference Manual*.

Version Requirements

You can use **dsi_non_blocking_commit** only with Adaptive Server 15.0 and later, and Oracle 10g v2 and later. Replication Server disables the **dsi_non_blocking_commit** configuration parameter for the connection for unsupported versions of Adaptive Server, Oracle, or other databases.

rs_non_blocking_commit System Function

rs_non_blocking_commit executes every time DSI connects to the replicate data server, if the **dsi_non_blocking_commit** value is from 1 to 60. If the value of **dsi_non_blocking_commit** is zero, **rs_non_blocking_commit** does not execute.

rs_non_blocking_commit has function-string class scope.

rs_non_blocking_commit function maps to the "**set delayed_commit on**" function string in Adaptive Server 15.0 and later, and to the corresponding "**alter session set commit_write = nowait;**" function string in Oracle 10g v2 and later. For all other non-Sybase databases, **rs_non_blocking_commit** maps to null.

rs_non_blocking_commit_flush System Function

rs_non_blocking_commit_flush executes at intervals equal to any number of minutes from 1 to 60 that you specify with **dsi_non_blocking_commit**. **rs_non_blocking_commit_flush** does not execute if the value of **dsi_non_blocking_commit** is zero.

rs_blocking_commit_flush has function-string class scope.

rs_non_blocking_commit_flush maps to the corresponding function string in Adaptive Server 15.0 and later, and Oracle 10g v2 and later. For all other non-Sybase databases, **rs_non_blocking_commit_flush** maps to null.

Example 1

Creates an instance of an **rs_non_blocking_commit_flush** function string for Adaptive Server:

```
create function string rs_non_blocking_commit_flush
         for sqlserver_derived_class
         output language
        'set delayed_commit off; begin tran; update rs_lastcommit set
         origin_time = getdate() where origin = 0; commit tran;
         set delayed_commit on'
```

Example 2

Creates an instance of an **rs_non_blocking_commit_flush** function string for Oracle:

```
create function string rs_non_blocking_commit_flush
         for oracle_derived_class
         output language
         'alter session set commit_write = immediate; begin tran;
         update rs_lastcommit set origin_time = getdate() where
         origin = 0; commit tran; alter session set commit_write =
nowait'
```

Non-Adaptive Server Databases Supported

Replication Server 15.2 with non-blocking commit enabled supports replication into Oracle 10g v2 and later because Oracle 10g v2 supports functionality similar to delayed commit.

Replication Server 15.2 heterogeneous datatype support (HDS) scripts have new function strings that support the non-blocking commit feature. Sybase Enterprise Connect Data Access for Oracle supports these function strings. See the *Replication Server Heterogeneous Replication Guide*.

Quoted Identifiers

In version 15.2, Replication Server enhances its support of quoted identifiers.

Object names that contain special characters such as spaces and nonalphanumeric characters, start with a character other than an alphabetic character, or that correspond to a reserved word, must be enclosed in double quote characters to be parsed correctly. These object names are referred to as quoted identifiers. Although Replication Server version 15.1 and earlier can accept quoted identifiers, forwarding quoted identifiers to data servers is not supported in these versions.

Note: To use quoted identifiers, you must use ECDA 15.0 ESD #3 or later.

As of Replication Server 15.2, quoted identifier support allows you to:

- Mark identifiers in a replication definition as quoted.
- Create a connection where you can forward quoted identifiers to data servers.

Embedded double quote characters in identifiers is not currently supported.

Data servers such as Adaptive Server, SQL Anywhere®, Microsoft SQL Server, Universal Database (UDB), and Oracle handle quoted identifiers differently in terms of supported length, special characters, and reserved words. In a heterogeneous environment, ensure that the quoted identifiers being replicated are valid on both the primary and replicate data servers.

Version Requirements

For replication of a quoted identifier to succeed, the primary Replication Server and the Replication Server that connects to the replicate data server version must be 15.2. However, intermediate Replication Servers in a route can be earlier versions.

Configuration Parameter to Enable Quoted Identifier Support

The **dsi_quoted_identifier** configuration parameter enables or disables quoted identifier support in the Data Server Interface (DSI).

Use the **create connection** or the **alter connection** command to set **dsi_quoted_identifier** on or off for a data server connection. The default value of **dsi_quoted_identifier** is off.

To check the value of **dsi_quoted_identifier**, use the **admin config** command.

Commands to Mark Identifiers as Quoted

create replication definition and **alter replication definition** commands allow you to mark quoted identifiers using the new parameter **quoted**.

When an identifier is marked and the **dsi_quoted_identifier** is set to on, the replicate servers that subscribe to the replication definition receives the marked identifier as a quoted identifier. If the **dsi_quoted_identifier** is off, the markings are ignored and the replicate server does not receive quoted identifiers.

Note: When replicating to a warm standby database and to replication definition subscribers, and the primary table name is marked as quoted but the replicate table name is not, or vice-versa, Replication Server sends both the primary table name and the replicate table name as quoted.

For details about the commands discussed in this section, see the *Replication Server Reference Manual*.

create replication definition Syntax Change

The **create replication definition** syntax has been modified to support quoted identifiers.

```
create replication definition replication_definition
with primary at data_server.database
[with all tables named [table_owner.] 'table_name' [quoted] |
[with primary table named [table_owner.]'table_name']
 with replicate table named [table_owner.]'table_name'] [quoted]]
(column_name [as replicate_column_name] [datatype [null | not null]
        [map to published_datatype]] [quoted]
[, column_name [as replicate_column_name]
        [datatype [null | not null]
        [map to published_datatype]] [quoted]...)
primary key (column_name [, column_name]...)
[searchable columns (column_name [, column_name]...)]
[send standby [{all | replication definition} columns]]
[replicate {minimal | all} columns]
[replicate_if_changed (column_name [, column_name]...)]
[always_replicate (column_name [, column_name]...)]
[with dynamic sql | without dynamic sql]
```

Example

Create a table *foo* with column *foo_col1* as a quoted identifier:

```
create replication definition repdef
   with primary at primaryDS.primaryDB
   with all tables named “foo”
   (“foo_col1” int quoted, “foo_col2” int)
   primary key (“foo_col1”)
```

alter replication definition Syntax Change

The **alter replication definition** syntax has been modified to support quoted identifiers.

```
alter replication definition replication_definition
{with replicate table named table_owner.]'table_name' |
add column_name [as replicate_column_name]
       [datatype [null | not null]]
       [map to published_datatype] [quoted],... |
alter columns with column_name
       [as replicate_column_name] [quoted | not quoted],...|
alter columns with column_name
        datatype [null | not null]
        [map to published_datatype],...|
alter columns column_name {quoted | not quoted}
add primary key column_name [, column_name]... |
drop primary key column_name [, column_name]... |
add searchable columns column_name [, column_name]... |
drop searchable columns column_name [, column_name]... |
send standby [off | {all | replication definition} columns] |
replicate {minimal | all} columns |
replicate_if_changed column_name [, column_name]... |
always_replicate column_name [, column_name]...} |
{with | without} dynamic sql
alter replicate table name {quoted | not quoted}
```

Example 1

Mark as quoted the table named *foo*:

```
alter replication definition repdef
   alter replicate table name “foo” quoted
```

Example 2

Unmark the column *foo_col1*:

```
alter replication definition repdef
   with replicate table named “foo”
   alter columns “foo_col1” not quoted
```

rs_set_quoted_identifier Function String

rs_set_quoted_identifier sets the connection appropriately for each data server type that receives quoted identifiers.

Data servers receive quoted identifiers differently. Adaptive Server, SQL Anywhere, and Microsoft SQL Server do not expect quoted identifiers, and require a special command to configure the connection for quoted identifiers. Oracle and UDB do not require the connection to be configured to accept quoted identifiers.

Use **rs_set_quoted_identifier** function string to set the DSI connection appropriately for each data server type. When **dsi_quoted_identifier** is on, Replication Server sends **rs_set_quoted_identifier** to the replicate data server to signal the data server to expect quoted

identifiers. If the replicate data server is Adaptive Server, SQL Anywhere, or Microsoft SQL Server, **rs_set_quoted_identifier** is set to **set quoted_identifiers on** command. Otherwise, **rs_set_quoted_identifier** is set to "".

rs_set_quoted_identifier has function-string-class scope.

Changes to rs_helprep

rs_helprep has been modified to display quoted identifiers as quoted. The examples illustrate how **create replication definition** and **alter replication definition** define quoted identifiers, and how **rs_helprep** displays these identifiers.

Example 1

Given this table and replication definition:

```
create table t1 (c1 int, c2 int)
```

```
create replication definition r1
   with primary at ost_wasatch_08.pdb1
   with all tables named t1
   (c1 int, "c2" int quoted)
    primary key (c1)
```

`rs_helprep r1` displays *c2* as a quoted identifier:

```
Replication Definition Name  PRS                  Type Creation Date
---------------------------- -------------------- ----
-------------------
r1                       ost_wasatch_09      Tbl  Nov 11, 2008 2:28PM

PDS.DB                Primary Owner        Primary Table
-------------------- -------------------- --------------------
ost_wasatch_08.pdb1                        t1

Replicate Owner      Replicate Table
-------------------- --------------------
                     t1

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- ----------- --------------
No             No              1000     On          None

Col. Name   Rep. Col. Name   Datatype   Len.   Pri. Col.   Searchable
---------   --------------   --------   ----   ---------   ----------
c1          c1               int           4           1            0
"c2"        "c2"             int           4           0            0

Function Name   FString Class                FString Source   FString
Name
-------------   ---------------------------   --------------   -----
-------
rs_delete       rs_sqlserver_function_class   Class
Default    rs_delete
rs_insert       rs_sqlserver_function_class   Class
Default    rs_insert
```

```
rs_select       rs_sqlserver_function_class   Class
Default    rs_select
rs_select_      rs_sqlserver_function_class   Class
Default    rs_select_
with_lock                                                    with_lock
rs_truncate     rs_sqlserver_function_class   Class
Default    rs_truncate
rs_update       rs_sqlserver_function_class   Class
Default    rs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription Name   Replicate DS.DB   Owner   Creation Date
-----------------   ---------------   -----   -------------
(return status = 0)
```

Example 2

Given the table and replication definition defined in example 1, when you define *t1* as a quoted identifier:

```
alter replication definition r1
alter replicate table name "t1" quoted
```

`rs_helprep r1` displays *c2* and *t1* as quoted identifiers:

```
Replication Definition Name  PRS                  Type Creation Date
---------------------------- -------------------- ----
-------------------
r1                       ost_wasatch_09      Tbl  Nov 11, 2008 2:28PM

PDS.DB                Primary Owner        Primary Table
--------------------- -------------------- --------------------
ost_wasatch_08.pdb1                        "t1"

Replicate Owner       Replicate Table
--------------------- --------------------
                      "t1"

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- ----------- --------------
No             No              1000     On          None

Col. Name   Rep. Col. Name   Datatype   Len.   Pri. Col.   Searchable
---------   --------------   --------   ----   ---------   ----------
c1          c1               int           4           1            0
"c2"        "c2"             int           4           0            0

Function Name   FString Class                FString Source   FString
Name
-------------   ---------------------------   --------------   -----
-------
rs_delete       rs_sqlserver_function_class   Class
Default    rs_delete
rs_insert       rs_sqlserver_function_class   Class
Default    rs_insert
rs_select       rs_sqlserver_function_class   Class
Default    rs_select
```

```
rs_select_      rs_sqlserver_function_class   Class
Default    rs_select_
with_lock                                                    with_lock
rs_truncate     rs_sqlserver_function_class   Class
Default    rs_truncate
rs_update       rs_sqlserver_function_class   Class
Default    rs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription Name   Replicate DS.DB   Owner   Creation Date
-----------------   ---------------   -----   -------------
(return status = 0)
```

Example 3

Given the replication definition defined in example 2, when you define *c2* as not quoted:

```
alter replication definition r1
alter columns c2 not quoted
```

`rs_helprep r1` displays *t1* as the only quoted identifier:

```
Replication Definition Name  PRS                  Type Creation Date
---------------------------- -------------------- ----
-------------------
r1                       ost_wasatch_09      Tbl  Nov 11, 2008 2:28PM

PDS.DB                Primary Owner        Primary Table
--------------------- -------------------- --------------------
ost_wasatch_08.pdb1                        "t1"

Replicate Owner       Replicate Table
--------------------- --------------------
                      "t1"

Send Min Cols. Used by Standby Min Vers Dynamic SQL SQL Stmt. Rep.
-------------- --------------- -------- ----------- --------------
No             No              1000     On          None

Col. Name   Rep. Col. Name   Datatype   Len.   Pri. Col.   Searchable
---------   --------------   --------   ----   ---------   ----------
c1          c1               int           4           1            0
c2          c2               int           4           0            0

Function Name   FString Class                FString Source   FString
Name
-------------   ---------------------------   --------------   -----
-------
rs_delete       rs_sqlserver_function_class   Class
Default    rs_delete
rs_insert       rs_sqlserver_function_class   Class
Default    rs_insert
rs_select       rs_sqlserver_function_class   Class
Default    rs_select
rs_select_      rs_sqlserver_function_class   Class
Default    rs_select_
with_lock                                                    with_lock
```

```
rs_truncate     rs_sqlserver_function_class   Class
Default    rs_truncate
rs_update       rs_sqlserver_function_class   Class
Default    rs_update

Subscriptions known at this Site 'ost_wasatch_09'.
Subscription Name   Replicate DS.DB   Owner   Creation Date
-----------------   ---------------   -----   -------------
(return status = 0)
```

Replication Server Gateway

Replication Server 15.2 introduces Replication Server gateway, which minimizes explicit login to the different servers.

In managing a replication system, the replication system administrator (RSA) must log in to multiple replication servers, ID Servers, and the corresponding Replication Server System Database (RSSD). The RSA must also frequently switch logins between Replication Server and the RSSD.

The Replication Server gateway uses your RSSD primary user name and password to log in to RSSD, your ID Server user name and password to log in to ID Server, your remote server identification (RSI) to log in to a remote Replication Server, and your maintenance user ID to log in to the remote Adaptive Server. You need not supply this information more than once, when you access Replication Server itself.

Limitations

When using Replication Server gateway, the client and the server must use the same locale set because Replication Server cannot perform character set conversion.

Cascading Connection

The Replication Server gateway supports cascading connections, which allow your Replication Server to communicate with servers it is not directly connected to.

Cascading connections also allow you to manage a replication domain using a single client connection. For example, you can connect to an ID Server, and then to the ID Server's RSSD. In this case, both the primary, controlling Replication Server and the ID Server are gateways; commands pass through to the ID Server's RSSD, and result sets pass back through to you.

Command to Enable Replication Server Gateway

The **connect** command has been added to turn Replication Server into a gateway to its RSSD, ID Server, or to a remote Replication Server.

Syntax

```
connect [to] [rssd | idserver | srv_name | ds_name.db_name]
```

Parameters

- **rssd** – turns Replication Server into a gateway to its RSSD. Allows the gateway to use *RSSD_primary_user* and *RSSD_primary_pw* entries in its configuration file.
- **idserver** – turns Replication Server into a gateway to its ID Server, provided that the Replication Server itself is not the ID Server. Allows the gateway to use *ID_user* and *ID_pw* entries in the configuration file.
- ***srv_name*** – name of the remote Replication Server you want the gateway to connect to. The Replication Server gateway uses RSI to log in to the remote server, and requires a direct route to the remote server.

  Note: Replication Server cannot directly connect to itself. However, you can work around this by using a cascading connection.
- ***ds_name.db_name*** – name of the remote data server and database that you want the gateway to connect to. The Replication Server gateway uses the maintenance user to log in to the remote data server. This allows you to perform tasks that maintenance users of the designated database are permitted to do. However, you cannot access the other databases defined in the data server you connected to.

  Replication Server gateway can directly connect to Adaptive Server, and to Sybase® IQ data servers that do not require Enterprise Connect Data Access (ECDA). For other data servers, Replication Server gateway has to use the ECDA to connect the Replication Server and the remote data server.

Usage

For details about the **connect** command, see the *Replication Server Reference Manual*.

Permissions

Issuing the **connect** command requires an **sa** role for the first login to Replication Server.

Commands to Track Connections

Cascaded connections created in the gateway are kept in a connection stack, with the Replication Server that issued the first **connect** command placed at the bottom of the stack. Use **show connection** and **show server** commands to manage your cascaded connections.

- **show connection** – lists the contents of the connection stack.
- **show server** – displays the current working server.

Usage

When your connection stack includes Replication Server versions 15.2, and 15.1 or earlier, and you issue a **disconnect** command, the **show connection** and **show server** commands may not display the expected output. This is because the **disconnect** command behaves differently in Replication Server 15.1 and earlier. In In these versions, a **disconnect** command terminates

the gateway mode, and returns the working server status to the Replication Server that issued the first **connect** command.

Command to Drop Connection

Use **disconnect** command to terminate a connection to a server.

Syntax

```
{disconnect | disc} [all]
```

```
select @variable = {expression | select_statement}
    [, @variable = {expression | select_statement} ...]
    [from table_list]
    [where search_conditions]
    [group by group_by_list]
    [having search_conditions]
    [order by order_by_list]
    [compute function_list [by by_list]]
```

Usage

- **disconnect** exits the connection stack one at a time. To exit from all the connections, use **disconnect all**.
- The **disconnect** command behaves differently in Replication Server 15.1 and earlier. In In these versions, a **disconnect** command terminates the gateway mode, and returns the working server status to the Replication Server that issued the first **connect** command. When your connection stack includes Replication Server versions 15.2, and 15.1 or earlier, and you issue a **disconnect** command, the **show connection** and **show server** commands may not display the expected output.
- For details about **disconnect**, see the *Replication Server Reference Manual*.

Row Count Validation for Non-SQL Statement Replication

To address errors in Replication Server, Replication Server 15.2 includes support for Replication Server error classes and error actions for row count verification errors not related to SQL statement replication.

Note: Replication Server ignores row count validation for those commands that are in a customized function string. See “SQL statement replication does not support autocorrection,” in Chapter 4, “Performance Tuning” in the *Replication Server Administration Guide Volume 2*.

Replication Server 15.2 introduces the Replication Server error class. Therefore, with version 15.2, a connection associates itself with two error class types—a data server error class and a Replication Server error class. You must associate a Replication Server error class with a connection before Replication Server can query the Replication Server error class for

overrides to the default Replication Server error actions. You can associate a connection with only one Replication Server error class. However, you can associate one Replication Server error class with multiple connections. Use the **set replication server error class** parameter for the **create connection** and **alter connection** commands to associate a Replication Server error class with a connection.

When Replication Server responds to errors, it looks first for the Replication Server error class assigned to the connection. If Replication Server does not find the Replication Server error class, Replication Server uses the default **rs_repserver_error_class** error class assigned to the server.

See also

- *Row Count Validation for SQL Statement Replication* on page 150

Command to Create Replication Server Error Classes

With Replication Server 15.2, you can use **create error class** to create Replication Server error classes that you can use to assign error actions for errors that occur in Replication Server.

Syntax

```
create [replication server] error class error_class
[set template to template_error_class]
```

Parameters

- **replication server –** indicates that the new error class is a Replication Server error class and not a data server error class.
- ***error_class* –** the name for the new error class. The name must be unique in the replication system and must conform to the rules for identifiers.

  Note: A Replication Server error class and a data server error class cannot share the same name.

- **set template to *template_error_class* –** use this clause to create an error class based on another error class. **create error class** copies the error actions from the template error class to the new error class.

Examples

- **Example 1 –** creates **my_rs_err_class** based on the default **rs_repserver_error_class**:

  ```
  create replication server error class my_rs_err_class
  set template to rs_repserver_error_class
  ```

Usage

You can drop a Replication Server error class using **drop error class**, and you can change the primary Replication Server of a Replication Server error class using **move primary**.

Command to Assign Error Actions

Use the **assign action** command at the primary site for the Replication Server error class to specify an error action.

Syntax

```
assign action
   {ignore | warn | retry_log | log | retry_stop | stop_replication}
   for error_class
   to server_error1 [, server_error2]...
```

Parameters

- ***error_class*** **–** the error class name for which the action is being assigned. With Replication Server 15.2, you can specify Replication Server error classes such as the default **rs_repserver_error_class** error class.
- ***server_error*** **–** the error number.

  You can specify these error numbers for Replication Server for error actions not related to SQL statement replication:

Examples

- **Example 1 –** assigns the **ignore** error action if Replication Server encounters error number 5185:

  ```
  assign action ignore for rs_repserver_error_class to 5185
  ```
- **Example 2 –** assigns the **warn** error action if Replication Server encounters error number 5186:

  ```
  assign action warn for rs_repserver_error_class to 5186
  ```

  If there is a row count error, this is an example of the error message that displays:

  ```
  DSI_SQLDML_ROW_COUNT_INVALID 5186
  Row count mismatch for SQLDML command executed on
  'mydataserver.mydatabase'.
  The command impacted 1000 rows but it should impact 1500 rows.
  ```

Usage

See “**assign action**” and Table 3-17 , in Chapter 3, “Replication Server Commands” in the *Replication Server Reference Manual*.

Non-SQL Statement Replication Error Numbers

Error numbers for Replication Server for error actions not related to SQL statement replication.

Table 21. Error Actions for Replication Server Error Classes

server_error	Error Message	Default Error Action	Description
5185	`Row count mismatch for the command executed on 'dataserver.database'. The command impacted x rows but it should impact y rows.`	**warn**	This message appears if the affected number of rows is different from the expected number of rows, after a command that is not part of SQL Statement Replication, or a stored procedure, or a row change with autocorrection enabled is sent to the data server.
5187	`Row count mismatch for the autocorrection delete command executed on 'dataserver.database'. The command deleted x rows but it should delete y rows.`	**warn**	This message appears if the affected number of rows is different from the expected number of rows, after a delete command is sent to the data server, and if autocorrection is enabled.

Stored Procedures to Display Replication Server Error Classes

Use the **rs_helpdb**, **rs_helpclass**, and **rs_helperror** stored procedures to display information about Replication Server error classes.

See Chapter 6, “Adaptive Server Stored Procedures” in the *Replication Server Reference Manual*.

Replication Server System Database Modifications

To support Replication Server error handling, two system tables in the Replication Server System Database (RSSD) have been modified.

System Table	Description
rs_classes	*classtype* column includes a new “R” value for Replication Server error classes.
rs_databases	*rs_errorclassid* is a new column for the Replication Server error class associated with the database.

SQL Statement Replication

Replication Server 15.2 supports SQL statement replication which complements log-based replication and addresses performance degradation caused by batch jobs.

In SQL statement replication, Replication Server receives the SQL statement that modified the primary data, rather than the individual row changes from the transaction log. Replication Server applies the SQL statement to the replicated site. RepAgent sends both the SQL data manipulation language (DML) and individual row changes. Depending on your configuration, Replication Server chooses either individual row change log replication or SQL statement replication.

SQL statement replication includes row count verification to ensure that the number of rows changed in the primary and replicate databases match after replication. If the number of rows do not match, you can specify how Replication Server handles this error.

See the *Adaptive Server 15.0.3 New Features Guide* for more information on SQL statement replication.

Product and Mixed-version Requirements

SQL statement replication requires Adaptive Server version 15.0.3 and later, primary and replicate Replication Server version 15.2 and later, and route version 15.2 and later.

Enabling SQL Statement Replication

Configure Replication Server and the primary database to replicate SQL statements.

1. Configure the primary database to log SQLDML.
2. Configure Replication Server to replicate SQLDML:
   a) Create replication definitions with SQLDML for table and multisite availability (MSA) replication.
   b) In Replication Server, set **WS_SQLDML_REPLICATION** parameter on for warm standby replication.

Modifications to System Configuration

Several Adaptive Server stored procedures support SQL statement replication.

Database-Level SQL Statement Replication

sp_setrepdbmode has been added to support SQL statement replication. **sp_setrepdbmode** allows you to enable SQL statement replication for a specific DML operation.

The DML operations that apply to SQL statement replication include:

- **U** – update

- **D** – delete
- **I** – insert select
- **S** – select into

When the database replication mode is set to any combination of **UDIS** the RepAgent sends both individual log records and the information needed by Replication Server to build the SQL statement.

For example, to replicate **delete** statements as SQL statement replication and also enable replication of **select into**, enter:

```
sp_setrepdbmode pdb, 'DS', 'on'
```

You can set SQL statement replication at the database level only when the database has been marked for replication by setting **sp_reptostandby** to **ALL** or **L1**.

See "**sp_setrepdbmode**" in the "System Changes" chapter in the *Adaptive Server Enterprise 15.0.3 New Features Guide*.

Table-Level SQL Statement Replication

sp_setrepdefmode has been enhanced to support SQL statement replication.

sp_setrepdefmode includes options to:

- Enable or disable SQL statement replication for a specific DML operation
- Configure the threshold that must be reached to activate SQL statement replication

The DML operations that apply to SQL statement replication include:

- **U** – update
- **D** – delete
- **I** – insert select

When the table replication mode is set to any combination of **UDI**, the RepAgent sends additional information to enable SQL statement replication for the specified DML operation.

For example, to enable SQL statement replication for **update**, **delete**, and **insert select** operations on table *t*, enter:

```
sp_setrepdefmode t, 'UDI', 'on'
go
```

See "**sp_setrepdefmode**" in the "System Changes" chapter in the *Adaptive Server Enterprise 15.0.3 New Features Guide*.

Session-Level SQL Statement Replication

Use session option **set repmode** to set replication mode to SQL statement replication.

You can specify session-level settings either when you log in, or at the beginning of a batch job. Session-level settings override both database-level and object-level settings.

Use **set repmode on** to enable SQL statement replication for the DML operation specified, for the duration of the session. Use **set repmode off** to remove all SQL statement replication

settings at the session level. For example, to replicate only **select into** and **delete** as SQL statements for the duration of the session, enter:

```
set repmode on 'DS'
```

See "**set repmode**" in "System Changes" in the *Adaptive Server Enterprise 15.0.3 New Features Guide*.

SQL Statement Replication Configuration

You can change replication options at database and table levels.

Database Replication Definition

Include the **replicate SQLDML** clause with the **create replication definition** or **alter replication definition** command to replicate SQL statements in a multisite availability (MSA) environment.

Syntax

This code segment displays the syntax for **create** and **alter** database replication definitions:

```
[[not] replicate setname [in (table list)] ]
```

where:

setname = **DDL** | **tables** | **functions** | **transactions** | **system procedures** | **SQLDML** | **'options'**

Parameters

- **'options' –** a combination of:
  - **U** – update
  - **D** – delete
  - **I** – insert select
  - **S** – select into
- **SQLDML –** also defined as a combination of **U**, **D**, **I**, and, **S** statements.

Examples

- **Example 1 –** uses the **'options'** parameter to replicate SQLDML on tables *tb1* and *tb2*:

  ```
  replicate 'UDIS' in (tb1,tb2)
  ```
- **Example 2 –** uses the **SQLDML** parameter to produce the same result as the **'options'** parameter in the previous example:

  ```
  replicate SQLDML in (tb1,tb2)
  ```
- **Example 3 –** filters out the **select into** statement for all tables. The second clause, **not replicate 'U' in (T)**, filters out updates on table *T*:

  ```
  create database replication definition dbrepdef
         with primary at ds1.pdb1
  ```

```
       not replicate ‘S’
       not replicate ‘U’ in (T)
go
```

- **Example 4 –** enables **update** and **delete** statements on all tables using the replicate 'UD' clause:

```
create database replication definition dbrepdef_UD
       with primary at ds2.pdb1
       replicate 'UD'
go
```

- **Example 5 –** applies **update** and **delete** statements for tables *tb1* and *tb2*:

```
alter database replication definition dbrepdef
      with primary at ds1.pdb1
      replicate 'UD' in (tb1,tb2)
go
```

Usage

- You can use multiple replicate clauses in a **create database replication** definition. However, for an **alter database replication** definition, you can use only one clause.
- If you do not specify a filter in your replication definition, the default is the **not replicate** clause. Apply **alter database replication definition** to change the SQLDML filters. You can either specify one or multiple SQLDML filters in a **replicate** clause.
- You can use multiple clauses to specify a table multiple times in the same definition. However, you can use each of **U**, **D**, **I**, and **S** only once per definition. For example:

```
create database replication definition dbrepdef
       with primary at ds2.pdb1
       replicate tables in (tb1,tb2)
       replicate 'U' in (tb1)
       replicate 'I' in (tb1,tb2)
go
```

Table Replication Definition

Include the **replicate SQLDML** clause for a **create** table replication definition to support SQL statement replication.

Syntax

This code segment displays the syntax for a **create** table replication definition:

```
[replicate {SQLDML [‘off’] | ‘options’}]
```

Parameters

- **‘options’ –** a combination of these statements:
  - **U** – update
  - **D** – delete

- **I** – insert select

Examples

- **Example 1 –** a sample **create replication definition** for a table:

```
create replication definition repdef1
       with primary at ds3.pdb1
       with all tables named 'tb1'

           (id_col int,
           str_col char(40))

       primary key (id_col)
       replicate all columns
       replicate ‘UD’
go
```

Usage

- If your replication definition has the **[replicate {minimal | all} columns]** clause, then the **[replicate {minimal | all} columns]** clause must always precede the **[replicate {SQLDML [‘off’] | ‘options’}]** clause.
- A table replication definition with the **send standby** clause can specify a **replicate ‘I’** statement. You can replicate an **insert select** statement as a SQL replication statement only in warm standby or MSA environments. A table replication definition without a **send standby** clause cannot replicate the **insert select** statement.

SQL Statement Replication Restrictions

SQL statement replication cannot perform autocorrection, and there are instances when SQL statement replication is not supported.

SQL statement replication is not supported when:

- A replicate database has a different table schema than the primary database.
- Replication Server must perform data or schema transformation.
- Subscriptions or articles include **where** clauses.
- Updates include one or more *text* or *image* columns.
- Function strings *rs_delete*, *rs_insert*, and *rs_update* are customized.

Autocorrection Support

SQL statement replication cannot perform autocorrection. If Data Server Interface (DSI) encounters a DML command for SQL statement replication and autocorrection is on, by default, DSI is suspended and stops replication. Use the **assign action** command with error number 5193 to specify how Replication Server handles this error.

Replication Server does not replicate SQLDML until the table level subscription is validated.

Row Count Validation for SQL Statement Replication

In Replication Server 15.2, you can specify how Replication Server responds to SQLDML row count errors that may occur during SQL statement replication.

SQLDML row count errors occur when the number of rows changed in the primary and replicate databases do not match after SQL statement replication. The default error action is to stop replication. You can use the **assign action** command at the primary site for the Replication Server error class to specify other error actions for SQLDML row count errors.

See also

- *Row Count Validation for Non-SQL Statement Replication* on page 141

Command to Assign Error Actions

Use the **assign action** command at the primary site for the Replication Server error class to specify an error action.

Syntax

```
assign action
   {ignore | warn | retry_log | log | retry_stop | stop_replication}
   for error_class
   to server_error1 [, server_error2]...
```

Parameters

- ***error_class*** **–** the error class name for which the action is being assigned. With Replication Server 15.2, you can specify Replication Server error classes such as the default **rs_repserver_error_class** error class.
- ***server_error*** **–** the error number.

  You can specify these error numbers for Replication Server for error actions not related to SQL statement replication:

Examples

- **Example 1 –** assigns the **ignore** error action if Replication Server encounters error number 5185:

  ```
  assign action ignore for rs_repserver_error_class to 5185
  ```
- **Example 2 –** assigns the **warn** error action if Replication Server encounters error number 5186:

  ```
  assign action warn for rs_repserver_error_class to 5186
  ```

  If there is a row count error, this is an example of the error message that displays:

  ```
  DSI_SQLDML_ROW_COUNT_INVALID 5186
  Row count mismatch for SQLDML command executed on
  ```

```
'mydataserver.mydatabase'.
The command impacted 1000 rows but it should impact 1500 rows.
```

Usage

See "**assign action**" and Table 3-17 , in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual*.

SQL Statement Replication Error Numbers

Error numbers for Replication Server for error actions related to SQL statement replication.

Table 22. Error Actions for SQL Statement Replication

server_error	Error Message	Default Error Action	Description
5186	`Row count mismatch for the command executed on 'dataserver.database'. The command impacted x rows but it should impact y rows.`	**stop_replication**	Row count verification error for SQL statement replication if the affected number of rows is different from what is expected.
5193	`You cannot enable autocorrection if SQL Statement Replication is enabled. Either enable SQL Statement Replication only or disable SQL Statement Replication before you enable autocorrection.`	**stop_replication**	Cannot enable autocorrection if SQL statement replication is enabled. Either enable SQL statement replication only or disable SQL statement replication before you enable autocorrection

Warm Standby Database Configuration for SQL Replication

By default, warm standby applications do not replicate the DML commands that support SQL statement replication. You need to perform extra configuration to use SQL replication.

To use SQL replication, you can:

- Create table replication definitions using **replicate SQLDML** and **send standby** clauses.
- Set the **WS_SQLDML_REPLICATION** parameter to on. The default value is **UDIS**. However, **WS_SQLDML_REPLICATION** has a lower precedence than the table replication definition for SQL replication. If your table replication definition contains a **send standby** clause for a table, the clause determines whether or not to replicate the DML statements, regardless of the **WS_SQLDML_REPLICATION** parameter setting.

Configuring Warm Standby Database for SQL Replication

By default, warm standby applications do not replicate the DML commands that support SQL statement replication. You need to perform extra configuration to use SQL replication. Perform one of these to use SQL replication:

- Create table replication definitions using **replicate SQLDML** and **send standby** clauses.
- Set the **WS_SQLDML_REPLICATION** parameter to on. The default value is **UDIS**. However, **WS_SQLDML_REPLICATION** has a lower precedence than the table replication definition for SQL replication. If your table replication definition contains a **send standby** clause for a table, the clause determines whether or not to replicate the DML statements, regardless of the **WS_SQLDML_REPLICATION** parameter setting.

Replication Server System Database Modifications

The *rs_dbreps*, *rs_dbsubsets*, and *rs_objects* system tables in the Replication Server System Database (RSSD) have been modified to support SQL statement replication.

System Table	Description
rs_dbreps	*status* column includes 4 new sets of 2-bit sets, each of which corresponds to a DML filter. The first bit of a set indicates if it is an empty filter and the second bit indicates if it is a negative statement set.
rs_dbsubsets	*type* column includes four new types: **U**, **L**, **I**, and, **S** corresponding to the DML **UDIS** filters. In this case, **L** is used for delete instead of **D**.
rs_objects	*attributes* column includes five new bits; one for each **U**, **D**, **I**, or **S** operation, and one to indicate if a table replication definition has fewer columns than the number of incoming data rows. A system function replication definition, **rs_sqldml**, also supports SQL statement replication.

Non-Adaptive Server Error Class Support

Replication Server 15.2 provides support for error classes and error action mapping for non-Adaptive Server Enterprise (non-ASE) replication databases.

You can use the default non-ASE error classes included in Replication Server 15.2. You can also create and alter your own error classes for non-ASE replicate databases.

To assign non-ASE error classes to specific connections on non-ASE replication databases, you can use the **create connection** and **alter connection** commands.

For more information about error classes and error handling, see the *Replication Server Administration Guide Volume 2*.

Native Error Codes

When Replication Server establishes a connection to a non-ASE replicate server, Replication Server verifies if the option to return native error codes from the non-ASE replicate server is enabled for the connection. If the option is not enabled, Replication Server logs a warning message that the connection works but error action mapping may not be correct.

See "**ReturnNativeError**," in the Replication Server Options documentation to set the option in the Enterprise Connect™ Data Access (ECDA) Option for ODBC for your replicate server.

See also

- *Non-Adaptive Server Replication Support Enhancements* on page 154

Default Non-ASE Error Classes

Replication Server 15.2 introduces several default non-Adaptive Server Enterprise (non-ASE) error classes. You cannot modify these default error classes.

Table 23. Default Non-ASE Error Classes

Database	Class Name
IBM DB2	*rs_db2_error_class*
IBM UDB	*rs_udb_error_class*
Microsoft SQL Server	*rs_msss_error_class*
Oracle	*rs_oracle_error_class*

Modified create error class Command

Replication Server 15.2 includes the **set template to** option for the **create error class** command.

Syntax

```
create error class error_class
[set template to template_error_class]
```

Examples

- **Example 1 –** creates the **my_error_class** error class for an Oracle database based on **rs_oracle_error_class** as a template:

  ```
  create error class my_error_class set template to
  rs_oracle_error_class
  ```

Usage

Use **create error class**, and **set template to**, and another error class as a template, to create your own error classes. **create error class** copies the error actions from the template error class to the new error class. See the *Replication Server Reference Manual*.

Modified alter error class Command

Replication Server 15.2 includes the **set template to** option for the **alter error class** command.

Syntax

```
alter error class error_class
set template to template_error_class
```

Examples

- **Example 1 –** alters **my_error_class** for an Oracle database based on **rs_sqlserver_error_class** as a template:

  ```
  alter error class my_error_class set template to
  rs_sqlserver_error_class
  ```

Usage

Use the **alter error class** command, and another error class as a template, to alter error classes. **alter error class** copies error actions from the template error class to the error class you want to alter and overwrites error actions that have the same error code. See the *Replication Server Reference Manual*.

Non-Adaptive Server Replication Support Enhancements

Replication Server 15.2 includes several enhancements to installation, configuration, and overall usability of setting up replication environments that include actively supported non-Adaptive Server Enterprise (non-ASE) data servers.

These enhancements automate the installation and configuration process by providing a pre-configured Replication Server environment that enables replication involving actively supported non-ASE data servers to be up and running quickly.

Actively supported data servers are data servers for which Sybase provides all the required software, documentation, and support for the data servers to serve as both a primary or a replicate data server. See the *Replication Agent Release Bulletin* for your platform for the list of actively supported non-ASE data servers.

For more information on support for non-ASE data servers, see the *Replication Server Heterogeneous Replication Guide* and the *Replication Server Administration Guide Volume 1*.

See also

- *Non-Adaptive Server Error Class Support* on page 152

Simplified Installation and Configuration

With Replication Server 15.2, you do not need to edit and execute scripts to install datatype definitions, function strings, and class-level translations for heterogeneous (non-ASE) datatype support.

The functions provided by the scripts are included as part of the Replication Server 15.2 installation, or are included in connection profiles that are installed with Replication Server 15.2. These enhancements simplify installation and configuration for non-ASE environments. Follow the simplified configuration instructions in Chapter 8, “Installing and Implementing Non-ASE Support Features” in the *Replication Server 15.2 Configuration Guide* for your platform.

Connection Profiles

With Replication Server 15.2, you can use connection profiles that contain connection configurations and replicate database object definitions relevant to each type of actively supported non-ASE data server. Connection profiles specify the function-string class, error class, and class-level translations to be installed.

You can use these connection profiles and simple syntax to create connections between actively supported data servers such as Adaptive Server Enterprise, IBM DB2, Microsoft SQL Server, and Oracle. Replication Server uses the connection profile to configure the connection and create replicate database objects for you.

You can also use connection profile options to specify other actions such as whether commands should be batched and the command separator to use.

Note: When you create a connection using a connection profile, the system table services (STS) caches are refreshed so that you do not need to restart Replication Server.

For more information on support for non-ASE data servers, see the *Replication Server Heterogeneous Replication Guide* and the *Replication Server Administration Guide Volume 1*. For the updated configuration process, see the *Replication Server Configuration Guide* for your platform.

using profile Clause

Use the **using profile** clause with the **create connection** command to create a connection between a non-ASE database and Adaptive Server using a connection profile.

Syntax

Here is a portion of the **create connection** syntax that shows the **using profile** and **display_only** clauses:

```
create connection to data_server.database
using profile connection_profile;version
```

```
set username [to] user
[other_create_connection_options]]
[display_only]
```

Parameters

- **connection_profile –** provide the connection profile you want to use to configure a connection, modify the Replication Server System Database (RSSD), and build replicate database objects.
- **version –** specify the particular version of a connection profile.
- **other_create_connection_options –** use *other_create_connection_options* to set connection options not specified in the profile, such as setting your password, or to override options specified in the profile, such as specifying a custom function string class to override the function string class provided in Replication Server. See "**create connection**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual* for all the parameters you can use with **create connection**.
- **display_only –** use with a connection profile to display but not execute commands and the servers where the commands would be executed. Use the client and Replication Server logs to see the results of using **display_only**.

Examples

- **Example 1 –** creates a connection to an Oracle replicate database:

  ```
  create connection to oracle.instance
  using profile rs_ase_to_oracle;standard
  set username to ora_maint
  set password to ora_maint_pwd
  ```

- **Example 2 –** creates a connection to a Microsoft SQL Server replicate database that is also a primary database. In this example, the command replaces any error class setting provided by the connection profile with another error class—**my_msss_error_class**:

  ```
  create connection to msss_server.msss_db
  using profile rs_ase_to_msss;standard
  set username to msss_maint
  set password to msss_maint_pwd
  set error class to my_msss_error_class
  with log transfer on
  ```

- **Example 3 –** creates a connection to a DB2 replicate database using a specific version of the profile—v9_1. In this example, the command overrides the command batch size provided by the connection profile with a new value—16384:

  ```
  create connection to db2.subsys
  using profile rs_ase_to_db2;v9_1
  set username to db2_maint
  set password to db2_maint_pwd
  set dsi_cmd_batch_size to '16384'
  ```

- **Example 4 –** use the **display_only** option to show the commands that will be executed if you use a particular profile. The commands and the command output display on your screen and are also written to the Replication Server log:

```
create connection to oracle.instance
using profile rs_ase_to_oracle;standard
set username to ora_maint
set password to ora_maint_pwd
display_only

go
```

Usage

For more information about **create connection**, see the *Replication Server Reference Manual*.

Command to List Available Connection Profiles

Use the **admin show_connection_profiles** command to list the profile name, version, and comments for each profile defined in Replication Server.

There is a connection profile for each primary and replicate database combination, such as Adaptive Server and Oracle, Oracle and Adaptive Server, and IBM DB2 and /Microsoft SQL Server, and so on.

See "**admin show_connection_profiles**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual* for more information about **admin show_connection_profiles** and the list of connection profiles.

Syntax

```
admin show_connection_profiles[, "match_string"]
```

Parameters

- **match_string –** use the *match_string* option to display only the connection profiles whose names contain the string you provide in the option.

Examples

- **Example 1 –** lists the names of all connection profiles currently defined in Replication Server:

  ```
  admin show_connection_profiles
  go
  ```

  Extract of output is:

  ```
  Profile Name              Version     Comments
  ------------              -------     --------
  rs_ase_to_db2             Standard    Standard ASE to DB2
  replication
                                        connection profile.
  rs_ase_to_udb             Standard    Standard ASE to DB2
  replication
                                        connection profile.
  ...
  ```

```
rs_db2_to_ase             Standard    Standard DB2 to ASE
replication
                                      connection profile.
rs_db2_to_msss            Standard    Standard DB2 to Microsoft
SQLServer
                                      connection profile.
...
```

- **Example 2 –** lists the names of all connection profiles currently defined in Replication Server that have the string “oracle” in the connection profile name:

  ```
  admin show_connection_profiles, “oracle”
  go
  ```

  Output is:

  ```
  Profile Name              Version     Comments
  ------------              -------     --------
  rs_ase_to_oracle          Standard    Standard ASE to Oracle
  replication
                                        connection profile.
  ```

System Tables for Connection Profiles

The *rs_profile* and *rs_profdetail* system tables support connection profiles.

System Table	Description
rs_profile	Stores currently defined profiles in Replication Server.
rs_profdetail	Stores the profile details.

New Features in Replication Server Version 15.1

Replication Server 15.1 introduces several new features and enhancements. They include enhancements to dynamic SQL, function replication, monitors and counters, datatype support, stable queue management, password encryption, timestamp support, and dump transaction. New features include support for Adaptive Server shared-disk cluster, Adaptive Server integer identity, partial update of LOB datatypes, and distributor status recording.

Dynamic SQL Enhancements

Dynamic SQL now supports heterogeneous replicate databases including Oracle, Universal Database (UDB), DB2, and Microsoft SQL.

Dynamic SQL in Replication Server enhances replication performance by allowing Replication Server Data Server Interface (DSI) to prepare dynamic SQL statements at the target user database and to run them repeatedly. **create/alter replication definition** commands allow you to control the application of dynamic SQL on each table through replication definition. See the *Replication Server Reference Manual* for information about **create/alter replication definition** commands.

You can change the dynamic SQL execution at the table level for a specific replicate database by using:

```
set dynamic_sql {on | off}
for replication definition with replicate at
data_server.database
```

To check for dynamic SQL usage, turn on **stats_sampling** and run **admin stats, dsi** command and look for DSIEDsqlPrepared, DSIEDsqlExecuted and other dynamic SQL related counters.

Use stored procedures **rs_helprep**, **rs_helpsub**, and **rs_helppubsub** to display dynamic SQL setting for each replication definition.

See "**rs_helprep**", "**rs_helpsub**", and "**rs_helppubsub**," in Chapter 6, "Adaptive Server Stored Procedures" in the *Replication Server Reference Manual* for information about using these stored procedures.

Limitations

Dynamic SQL commands support the data within Sybase range. If data falls outside Sybase ranges that cause dynamic SQL to fail, DSI logs an error message and resends dynamic SQL using the language command. DSI shuts down only if the language command also fails.

If this condition happens frequently, disable dynamic SQL from the table replication definition or use the **set dynamic_sql off** command.

Use any of these commands to turn **dynamic_sql off**:

- **alter connection... set dynamic_sql off** – turns dynamic SQL off for all commands replicating to this connection.
- **create/alter replication definition...without dynamic_sql** – turns dynamic SQL off for all commands using this replication definition.
- **set dynamic_sql off for replication definition with replicate at...** – turns dynamic SQL off for all commands using this replication definition at this replicate connection.

Function Replication Enhancements

In Replication Server 15.1, you can create a function replication definition that has a different name than the function name.

Replication Server enforces different primary and replicate function names for the request function replication definition. The **maint_user** runs the transaction at the replicate database if the function is replicated through applied function replication definition. The **origin_user** runs the transaction if the function is replicated through request function replication definition at the replicate database.

These enhancements let you:

- Replicate multiple functions with the same name from different databases.
- Have multiple replication definitions for one primary function and each specifies a different replicate function for a different replicate site.

To manage function replication definition, use:

- **create applied function replication definition**
- **create request function replication definition**
- **alter applied function replication definition**
- **alter request function replication definition**

Mixed-Version Support

This enhancement supports mixed-version environments. However, function replication definitions that have different primary function name and replication definition name are not replicated to Replication Servers earlier than 15.1.

Warning! If your system has an earlier version of a request function replication definition, drop the earlier-version definition before creating a 15.1 replication definition for the same primary function.

Warm Standby and Multisite Availability (MSA) Support

In a warm standby or MSA environment, there is only one parameter list for all the function replication definitions of the same primary function. If you alter one function replication

definition to add a parameter, the new parameter is added to all the function replication definitions created for this function.

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

Limitations

Enhanced function replication has these limitations:

- All the function replication definitions created for the same function must have the same parameter list with the same name and datatype.
- If you created a function replication definition with differing primary function name and replication definition name in version 15.1, any earlier version of the request function replication definition for the same primary function is disabled.
- You cannot have both applied function replication definitions and request function replications for a primary function. The function replication definition created by using **create function replication definition** command is considered an applied function at the function primary site.
- For each applied and request function replication definition, you must create a corresponding subscription to replicate a function.

Adaptive Server Shared-Disk Cluster Support

Replication Server and the RepAgent thread both support the Adaptive Server shared-disk cluster environment, which is where many Adaptive Servers share a single set of disks or databases.

In a Sybase shared-disk cluster, a database can be either a replication source or a replication destination. You can perform all tasks, such as configuring RepAgent or marking tables for replication, from any instance in the cluster.

See Chapter 5 "Setting Up and Managing RepAgent" in the *Replication Server Administration Guide Volume 1*.

Enhanced Monitors and Counters

The enhanced monitors and counters feature allows you to collect the information of the most active tables, procedures and related statistics, and store this information into the Replication Server System Database (RSSD) *rs_statdetail* table and related tables.

You can use this information to diagnose the replicate database performance issues such as missing indexes on the primary keys, and latency problems in the Replication Agent™ and Stable Queue Transaction interface (SQT)/Distributor (DIST) processing.

New Active Object Counters

New counters have been added to count the statement execution time on tables and procedures.

- AOBJInsertCommand
- AOBJUpdateCommand
- AOBJDeleteCommand
- AOBJWritetextCommand
- AOBJExecuteCommand

To flush the active object counter metrics to the RSSD, run any of these commands:

- **admin stats, “all”, save**
- **admin stats, dsi, save**
- **admin stats, sysmon, save**

See **rs_helpcounter** in the *Replication Server Reference Manual* for details on displaying information about counters.

New Procedure Interface

To bring out the most active tables and procedures, and related statistics information, Replication Server 15.1 introduces the stored procedures **rs_stat_populate** and **rs_stat_genreport**.

rs_stat_populate reads data from *rs_statdetail*, summarizes, augments, denormalizes, and saves result into **rs_statreport**, where **rs_stat_genreport** reads data and generates report.

Load this script into the RSSD after upgrading to Replication Server 15.1:

```
$SYBASE/$SYBASE_REP/scripts/
rs_install_statreport_v1510_[ase|asa].sql
```

After loading the script, run the stored procedures **rs_stat_populate** and **rs_stat_genreport**. Running these stored procedures produces this information:

- Replication Server performance overview – overview information about your Replication Servers, such as DIST processing, DSI processing, and so on.
- Replication Server performance analysis – performance analysis and tuning suggestions based on critical Replication Server counters. The detailed description is available in the script file.
- Active object identification result – lists the active table and procedure names, owner names, execution times, and so on.

For more information about **rs_stat_populate**, **rs_stat_genreport**, **rs_statreport**, and *rs_statdetail*, see the script file.

Improved Stable Queue Management

Replication Server 15.1 simplifies stable queue management.

Enhanced queue **dump** commands provide flexibility in identifying the stable queues, controlling the stable queue contents to dump, and supporting additional output file options. Replication Server 15.1 also includes new commands that allow you to delete and restore specific transactions from the Stable Queue Manager (SQM).

For more information about the stable queue management, see the *Replication Server Administration Guide Volume 1*. For details about the following commands, see the *Replication Server Reference Manual*.

Changes to sysadmin dump_queue

The **sysadmin dump_queue** syntax has been modified to provide flexibility to stable queue management.

Enhancements to **sysadmin dump_queue** include:

- An option to specify the server or database name instead of the queue number when identifying the stable queue to dump.
- An option to specify the number of commands to dump.
- Filtering options such as dumping only the begin and end commands of a transaction and dumping everything in the queue as comments except SQL statements.
- The option to direct the output to the Replication Server log or to a user-defined log file.
- An option to start the data dump from where the previous **sysadmin dump_queue** command stopped for that particular queue and session.

The modified **sysadmin dump_queue** syntax is:

```
sysadmin dump_queue {, q_number | server [,database]},
        {q_type,seg, blk, cnt
        [, num_cmds]
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client | “log” | file_name}] |
        “next” [, num_cmds]}
```

Changes to sysadmin sqt_dump_queue

The **sysadmin sqt_dump_queue** syntax has been modified to provide flexibility to stable queue management.

Enhancements to **sysadmin sqt_dump_queue** include:

- An option to specify the server or database name instead of the queue number when identifying the stable queue to dump.

- An option to dump all committed transactions and read transactions found in the SQT cache.
- An option to specify the number of commands to dump.
- Filtering options such as dumping only the begin and end commands of a transaction and dumping everything in the queue as comments except SQL statements.
- The option to direct the output to the Replication Server log or to a user-defined log file.

The modified **sysadmin sqt_dump_queue** syntax is:

```
sysadmin sqt_dump_queue {, q_number | server [,database]},
        q_type, reader
        [,{open | closed | read}]
        [, num_cmds]
        [, {L0 | L1 | L2 | L3}]
         [, {RSSD | client | “log” | file_name}]
```

Modified resume connection Command

The **resume connection skip transaction** option has been enhanced to support skipping a specified number of transactions in the connection queue before resuming the connection.

Skipped transactions are written to the database exception log, and to either the Replication Server log or the alternative log file specified by the **sysadmin dump_file** command. The maximum number of transactions that this command can skip is the number of transactions in the Data Server Interface (DSI) outbound queue.

The modified **resume connection** syntax is:

```
resume connection to data_server.database
        [skip [n] transaction |  execute transaction]
```

Modified sysadmin log_first_tran Command

A new option *n* has been added to the **sysadmin log_first_tran** command. Use the new option to specify the number of transactions to write to the database exceptions log, and to either the Replication Server log or the alternative log file specified by the **sysadmin dump_file** command.

The modified **sysadmin log_first_tran** syntax is:

```
sysadmin log_first_tran [,n], data_server, database
```

New sysadmin sqm_zap_tran Command

sysadmin sqm_zap_tran deletes a specific transaction from the stable queue and returns a message stating the number of deleted commands. The transaction is identified through the local queue ID (LQID).

The **sysadmin sqm_zap_tran** syntax is:

```
sysadmin sqm_zap_tran {, q_number, | server [,database]},
        q_type, lqid [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client  | “log” | file_name}]
```

Note: Replication Server must be in standalone mode to use this command.

New sysadmin sqm_unzap_tran Command

sysadmin sqm_unzap_tran restores a transaction in the stable queue and returns a message stating the number of restored transaction commands. The transaction is identified through the LQID.

The **sysadmin sqm_unzap_tran** syntax is:

```
sysadmin sqm_unzap_tran {, q_number, | server [,database]},
        q_type, lqid [,  {L0 | L1 | L2 | L3}]
        [, {RSSD | client  | “log” | file_name}]
```

Note: Replication Server must be in standalone mode to use this command.

New sysadmin dump_tran Command

Use **sysadmin dump_tran** to dump the statements of a specific stable queue transaction into a log file. The transaction is identified through the LQID.

The **sysadmin dump_tran** syntax is:

```
sysadmin dump_tran {{, q_number, | server [,database]},
        q_type, lqid
        [, num_cmds]
        [, {L0 | L1 | L2 | L3}]
        [, {RSSD | client  | “log” | file_name}] |
        “next” [, num_cmds]}
```

Changes to the locales Directory

The Replication Server release area and localization directory structure have been modified. You can now install or uninstall multiple Sybase products, such as Replication Server and Adaptive Server, on the same computer and same directory. You can also install multiple versions of Replication Server in the same directory.

The changes to the `locales` directory include:

- Replication Server-specific `locales` files have been moved from `$SYBASE/locales` to a new directory `$SYBASE/$SYBASE_REP/locales`.
- All `<charset>` subdirectories have been consolidated into one `<utf8>` subdirectory for each language in the new directory `$SYBASE/$SYBASE_REP/locales`.

  The UTF-8 character set that is used for all supported languages instead of using different character sets. You can convert UTF-8 into other characters and vice versa.

  Replication Server reads messages from a localization file, and converts the messages to the specified character set format during runtime.
- **rs_init** locale files are now located in `$SYBASE/$SYBASE_REP/locales/<language>/utf8/sybinit`.

Extended Password Encryption Support

Replication Server 15.1 uses Sybase Common Security Infrastructure (CSI) for server or client authentication, cryptography for encryption and decryption of passwords that are stored in the RSSD tables, and key-pair generation to support extended password encryption.

Extended password encryption uses asymmetric key encryption, which allows Open Client applications with connection property **CS_SEC_EXTENDED_ENCRYPTION** enabled to connect to Replication Server. It also allows Replication Server to enable **CS_SEC_EXTENDED_ENCRYPTION** when connecting to other servers.

Asymmetric key encryption uses a public key to encrypt the password and a private key to decrypt the password. The private key is not shared across the network, and is therefore secure.

Note: To use the extended password encryption feature, you must have a server that supports extended password encryption, such as Adaptive Server 15.0.2 ESD #2 or later. Additionally, this feature is not supported in HP-Itanium platform in 15.1 release.

rs_ticket Stored Procedure Version 2

rs_ticket is now at version 2 and provides support for non-Adaptive Server Enterprise (non-ASE) databases.

You can use **rs_ticket** stored procedure without additional configuration and administration. Tickets are automatically inserted into the **rs_ticket_history** table, which is located in the replicate database. Tickets are sharable among multiple applications, where they are issued without obstruction from tickets of other applications.

With **rs_ticket** version 2, more ticket information is provided for better usability, such as the Replication Server Interface (RSI) timestamp, which you can retrieve without writing complex queries. You can directly query the **rs_ticket_history** table for Replication Server performance. If the computer time or tickets are not synchronized across multiple time zones, you can change the timestamp columns to adjust the ticket date.

The earlier version of **rs_ticket** has been renamed **rs_ticket_v1**. To use the earlier version, alter **rs_ticket_report** function string with your old content or with the default, **exec rs_ticket_report ?rs_ticket_param!param?**.

Note: If you previously disabled **dsi_rs_ticket_report** and upgraded to Replication Server 15.1, the **dsi_rs_ticket_report** setting is reenabled after the upgrade process has finished.

These are the format changes that have been made to **rs_ticket**:

- Changed version number to 2, V=2; if a ticket has version number equal to 1, Replication Server does not write date to tickets.

- The ticket size has been increased from 255 to 1024 bytes.
- The *timestamp* format has been changed to include the date. The new *timestamp* format is mm/dd/yy hh:mm:ss:mmm.
- A Replication Server Interface (RSI) module timestamp that forces the RSI sender to parse RSI messages has been added. Tickets pass more than one RSI modules. However, the **rs_ticket_history** table keeps only the last RSI timestamp.
- Primary and target Replication Server names have been added to identify where a ticket comes from and where it goes to.
- Primary and replicate database names have been added.
- Two DSI counters have been added:
  - DSI_T=xxx – total transactions that the Data Server Interface (DSI) reads.
  - DSI_C=xxx – total commands that the DSI reads.

See the *Replication Server Reference Manual* for detailed information about using **rs_ticket** version 2.

New Replication Server Counters

New counters have been added for REPAGENT, RSIUSER, SQM, DSI, and DSIEXEC.

Counter	Description
RepAgentExecTime	The amount of time, in milliseconds, that the RepAgent user thread is scheduled by Open Client/Server™.
RSIUExecTime	The amount of time, in milliseconds that the RSI user thread is scheduled by Open Client/Server.
SQMWaitSegTime	The amount of time waiting for allocating segments.
DSINoDsqlNULL	Number of commands that cannot use dynamic SQL statements because of NULL value in **where** clauses.
DSINoDsqlDatatype	Number of commands that cannot use dynamic SQL statements because of *text*, *image*, *java* and ineligible UDDs.
DSINoDsqlRepdef	Number of commands excluded from dynamic SQL by replication definition.
DSINoDsqlColumn-Count	Number of commands excluded from dynamic SQL because the number of parameters exceeds 255.
DSINoDsqlMissingCols	Number of commands excluded from dynamic SQL because some columns are not available at DSI. This can be caused by minimal columns feature.
DSIEDsqlPrepared	Dynamic SQL statements prepared at target database by a Data Server Interface executor (DSI/E).
DSIEDsqlDealloc	Dynamic SQL statements deallocated at target database by a DSI/E.

Counter	Description
DSIEDsqlExecuted	Dynamic SQL statements executed at target database by a DSI/E.
DSIEDsqlDeallocSche-ma	Dynamic SQL statements deallocated at replicate database by a DSI/E because of schema change.
DSIEDsqlDeallocExec-Fail	Dynamic SQL statements deallocated at replicate database.

See "**rs_helpcounter**" in Chapter 6 "Adaptive Server Stored Procedures" in the *Replication Server Reference Manual* for commands to retrieve information about counters.

Extended Support for Large Object Datatypes

Replication Server 15.1 supports the replication of Microsoft SQL Server 2005 datatypes *varchar(max)*, *nvarchar(max)*, and *varbinary(max)*. These datatypes can each store up to 2,147,483,647 bytes of data.

Replication Server introduces large-object (LOB) datatypes as user-defined datatypes (UDDs) in the table-level replication environment. Replication Server also supports database-level replication for new LOB types. The new LOB types are directly mapped to *text*, *unitext*, and *image* datatypes.

The base type of UDDs is:

New LOB Datatype	Base Type
varchar(max)	*text*
nvarchar(max)	*unitext*
varbinary(max)	*image*

For more information about the new LOB datatypes, see the *Replication Server Reference Manual*.

Limitations

The limitations of the new LOB datatypes are:

- You cannot define as a primary key a LOB column in the table replication definition.
- You cannot define as searchable a LOB column in the table replication definition or function replication definition.
- You cannot replicate stored procedures that include one of the new LOB datatypes as a parameter.
- You cannot use text pointers to manipulate the data of the new LOB datatypes.

Mixed-version Support
In a mixed-version environment, the primary and replicate Replication Server must have a site version of 15.1 and an LTL version of 710.

Partial Update of Large Object Datatypes

A partial-update transaction allows you to directly insert a character string or overwrite an existing character string of a table column without issuing a **delete** and **replace** commands.

Replication Server 15.1 supports replication of partial-update transaction to supported large-object (LOB) datatypes.

To implement partial update, use the new **rs_updatetext** LTL command:

```
{distribute|_ds} command_tags {applied|_ap} 'table'.rs_updatetext
{partialupd|_pu} [{first|_fi}] [last] [{changed|_ch}] [with log]
[{withouttp|_wo}] [{offset|_os}=offset {deletelen|
_dln}=deletelength]
[{textlen|_tl}=length] text_image_column
```

Limitations
Partial update:

- Does not support multiple character set conversion.
- Support is restricted to Microsoft SQL Server 2005.

For more information about partial update, see the *Replication Server Design Guide*.

Extended timestamp Support

A new datatype, *timestamp* has been added to Replication Server 15.1. The *timestamp* datatype allows the replication of *timestamp* columns to replicate, standby, and MSA databases.

You can also define *timestamp* as a primary key in a replication definition, and as a searchable column in a replication definition and a function replication definition.

timestamp is defined as *varbinary*(8) with a status bit indicator that differentiates it from *varbinary*.

The **send_timestamp_to_standby** configuration parameter has been added to support *timestamp* replication. When **send_timestamp_to_standby** is enabled and there are no replication definitions, *timestamp* columns are sent to the replicate database.

Note: The replicate Adaptive Server must be version 15.0.2 or later to support *timestamp* in replication definition.

See the *Replication Server Reference Manual* for more information about the *timestamp* datatype. See the *Replication Server Administration Guide Volume 1* for information about replicating *timestamp* columns.

New opaque Datatype

The *opaque* datatype handles replication of datatypes that Replication Server does not support.

RepAgent provides formatting data that can be directly applied in the target database. The *opaque* datatype handles replication of datatypes that can store unspecified or inconsistent values, such as *anydata* datatype and the Microsoft SQL Server *sql_variant* datatype.

Limitations

Limitations of the *opaque* datatypes include:

- You cannot use *opaque* datatypes in searchable columns and **where** clauses of replication definitions, subscriptions, and articles.
- You cannot use a **map to** clause with *opaque* datatypes.
- You cannot use dynamic SQL when an *opaque* datatype column or parameter exists in your replication definition.
- You cannot use the *opaque* datatype if your function string has a remote procedure call (RPC).
- You cannot apply character set conversion or byte-order conversion to *opaque* datatypes.

In a mixed-version environment, the primary and replicate Replication Server must have a site version of 15.1 and an LTL version of 710.

For more information about *opaque* datatypes, see the *Replication Server Reference Manual*.

Dump Transaction Enhancements

The Log Transfer Language (LTL) **dump** subcommand and **rs_dumptran** function string have been enhanced to support replication of **dump** transaction commands with the **with standby_access** parameter.

New dump Subcommand Parameters

standby, **stdb**, and **status** parameters have been added to the **dump** subcommand syntax to support **with standby_access**.

```
{distribute|_ds} command_tags dump [ database |
  {transaction | tran | _tr}[standby | stdb | status]]
   database_name, 'dump_label' id
```

Table 24. New dump Subcommand Parameters

Parameter	Description
standby / **stdb**	An optional keyword that tells Replication Server that the command is a **dump transaction** command that uses **with standy_access**.
status	Dump status: • 0 – the **dump transaction** command does not contain **with standby_access**. • 1 – the **dump transaction** command contains **with standby_access**.

Example 1

RepAgent sends a **dump transaction** command to the Replication Server:

```
distribute @origin_time='Apr 15 1988 10:23:23.010PM',
@origin_qid=0x0000000000000000000000000000000000000000
00000000000000000000000377,@tran_id=0x0000000000000000
00000372 dump tran stdb devdb 'Apr 15 1988
10:23:23.011PM'0x0000000000000000000000000000000000000
00000000000000000000000000377
```

rs_dumptran Modification

A system variable *rs_dump_status* has been added to the **rs_dumptran** function string to support **with standby_access**.

Valid values for *rs_dump_status* are:

- 0 – the **dump transaction** command does not contain **with standby_access**.
- 1 – the **dump transaction** command contains the parameter **with standby_access**.

The changes to rs_dumptran are supported in Replication Server 15.1 or later. If you are using an earlier version of the Replication Server as the primary replication server, Replication Agent sends the **dump transaction** command without the new clause.

If you are using an earlier version of Replication Server as the replicate server, the primary Replication Server sends the **dump transaction** command without the new clause.

For more information about **dump transaction** enhancement, see the *Replication Server Design Guide*.

Distributor Status Recording

With Replication Server 15.1, you can now save the distributor (DIST) status of a distributor thread in the RSSD.

A DIST thread reads transactions from the inbound queue and writes replicated transactions into the outbound queue. A DIST thread is created when Replication Server connects to the primary database, and can be suspended or resumed manually, or through a Replication Server

configuration. Resuming and suspending a DIST thread modifies the DIST status of the thread.

DIST status recording allows the DIST thread to retain its status even after a Replication Server has been shut down.

For more information about DIST status recording, see the *Replication Server Reference Manual*.

Enhanced text Update

Replication Server supports the replication of large objects such as **text** and **image** to non-ASE servers by passing a **writetext** command to DirectConnect Anywhere™, where it is converted to an **update** statement.

The **writetext** command includes large-object pointers that an **update** statement uses to search and propagate the replicate database. Most data servers have their own unique implementation of updating large objects. Therefore, large-object replication to these servers becomes slow and inefficient, often requiring a full table scan of the replicate database for a single update.

Replication Server 15.1 provides an option to include primary keys with **writetext** commands sent to DirectConnect Anywhere. With the primary keys, DirectConnect Anywhere can create **update** statements that can efficiently search and replicate the replicate database.

Replication Server 15.1 introduces the Data Server Interface (DSI) configuration parameter **dsi_alt_writetext**. Use **dsi_alt_writetext** to instruct the Replication Server to include a text pointer or a set of primary keys with the **writetext** command.

Note: You need ECDA 15.0 ESD #2 to use this feature.

For more information, see the *Replication Server Reference Manual*.

Adaptive Server Integer Identity Support

Replication Server 15.1 supports the replication of Adaptive Server datatypes used as *identity* values.

The Adaptive Server 15.0 allows you to use these datatypes as *identity* values:

- *bigint*
- *int*
- *numeric*
- *smallint*
- *tinyint*
- *unsigned bigint*
- *unsigned int*

- *unsigned smallint*

Replication Server supports replication of the above datatypes. When you create a replication definition for a table that contains an *identity* column, specify *identity* as the datatype for the column.

Stable Queue Manager Performance Enhancements

The Stable Queue Manager (SQM) performance has been enhanced to include stable queue caching, segment preallocation, and support for direct I/O file access.

Stable Queue Caching

Replication Server uses a simple caching mechanism to optimize I/O. This mechanism reduces write latency and improves reader speed, since data can usually be read quickly from the cache.

A cache is made up of multiple pages and each page is made up of multiple adjoining blocks. A cache is allocated for each queue at start-up time. Changing the page size changes the size of I/O in the stable queue devices. When a page is full, the entire page is written in one single write operation.

In stable queue caching, the page pointer moves forward and rotates back at the end of the cache. SQM flushes the current page if the writer has filled the message queue and is blocked when waiting for messages. Only blocks with data are written to a disk when flushing a page that is not full.

Commands to Configure Stable Queue Cache Parameters

Examples of stable queue cache parameters and the commands that you can use to configure them.

Example 1

Set the server-wide caching default value using:

```
configure replication server set sqm_cache_enable to
"on|off"
```

Example 2

Enable or disable the caching for a queue and override the server-level setting using:

```
alter queue q_number, q_type,
  set sqm_cache_enable to "on|off"
```

When **sqm_cache_enable** parameter is disabled, SQM module returns back to the earlier mechanism, which maintains a fixed 16K; one-block buffer.

Example 3

Set the server-wide page size default value using:

```
configure replication server set sqm_page_size to
"num_of_blocks"
```

Example 4

Set the page size for a specified queue using:

```
alter queue q_number, q_type, set sqm_page_size to
"num_of_blocks"
```

num_of_blocks specifies the number of 16K blocks in a page. Configuring the page size also sets the I/O size of Replication Server. For example, if you set the page size to 4, this instructs the Replication Server to write to stable queue in 64K chunks.

Example 5

Set the server-wide cache size default value using:

```
configure replication server set sqm_cache_size to
"num_pages"
```

Example 6

Set the cache size for a specified queue using:

```
alter queue q_number, q_type, set sqm_cache_size to
"num_pages"
```

num_pages specifies the number pages in the cache.

All SQM configuration commands are static, thus you must restart the server for these commands to take effect. See the *Replication Server Reference Manual* for detailed information about these configuration parameters.

Segment Preallocation

Replication Server 15.1 preallocates segments in the background to reduce segment allocation latency. Segment allocation imposes significant latency to writer threads especially when the RSSD is on a remote Adaptive Server.

When a writer thread needs a new segment, it checks whether the preallocated segment is available, if it is not, the thread requests to allocate the segment. Once the writer thread gets the new segment, a preallocation request is made so that the segment is allocated in the background. By the time the writer thread needs a new segment, it is already available.

Enable or disable segment preallocation using:

```
configure replication server set sqm_seg_prealloc to
"on|off"
```

This command is static, which means you must restart the server for it to take effect. It supports only the server-level configuration.

Support for Direct I/O File Access

For file system partitions, direct I/O reduces the I/O latency as compared to the synchronous I/O, DSYNC.

Note: Direct I/O is supported only on Sun Solaris SPARC.

Adjust the stable queue cache properly. A proper cache size ensures that most read transactions are completed within the cache. Configure direct I/O using:

```
configure replication server set sqm_write_flush to
"dio"
```

This command enables direct /IO and is effective only when the stable queue is on the file system. The direct I/O method allows the Replication Server to read or write directly to the disk without the buffering the file system.

This command is static, which means you must restart the server for it to take effect.

New Features in Replication Manager 15.1

Replication Manager supports several Replication Server 15.1 features.

Enhanced Support for Dynamic SQL

With Replication Manager 15.1, you can enable dynamic SQL replication in table replication definitions in mixed-version replication environments where the Replication Server version is 15.0.1 or later.

The Replication Manager GUI has been modified. The General tab of the Create/Alter Replication Definition dialog box now includes an option to replicate dynamic SQL. This option is selected by default. If you do not require dynamic SQL replication, unselect the option. If you are working with a version of Replication Server for which Replication Manager does not support dynamic SQL, the option is automatically disabled.

Note: If you modify the dynamic SQL replication property of a replication definition, the changes apply only to the modified replication definition. Other replication definitions for the same table remain unchanged.

See “Dynamic SQL for enhanced Replication Server performance” in the *Replication Server Administration Guide Volume 2* for detailed information about dynamic SQL replication in Replication Server.

Enhanced Support for Function Replication Definitions

With Replication Manager 15.1, you can create multiple function replication definitions with names that differ from the corresponding primary function names.

To support this feature, the following changes have been incorporated in the Replication Manager GUI:

- On the General tab of the Add New Function Replication Definition dialog box, after you create a function replication definition, the Add New Function Replication Definition icon is not replaced by the Function Replication Definitions folder. The Add New Function Replication Definition icon and the Function Replication Definitions folder coexist so that you can create multiple function replication definitions.
- On the General tab of the Add New Function Replication Definition dialog box, the Replication Definition text field is now available for you to create your own function replication definition name. You can create a function replication definition name that differs from the selected stored procedure in the Replication Definition field.

- The Replicate Stored Procedure option available on the General tab instead of the Advanced tab.

See the *Replication Server Administration Guide Volume 1* for detailed information about working with function replication.

Support for Large-Object Datatypes

With Replication Manager 15.1, you can manage large-object (LOB) datatypes that are defined in your replication environment.

Replication Manager supports Microsoft SQL Server LOB datatypes of *varchar(max)*, *nvarchar(max)*, and *varbinary(max)*. However, if a column contains any of these LOB datatypes, the column cannot be a primary key, or marked as searchable.

The following changes have been made in the Replication Manager GUI:

- If you have LOB datatypes in your replication environment, the Replication Definition Datatype and Published Datatype lists in the Create Replication Definition dialog box display LOB.
- On the Columns tab in the Add New Table Replication Definition dialog, when you select a column that contains *varchar(max)*, *nvarchar(max)*, and *varbinary(max)* datatypes, the Primary Key and Searchable options are disabled.

See the *Replication Server Administration Guide Volume 1* for detailed information about working with LOB datatypes.

Sybase Central 6.0

Replication Manager 15.1 runs in Sybase Central 6.0.

Support for opaque Datatypes

In Replication Manager 15.1, you can replicate `opaque` data. `opaque` data can store unspecified or inconsistent values such as the `anydata` datatype of Oracle or the `sql_variant` datatype of Microsoft SQL Server.

This is how Replication Manager supports `opaque` data:

- In a mixed-version replication environment, you can replicate `opaque` data when the Replication Server is version 15.0.1 or later.
- The Replication Manager GUI has been modified. On the Columns tab of the Add New Table Replication Definition window, a new option “opaque” appears on the Replication Definition list in the Datatypes area.

- You cannot make columns with `opaque` data searchable.
- The `opaque` datatype is supported both in function and table replication definitions.

For detailed information about replication of supported datatypes, see the *Replication Server Reference Manual*.

Support for timestamp Datatypes

Replication Manager 15.1 introduces `timestamp` replication.

- In a mixed-version replication environment, you can replicate `timestamp` datatypes where the Replication Server version is 15.0.1 or later.
- The Replication Manager GUI has been modified. On the Columns tab of the Add New Table Replication Definition window, a new option “time stamp” appears on the Replication Definition list in the Datatypes area.
- `timestamp` datatype is supported both in function and table replication definitions.

For detailed information about replication of supported datatypes, see the *Replication Server Reference Manual*.

Obtaining Help and Additional Information

Use the Sybase Getting Started CD, Product Documentation site, and online help to learn more about this product release.

- The Getting Started CD (or download) – contains release bulletins and installation guides in PDF format, and may contain other documents or updated information.
- Product Documentation at *http://sybooks.sybase.com/* – is an online version of Sybase documentation that you can access using a standard Web browser. You can browse documents online, or download them as PDFs. In addition to product documentation, the Web site also has links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, Community Forums/Newsgroups, and other resources.
- Online help in the product, if available.

To read or print PDF documents, you need Adobe Acrobat Reader, which is available as a free download from the *Adobe* Web site.

Note: A more recent release bulletin, with critical product or document information added after the product release, may be available from the Product Documentation Web site.

Technical Support

Get support for Sybase products.

If your organization has purchased a support contract for this product, then one or more of your colleagues is designated as an authorized support contact. If you have any questions, or if you need assistance during the installation process, ask a designated person to contact Sybase Technical Support or the Sybase subsidiary in your area.

Downloading Sybase EBFs and Maintenance Reports

Get EBFs and maintenance reports from the Sybase Web site.

1. Point your Web browser to *http://www.sybase.com/support*.
2. From the menu bar or the slide-out menu, under **Support**, choose **EBFs/Maintenance**.
3. If prompted, enter your MySybase user name and password.
4. (Optional) Select a filter from the **Display** drop-down list, select a time frame, and click **Go**.
5. Select a product.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as an authorized support contact. If

you have not registered, but have valid information provided by your Sybase representative or through your support contract, click **My Account** to add the "Technical Support Contact" role to your MySybase profile.

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

Sybase Product and Component Certifications

Certification reports verify Sybase product performance on a particular platform.

To find the latest information about certifications:

- For partner product certifications, go to *http://www.sybase.com/detail_list?id=9784*
- For platform certifications, go to *http://certification.sybase.com/ucr/search.do*

Creating a MySybase Profile

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

1. Go to *http://www.sybase.com/mysybase*.
2. Click **Register Now**.

Accessibility Features

Accessibility ensures access to electronic information for all users, including those with disabilities.

Documentation for Sybase products is available in an HTML version that is designed for accessibility.

Vision impaired users can navigate through the online document with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Sybase HTML documentation has been tested for compliance with accessibility requirements of Section 508 of the U.S Rehabilitation Act. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

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

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

You may find additional information about accessibility features in the product documentation.

Index

D

E

F

H

O

P

Q

R

S

T

U

V

W