



New Features Guide

Replication Server[®] 15.6

DOCUMENT ID: DC00783-01-1560-02

LAST REVISED: July 2011

Copyright © 2011 by Sybase, Inc. All rights reserved.

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

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

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

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

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

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

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

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

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

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

Contents

Conventions	1
New Feature in Replication Server Version 15.6 ESD #1	5
Replication Server and Sybase IQ InfoPrimer Integration	5
Using the Replication Server and Sybase IQ InfoPrimer Integration	6
Parameters	11
Replication Server Components	12
Default Datatype Translation	15
Unsupported Features	16
Licensing	16
New Features in Replication Server 15.6	17
Replication Server Licensing	17
Subcapacity Licensing	17
Replication Server 15.6 Product Editions and Licenses	17
Replication from Oracle to Sybase IQ Using Real-Time Loading	19
Real-Time Loading Solution	19
Sybase IQ Replicate Data Servers	23
Replication Intrusions and Impacts on Sybase IQ	23
Replicate Database Connectivity for Sybase IQ	24
Sybase IQ Replicate Database Permissions	25
Sybase IQ Replicate Database Configuration Issues	26
Scenario for Replication to Sybase IQ	29
Tables with Referential Constraints	33
Display RTL Information	34

Net-Change Database	35
Mixed-Version Support and Backward Compatibility	36
Migrating from the Staging Solution to RTL	36
Performance Enhancements	38
Enhanced Retry Mechanism in HVAR and RTL	38
Increased Queue Block Size Enhancement	39
Usability and Process Improvements	45
Replicate Database Resynchronization for Adaptive Server	45
Delete Exceptions by Range	56
Controlling Row Count Validation	60
Display Table Names in Row Count Validation Error Message	61
Seamless Upgrade	62
Enhancements to Adaptive Server Replication Support	63
In-Memory and Relaxed-Durability Databases ...	63
Bulk Copy-in of image and Java Datatypes	64
New Features in Replication Server Version 15.5	67
Replication Server 15.5 Product Editions and Licenses	67
Sybase IQ Replication Using Real-Time Loading	68
Enhancements to Heterogeneous Replication Support	69
Parallel DSI Support in a Heterogeneous Environment	69
Heterogeneous Warm Standby Support for Oracle	71
Trigger Control at the Oracle Replicate Database	71
Performance Enhancements	72
Replication Server – Advanced Services Option	72

Dynamic SQL Enhancements	74
Function-String Efficiency Improvements	75
Usability and Process Enhancements	76
Enhanced Replication Definition Change	
Request Process	77
Replication Task Scheduling	78
Replication Delay	79
Replicate Database Resynchronization	79
Row Count Validation Changes	80
Enhanced alter error Class	81
Toolset for Implementing a Reference	
Replication Environment	81
Enhanced admin who Command	83
Database Generation Number Reset	84
Insertion of rs_ticket Markers into the Inbound	
Queue	84
Changes to Default Settings and Reserved Words	84
Changes to Parameter Default Values	85
Changes to the RSSD Locking Schema	86
Reserved Words	86
Enhancements to Adaptive Server Replication	
Support	86
bigdatetime and bigtime Replication	86
Deferred Name Resolution	87
SQL Statement Replication Threshold Setting ...	88
Incremental Data Transfer	88
In-Memory and Relaxed-Durability Databases ...	88
Mixed-Version Environments	90
Newly Supported Operating Systems	90
Support for 64-bit Computing Platforms	90
Changes to Replication Server Configuration	
Parameters	91
Changes to memory_limit Configuration	
Parameter	92
New Feature in Replication Manager 15.5	93

Enabling bigdatetime and bigtime Replication	93
New Features in Replication Server Version 15.2	95
Support for DSI Bulk Copy-in	95
Enhanced Subscription Materialization	95
New Connection Parameters	96
New Counters for Bulk Copy-in	97
Limitations	97
Non-blocking Commit	98
Adaptive Server Delayed Commit Feature	98
dsi_non_blocking_commit Configuration Parameter	99
rs_non_blocking_commit System Function	100
rs_non_blocking_commit_flush System Function	100
Non-Adaptive Server Databases Supported	101
Quoted Identifiers	101
Configuration Parameter to Enable Quoted Identifier Support	101
Commands to Mark Identifiers as Quoted	102
rs_set_quoted_identifier Function String	103
Changes to rs_helprep	104
Replication Server Gateway	107
Cascading Connection	107
Command to Enable Replication Server Gateway	107
Commands to Track Connections	108
Command to Drop Connection	109
Row Count Validation for Non-SQL Statement Replication	109
Command to Create Replication Server Error Classes	110
Command to Assign Error Actions	111
Stored Procedures to Display Replication Server Error Classes	112

Replication Server System Database	
Modifications	112
SQL Statement Replication	113
Enabling SQL Statement Replication	113
Modifications to System Configuration	113
SQL Statement Replication Configuration	115
Row Count Validation for SQL Statement	
Replication	118
Warm Standby Database Configuration for SQL	
Replication	119
Configuring Warm Standby Database for SQL	
Replication	120
Replication Server System Database	
Modifications	120
Non-Adaptive Server Error Class Support	120
Default Non-ASE Error Classes	121
Modified create error class Command	121
Modified alter error class Command	122
Non-Adaptive Server Replication Support	
Enhancements	122
Simplified Installation and Configuration	123
Connection Profiles	123
New Features in Replication Server Version 15.1	127
Dynamic SQL Enhancements	127
Function Replication Enhancements	128
Adaptive Server Shared-Disk Cluster Support	129
Enhanced Monitors and Counters	129
New Active Object Counters	130
New Procedure Interface	130
Improved Stable Queue Management	131
Changes to sysadmin dump_queue	131
Changes to sysadmin sqt_dump_queue	131
Modified resume connection Command	132
Modified sysadmin log_first_tran Command	132
New sysadmin sqm_zap_tran Command	132

New sysadmin sqm_unzap_tran Command	133
New sysadmin dump_tran Command	133
Changes to the locales Directory	133
Extended Password Encryption Support	134
rs_ticket Stored Procedure Version 2	134
New Replication Server Counters	135
Extended Support for Large Object Datatypes	136
Partial Update of Large Object Datatypes	137
Extended timestamp Support	137
New opaque Datatype	138
Dump Transaction Enhancements	138
New dump Subcommand Parameters	138
rs_dumptran Modification	139
Distributor Status Recording	139
Enhanced text Update	140
Adaptive Server Integer Identity Support	140
Stable Queue Manager Performance Enhancements	
.....	141
Stable Queue Caching	141
Segment Preallocation	142
Support for Direct I/O File Access	143
New Features in Replication Manager 15.1	145
Enhanced Support for Dynamic SQL	145
Enhanced Support for Function Replication	
Definitions	145
Support for Large-Object Datatypes	146
Sybase Central 6.0	146
Support for opaque Datatypes	146
Support for timestamp Datatypes	147
New Features in Replication Server Version 15.0.1	149
Configuration and Tuning Enhancements	149
Dynamic Configuration Parameters	149
admin config	150
Monitors and Counters Enhancements	150
Enhanced admin stats Command	150

cancel Option	151
Asynchronous Command Status Check	151
Storage of Previously Saved Data in RSSD	152
Dynamic SQL for Enhanced Replication Server	
Performance	152
Commands to Configure Dynamic SQL	153
Master Database Replication	153
rs_subcmp Enhancement	154
New Command Line and Configuration File	
Parameters	155
Examples	156
Schema Comparison	156
Schema Types Supported by rs_subcmp	156
Schema Subtypes Supported by rs_subcmp	157
New Command Line Parameters	157
New Configuration File Parameters	158
Schema Comparison Examples	158
Manual Data Reconciliation	159
New rs_subcmp Parameters	159
New Features in Replication Manager 15.0.1	161
Dynamic Configuration Support	161
Heterogeneous Data Server Support	161
Non-Sybase Data Server Support	162
Replication Agents and Mirror Replication	
Agents Support	162
Replication Manager Plug-in Enhancements	163
New Features in Replication Server Version 15.0	165
Support for Longer Identifiers	165
New Datatype: bigint	166
Range and Storage Size of Signed Integer	
Datatypes	166
New Unsigned Integer Datatypes	166
New Unicode Datatype: unitext	167
Computed Columns Replication	168
rs_subcmp Support for Computed Columns	169

Encrypted Columns Replication	169
Partitioned Tables Replication	169
Larger Disk Partitions	171
Larger text and image Size	171
Embedded Replication Server System Database	172
New Password Encryption Algorithm	172
Commands to Migrate to the New Algorithm	173
Mixed-Version Support Enhancement	173
New Interface for Monitors and Counters	174
Isolation Level Support	175
Bidirectional Replication Support for DDL in MSA	176
Setting Up Bidirectional Replication	176
Command Batching for Non-ASE Servers	177
SySAM License Management Changes	177
New Features in Replication Manager 15.0	179
New User Interface Features	179
Two-Tier Management Solution	179
Three-Tier Management Solution	179
Replication Manager Plug-in Replaces	
Replication Server Plug-in	180
Online Help	180
Status Visual Monitor	180
Event Log Pane	181
Background Processing	181
Replication Manager Logging Enhancement	182
Script Editors	182
Replication Server Features Supported	183
New Datatypes Supported	183
DirectConnect Support	183
Replication Support	183
Management of Route Upgrades	184
Troubleshooting Replication Problems	184
Option to Hide Connection Status	186
Warm Standby Wizards	187
Thread Management	190

Replication Monitoring Services	193
Three-Tier Management Solution	193
RMS and RSM Feature Comparison	194
Supported Replication Environment Servers	195
Starting RMS	195
Stopping RMS	196
Connecting to RMS in Sybase Central	196
Replication Environment Management Using RMS ...	197
Adding a Server	197
Dropping a Server	198
Viewing Monitored Objects in Sybase Central ...	198
Setting Configuration Parameters for Monitored Replication Objects	198
Management of a Logical Group of Servers	199
Suspending or Resuming Components in the Replication Environment	199
Shutting Down Monitored Servers	199
Server Rollup Status	199
Latency and Heartbeat Information Generation	200
Creating an Event Trigger for a Replication Server	200
Related API Commands	201
New Features in Replication Server Version 12.6	203
Multisite Availability	203
Symmetric Multiprocessors Support	204
The Embedded RSSD	205
Performance Enhancements	205
Better Management of Empty Transactions	205
Internal Commit Control for Parallel Processing	206
New Replication Server Configuration Parameters	206
New Database Configuration Parameters	206
Changed Database Configuration Parameters .	207

date and time Datatypes	207
Support for Sending Encrypted Passwords	208
New Bulk Materialization Method	209
Chinese Character Set (GB18030) Support	209
Obtaining Help and Additional Information	211
Technical Support	211
Downloading Sybase EBFs and Maintenance Reports	211
Sybase Product and Component Certifications	212
Creating a MySybase Profile	212
Accessibility Features	212
Index	215

Conventions

Learn about the style and syntax conventions used in Sybase® documentation.

Style conventions

Key	Definition
<code>monospaced(fixed-width)</code>	<ul style="list-style-type: none"> • SQL and program code • Commands to be entered exactly as shown • File names • Directory names
<i>italic monospaced</i>	In SQL or program code snippets, placeholders for user-specified values (see example below).
<i>italic</i>	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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 ODBC – Replication Agent for IBM DB2 UDB and Microsoft SQL Server
- Replication Agent for Oracle – Replication Agent for Oracle

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.

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.

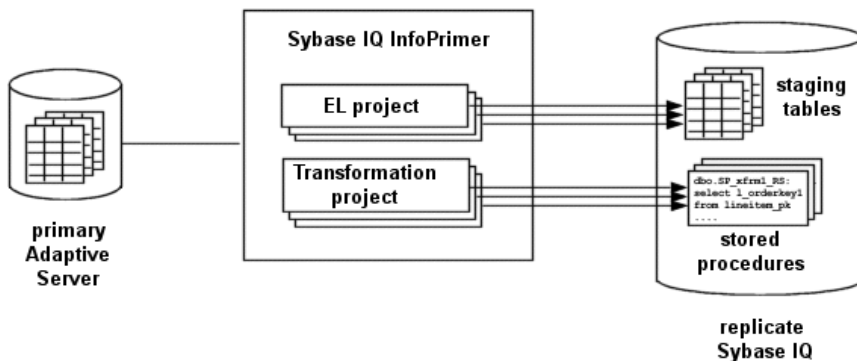
1. Before materialization:

- 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.

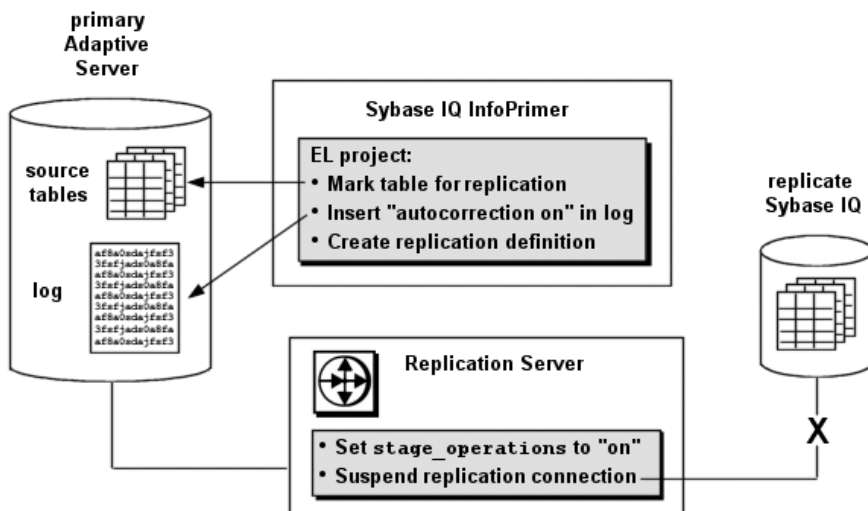


- 2.** In your Replication Server instance, use the **stage_operations** connection parameter to configure the replicate database connection to stage operations for the tables specified in your EL project.

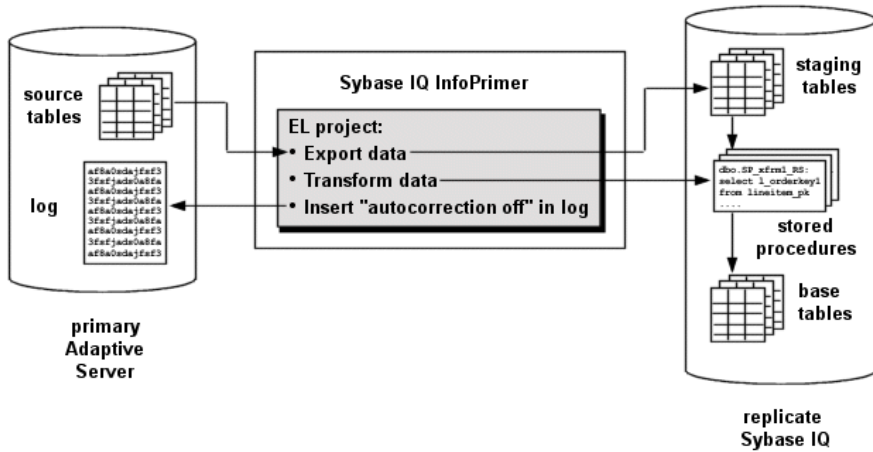
Note: If `stage_operations` is set to on, Replication Server ignores the setting of `dsi_compile_enable` and enables RTL for the connection. Operations are compiled, as when `dsi_compile_enable` is set to on, and then staged.

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.

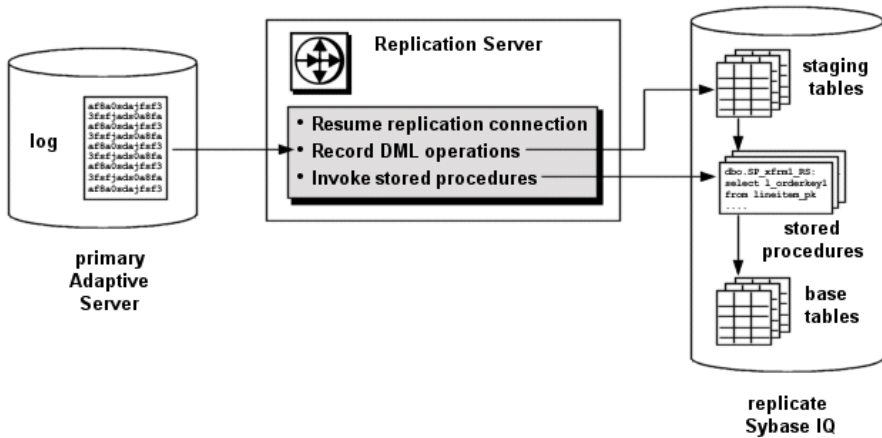


3. Your Sybase IQ InfoPrimer EL project exports primary data for each table into the corresponding staging tables on the replicate Sybase IQ, executes the transformation stored procedures, and inserts an `autocorrection off` record in the primary database log.



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.



See also

- *The rs_status Table* on page 12

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 off. For example:

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

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
action	varchar (1)	<ul style="list-style-type: none"> • 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)	<ul style="list-style-type: none"> • P – action in progress • X – execution complete • E – execution error

Column	Datatype	Description
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

Licensing

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

Table 1. 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

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 2. 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 3. 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.

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 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)` work table.
2. RTL performs an **insert** into the work table:

```
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 28

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:

- Triggers defined at the replicate database do not fire when the RTL compiler removes rows. Triggers require every logged SQL statement to be applied. If you require table triggers to fire all the time, 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**. An update on a table without table replication definitions immediately changes the table status to noncompilable.
- 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 issue 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. For example in:

- Sybase IQ 15.0 and later:

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

- Sybase IQ versions earlier than 15.0 such as, Sybase IQ 12.7:

```
ALTER DBSPACE dbspace-name ADD 2 GB
```

See the Sybase IQ documentation for your version for more information.

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 requires 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 work tables 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 set up 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`

New Features in Replication Server 15.6

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-compileable 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 **rs_oracle_to_iq** connection profile, and specify your replicate Sybase IQ data server and database.

For example:

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

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

Once you alter the connection to Sybase IQ and set **dsi_compile_enable** on, 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 100,000 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 autosynchronization 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. For example, after you enable RTL, you can set different values for the parameters:

- 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 "**alter connection**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual* for full descriptions of **dsi_compile_max_cmds**, **dsi_bulk_threshold**, and **dsi_command_convert**.

See also

- *Enhanced Retry Mechanism in HVAR and RTL* on page 38
- *Controlling Row Count Validation* on page 60

System Table Support in Replication Server

Replication Server uses the `rs_tbconfig` table to store 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.

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

See also

- *Replicate Database Connectivity for Sybase IQ* on page 24

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 `lcl_character_case` Replication Agent for Oracle configuration parameter. See "[lcl_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 21

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 `iqdb` database of the `IQSRVR` data server (`IQSRVR.iqdb`), enter:

```
admin config, "connection", IQSRVR, iqdb
```
 - To verify that **dsi_compile_enable** is **on** for the connection to `IQSRVR.iqdb`, enter:

```
admin config, "connection", IQSRVR, iqdb, 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", IQSRVR, iqdb, "enable"
```

Note: You must enclose "enable" in quotes because it is a reserved word in Replication Server. See "Reserved words," in Chapter 2, "Topics" in the *Replication Server Reference Manual* for the list of reserved words.

- Table-level:
 - To display all configuration parameters after using **dsi_command_convert** to set **d2none** on the `tbl` table in the `iqdb` database of the IQSRVR data server, enter:


```
admin config, "table", IQSRVR, iqdb
```

See "**admin config**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual* for syntax, parameters, and more examples.

Display Table References

Use the **rs_helprep** stored procedure 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 "**rs_helprep**," in Chapter 6, "RSSD Stored Procedures" in the *Replication Server Reference Manual* for syntax, parameters, and more examples.

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.

1. To access net-change database instances, use the Sybase IQ **insert ... LOCATION** command. See *Sybase IQ Reference: Statements and Options > SQL Statements > INSERT statement*.

Replication Server allows this external process.

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

See "**sysadmin cdb**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual*.

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 inbound 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 rssid_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 28

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 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:

New Features in Replication Server 15.6

- 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 17

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_locator` 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
|           |
```

New Features in Replication Server 15.6

```
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

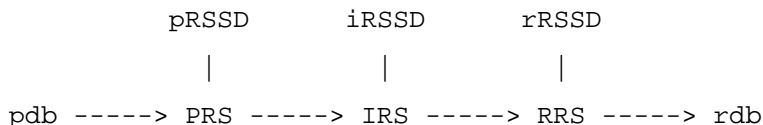
Learn to set the queue block size of the primary Replication Server in this example of a replication system with an intermediate route.

The replication system consists of:

- primary database – pdb

New Features in Replication Server 15.6

- 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



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

Learn how to configure resynchronization for Adaptive Server databases.

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.

With Adaptive Server 15.5 ESD #1, RepAgent, the Replication Agent for Adaptive Server, supports the full functionality of database resynchronization, such as automatic generation of the resync marker.

Configuring Database Resynchronization

Configure resynchronization for Adaptive Server databases.

1. Stop replication processing by suspending RepAgent.
2. Instruct Replication Server to be in resync mode. While 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.

You must use commands and parameters from both Replication Server and RepAgent for database resynchronization.

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 "**resume connection**," in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual*.

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**.

Sending 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

- *Sending the Dump Database Marker to Replication Server* on page 48
- *Resynchronizing Both the Primary and Replicate Databases from the Same Dump* on page 53

Obtaining 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.*

Sending 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. DSI no longer rejects committed transactions and all transactions that commit after the dump point, which is indicated by the dump database marker, are replicated.

Monitoring DSI Thread Information

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

State	Description
SkipUntil Re-sync	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.

Applying 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.

These are the messages:

- 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.
```

- When Replication Server receives the dump database marker:

```
DSI for data_server.database received and processed
Dump 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

There are procedures you must follow to resynchronize databases in different scenarios. After completing these procedures, the primary and replicate databases are transactionally consistent.

To execute these procedures, 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 Chapter 5, "Managing RepAgent and Supporting Adaptive Server" in the *Replication Server Administration Guide Volume 1*.

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 54

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, 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. 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.
```

After DSI processes the resync marker for the replicate database, you can apply the dump to the replicate database.

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 using:

```
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 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, 'current date'"
```

RepAgent automatically generates a dump database marker at a time based on the end of dump position that you obtained in step 5 and set 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:

New Features in Replication Server 15.6

```
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 only for Adaptive Server.

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, generate the **resync marker** manually:

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

```
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:

New Features in Replication Server 15.6

```
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 describes 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:

New Features in Replication Server 15.6

```
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 49

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** to delete a single transaction.

```
rs_delexception 1234
```

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

```
rs_delexception 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_systemtext* 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_start* 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_sysext` 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.

New Features in Replication Server 15.6

```
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 49

Resynchronizing a Replicate In-Memory or Relaxed-Durability Database with Bulk Materialization

Use the simulate atomic materialization method of bulk materialization to restore an in-memory or relaxed-durability database acting as the replicate 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 as the replicate 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 **sysadmin sqm_purge_queue**, in Chapter 3, "Replication Server Commands" in the *Replication Server Reference Manual*.

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 "Bulk materialization," in Chapter 11, "Managing Subscriptions" in the *Replication Server Administration Guide Volume 1*, to build the in-memory or relaxed-durability database. If you use:
 - Simulate atomic materialization — execute steps 4 to 9 in the procedure for "Simulate atomic materialization."
 - Simulate nonatomic materialization — execute steps 4 to 13 in the procedure for "Simulate nonatomic materialization."

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 4. 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 68
- *Replication Server – Advanced Services Option* on page 72

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 Chapter 11, “Sybase IQ Replicate Data Server Issues” in the *Replication Server Heterogeneous Replication Guide*.

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 90
- *Replication Server 15.5 Product Editions and Licenses* on page 67

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 5. 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 6. New Configuration Parameter

Parameter	Value	Default	Description
dsi_max_cmds_in_batch	Integer	100	Defines maximum number of source commands for which output commands can be batched. Range: 1 – 1000

Table 7. 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 8. 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 67

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 90

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 directly reads SQL statements from the Stable Queue Transaction thread (SQT) cache. This reduces contention between inbound and outbound queues, and improves Replication Server performance.

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, “Performance 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.

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'
```

Use **admin config** to display **replicate_minimal_columns** configuration information to verify that version 15.5 extends the parameter to connections.

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_locator*, and *rs_objects* system tables.

System Table	Description of Change
<i>rs_columns</i>	<i>version</i> column added.
<i>rs_locator</i>	<i>C</i> , <i>F</i> , and <i>S</i> values added to <i>type</i> column.
<i>rs_objects</i>	<i>active_inbound</i> , <i>attributes2</i> , and <i>version</i> 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_scheduledtxt* 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 9. 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 data 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 10. New and Changed Errors

server_error	Error Message	Default Error Action	Description
5185	Row count mismatch for the command executed on <code>'dataserver.database'</code> . The command impacted <code>x</code> rows, but it should impact <code>y</code> 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.

server_error	Error Message	Default Error Action	Description
5187	Row count mismatch for the autocorrection delete command executed on <code>'dataserver.database'</code> . The command deleted <code>x</code> rows, but it should delete <code>y</code> 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 <code>'dataserver.database'</code> . The delete command generated by <code>dsi_command_convert</code> deleted <code>x</code> rows, whereas it should delete <code>y</code> 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 109

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:

New Features in Replication Server Version 15.5

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 11. 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 12. 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 13. Additional Columns for DIST and DSI Thread Modules

Thread	Column Name	Description	Value
DIST	<i>RSTicket</i>	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	<i>SqtMaxCache</i>	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	<i>RSTicket</i>	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

Thread	Column Name	Description	Value
DSI	<i>dsi_rs_ticket_report</i>	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 14. Changes to Parameter Default Values

Parameter	Old Value	New Value	Downgrading from Version 15.5
exec_cmds_per_time-slice	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.
in-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.
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)	<ul style="list-style-type: none"> 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: <i>rs_columns, rs_functions, rs_objects</i>	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 93*

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**.

New Features in Replication Server Version 15.5

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 DML logging 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 15. 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 16. 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 , sql_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 86

New Feature in Replication Manager 15.5

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.

New Features in Replication Server Version 15.2

- 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

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`
```

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] computed]
  [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_coll` as a quoted identifier:

```
create replication definition repdef
  with primary at primaryDS.primaryDB
  with all tables named "foo"
  ("foo_coll" int quoted, "foo_col2" int)
  primary key ("foo_coll")
```

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_coll*:

```

alter replication definition repdef
    with replicate table named "foo"
    alter columns "foo_coll" 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

New Features in Replication Server Version 15.2

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		
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

```

New Features in Replication Server Version 15.2

```

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 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 118

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 17. Error Actions for Replication Server Error Classes

server_error	Error Message	Default Error Action	Description
5185	Row count mismatch for the command executed on <code>'dataserver.database'</code> . The command impacted <code>x</code> rows but it should impact <code>y</code> 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 <code>'dataserver.database'</code> . The command deleted <code>x</code> rows but it should delete <code>y</code> 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
<code>rs_classes</code>	<code>classtype</code> column includes a new “R” value for Replication Server error classes.
<code>rs_databases</code>	<code>rs_errorclassid</code> 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

New Features in Replication Server Version 15.2

- **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 dsl.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 'tbl'

      (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 109

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 18. Error Actions for SQL Statement Replication

server_error	Error Message	Default Error Action	Description
5186	Row count mismatch for the command executed on <code>'dataserver.database'</code> . The command impacted <code>x</code> rows but it should impact <code>y</code> 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
<i>rs_dbreps</i>	<i>status</i> 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.
<i>rs_dbsubsets</i>	<i>type</i> 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 .
<i>rs_objects</i>	<i>attributes</i> 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 122

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 19. Default Non-ASE Error Classes

Database	Class Name
IBM DB2	<i>rs_db2_error_class</i>
IBM UDB	<i>rs_udb_error_class</i>
Microsoft SQL Server	<i>rs_mssql_error_class</i>
Oracle	<i>rs_oracle_error_class</i>

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 120

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
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
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
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 replication	Standard	Standard ASE to DB2
rs_ase_to_udb replication	Standard	connection profile. Standard ASE to DB2
...		connection profile.

New Features in Replication Server Version 15.2

```
rs_db2_to_ase      Standard      Standard DB2 to ASE
replication
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
<i>rs_profile</i>	Stores currently defined profiles in Replication Server.
<i>rs_profdetail</i>	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_helpprep**, **rs_helpsub**, and **rs_helppubsub** to display dynamic SQL setting for each replication definition.

See “**rs_helpprep**”, “**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.

New Features in Replication Server Version 15.1

- 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 <i>text</i> , <i>image</i> , <i>java</i> 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.
DSIEDsqlDeallocSchema	Dynamic SQL statements deallocated at replicate database by a DSI/E because of schema change.
DSIEDsqlDeallocExecFail	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
<i>varchar(max)</i>	<i>text</i>
<i>nvarchar(max)</i>	<i>unitext</i>
<i>varbinary(max)</i>	<i>image</i>

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
```


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:

New Features in Replication Server Version 15.1

```
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.

New Features in Replication Manager 15.1

- 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*.

New Features in Replication Manager 15.1

New Features in Replication Server Version 15.0.1

Replication Server 15.0.1 introduces dynamic SQL, master database replication, schema comparison, and manual data reconciliation. Several enhancements have also been made to enable more efficient configuration and tuning of Replication Server.

Configuration and Tuning Enhancements

Replication Server 15.0.1 now supports dynamic configuration.

Dynamic configuration simplifies Replication Server customization and performance tuning by:

- Providing the ability to some of the Replication Server configuration parameters to handle dynamic modification of values.
- Introducing **admin config**, a Replication Command Language (RCL) that retrieves the server, connection, logical connection, and route parameters and their runtime values.

Dynamic Configuration Parameters

Several Replication Server configuration parameters are changed to dynamic, allowing you to change their values using the `configure replication server` command. You no longer need to restart the Replication Server for the new values to take effect. You can use the new **admin config** command to retrieve the values of these parameters.

Table 21. Dynamic Configuration Parameters

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

admin config

admin config lets you retrieve the values of the configuration parameters used to customize and tune the Replication Server.

In Replication Server 15.0.1, these parameters are categorized as:

- Server parameters
- Connection parameters
- Logical connection parameters
- Route parameters

Syntax

The **admin config** syntax is:

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

See Chapter 3, “Replication Server Commands” of the *Replication Server Reference Manual*, for detailed information in using **admin config** command.

Monitors and Counters Enhancements

The process of monitoring the counters that provide performance information has been improved in Replication Server 15.0.1. Sampling and saving statistics to the RSSD now runs asynchronously, allowing you to execute other Replication Server commands or terminate the session while the sampling continues in the background.

You can check the progress of asynchronous commands to prevent overlaps. When a command is denied due to overlapping, you can cancel the commands in progress to execute another command.

Enhanced admin stats Command

The **admin stats** command syntax and behavior has changed.

The modified command syntax is:

```
admin { stats | statistics } [, sysmon | "all"
| module_name [, inbound | outbound ] [, display_name ] ]
[, server[, database ] | instance_id ]
[, display |, save [, obs_interval ] ]
[, sample_period]
```

The behavior for the **save** parameter with a *sample_period* has also changed. The **admin stats** returns the session immediately. Its original *num_obs* option has been changed to *obs_interval*, which can be a numeric value in seconds, or a quoted time format string hh:mm[:ss].

Example

To start sampling and saving statistics to the RSSD for one hour and thirty minutes at 20-second intervals, enter:

```
admin stats, "all", save, 20, "01:30:00"
```

cancel Option

The new command syntax also supports the **cancel** option, which stops the currently running command.

```
admin { stats | statistics } , cancel
```

Note: For multiple observation intervals (*obs_interval*), the data that is saved at the time of cancellation is not deleted.

Replication Server checks the unfinished command to avoid corrupted sampling. If there is any command in progress, running either in the background or in other user sessions, the following commands cannot be executed:

- **admin stats** with any variation of save. For example:

```
admin stats, "all", save, 30
```

- **admin stats** with a sampling period. For example:

```
admin stats, "all", display, 120
```

- Commands that resets counters. For example:

```
admin stats, reset
```

When a command is denied, the following error message is returned, and **admin stats** with the **cancel** option is required to stop the unfinished command before executing another command.

```
Admin command failed. Existing command is currently in progress
```

Commands that display current statistics of your session are allowed to be executed even if there is another command in progress. For example:

```
admin stats, sqm, inbound, display
```

Asynchronous Command Status Check

The output of the existing **admin stats, status** command has been modified to include the status of the sampling command.

Example

```
1> admin stats, status
2> go
```

```
Command in progress, sampling period 00:30:00, time elapsed 00:02:32
```

```
Sybase Replication Server Statistics Configuration
```

```
=====
Configuration          Default          Current
-----
```

<code>stats_sampling</code>	<code>off</code>	<code>on</code>
<code>stats_show_zero_counters</code>	<code>off</code>	<code>off</code>
<code>stats_reset_rssd</code>	<code>on</code>	<code>on</code>

Storage of Previously Saved Data in RSSD

Replication Server 15.0.1 has a new configuration parameter, `stats_reset_rssd`, which allows you to optionally keep the old sampling data in the Replication Server System Database (RSSD) by setting its value to `off`. By default, `stats_reset_rssd` is turned on.

You can configure `stats_reset_rssd` using:

```
configure replication server
set stats_reset_rssd to {"on", "off"}
```

See Chapter 3, “Replication Server Commands” of the *Replication Server Reference Manual*, for detailed information about the `admin stats` command.

Dynamic SQL for Enhanced Replication Server Performance

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 execute them repeatedly.

Instead of sending SQL language commands to the target database, only the literals are sent on each execution, thereby eliminating the overheads brought by SQL statement syntax checks and optimized query plan builds.

You can use dynamic SQL in a user database connection for a language command if:

- The command is **insert**, **update**, or **delete**.
- There are no *text*, *image*, *unitext*, or *java* columns in the command.
- There are no NULL values in the **where** clause of an **update** or **delete** command.
- There are no more than 255 parameters in the command:
 - **insert** commands can have no more than 255 columns.
 - **update** commands can have no more than 255 columns in the **set** clause and **where** clause combined.
 - **delete** commands can have no more than 255 columns in the **where** clause.
- The command does not use user-defined function strings.

Limitations

Dynamic SQL has these limitations:

- If a table is replicated to a standby or MSA connection using an internal replication definition and dynamic SQL is enabled for the connection, any new replication definition for the table should define the column order consistent with the column order in the

primary database. Otherwise, the existing prepared statements may be invalidated, and may require the standby or MSA connection to be restarted.

- Dynamic SQL requires ASE or DirectConnect™ 12.6.1 ESD#2 for UDB as target database.

Commands to Configure Dynamic SQL

Commands used to configure dynamic SQL at the server or database level.

Configure dynamic SQL at a server or a connection level by issuing these commands:

```
configure replication server
set { dynamic_sql |
      dynamic_sql_cache_size |
      dynamic_sql_cache_management }
to value
```

```
alter connection to server.db
set { dynamic_sql |
      dynamic_sql_cache_size |
      dynamic_sql_cache_management }
to value
```

The server-level configurations provide the default values for the connections created or started in the future. For database level configurations:

- **dynamic_sql** – turns dynamic SQL on or off for a connection. Other dynamic SQL related configuration parameters take effect only if this parameter is set to on.
- **dynamic_sql_cache_size** – tells the Replication Server how many database objects may use the dynamic SQL for a connection. This parameter is provided to limit the resource demand on the data server.
- **dynamic_sql_cache_management** – manages the dynamic SQL cache for a connection. Once the dynamic SQL statements reaches **dynamic_sql_cache_size** for a connection, it either stops allocating new dynamic SQL statements if the value is **fixed**, or keeps the most recently used statements and deallocates the rest to make room for the new statements if the value is **mru**.

Master Database Replication

The Adaptive Server RepAgent thread now supports master database replication in Adaptive Server 15.0, ESD #2 and later. The master database can be replicated using warm standby with any version of Replication Server that supports warm standby, including Replication Server version 12.0 and later.

In addition, the master database can be replicated using multisite availability (MSA) with any version of Replication Server that supports MSA.

Replication of the master database is limited to DDL commands and system procedures used to manage logins and roles.

The supported DDL commands are:

- **alter role**
- **create role**
- **drop role**
- **grant role**
- **revoke role**

The supported system procedures are:

- **sp_addlogin**
- **sp_displaylevel**
- **sp_droplogin**
- **sp_locklogin**
- **sp_modifylogin**
- **sp_password**
- **sp_passwordpolicy**
- **sp_role**

rs_subcmp Enhancement

In Replication Server 15.0.1, **rs_subcmp** uses a hash algorithm to improve its performance. Hash algorithm compresses the data in primary and replicated tables. The compressed data is then fetched by **rs_subcmp**.

Instead of taking the entire row of data during comparison between the primary table and replicated table, **rs_subcmp** now transfers only the compressed data of each data row from the primary or replicated tables, and then verifies or reconciles inconsistencies between them.

With the hash algorithm, the amount of data to be transferred from the primary and replicated tables to **rs_subcmp** is reduced, decreasing the total running time of **rs_subcmp**.

Note: Sybase suggests that you set the Adaptive Server Enterprise parameter **default data cache** to 128M or higher to get a better comparison performance.

Limitations

The enhanced **rs_subcmp** requires Adaptive Server Enterprise 15.0.2 or later and cannot handle case sensitive comparison. It also cannot handle *text*, *unitext*, or *image* datatypes and does not allow you to specify the precision for the float datatype (maximum precision is used).

New Command Line and Configuration File Parameters

Replication Server 15.0.1 introduces new **rs_subcmp** parameters that support the use of the hash algorithm.

Table 22. New Command Line Parameters for Hash Algorithm

Parameter	Description
-h	Perform faster comparison.
-H	Normalize data when using hash algorithm across different platforms or character sets.

Table 23. New Configuration File Parameters for Hash Algorithm

Parameter	Description
FASTCMP	The same as command line parameter -h . The syntax is: FASTCMP = Y N. Values are: <ul style="list-style-type: none"> • Y – perform fast comparison using compressed data. • N (default) – perform normal comparison as before.
HASH_OPTION	The same as command line parameter -H . The syntax is: HASH_OPTION = <i>lsb</i> <i>msb</i> <i>unicode</i> <i>unicode_lsb</i> <i>unicode_msb</i> . If this parameter does not exist in the configuration file, by default, rs_subcmp uses the native byte order and character set.

String Options for the H and HASH_OPTION Parameters

Table 24. String options

Option	Description
<i>lsb</i>	All byte-order dependent data is normalized to lsb-first (little-endian) before data compression.
<i>msb</i>	All byte-order dependent data is normalized to msb-first (big-endian) byte order before data compression.
<i>unicode</i>	Character data is normalized to unicode (UTF-16) before data compression.

Note: UTF-16 string looks very much like an array of short integers and is byte-order dependent, thus Sybase suggests that you use *lsb* and *msb* with Unicode for platform independence. For example, use *unicode_lsb* or *unicode_msb*.

See Chapter 7, “Executable Programs” of the *Replication Server Reference Manual*, for more information about **rs_subcmp** and its parameters.

Examples

Illustrates the use of the enhanced **rs_subcmp**.

Example 1

Instructs **rs_subcmp** to get the compressed data and reconcile any inconsistencies between the primary and replicated tables, enter:

```
rs_subcmp [-h] [...]
```

Example 2

Instructs **rs_subcmp** to normalize data when using hash algorithm, enter the following command, where options can be *lsb*, *msb*, *unicode*, *unicode_lsb*, and *unicode_msb*:

```
rs_subcmp [-h -H options] [...]
```

Schema Comparison

In Replication Server 15.0.1, the **rs_subcmp** function has been expanded to include schema comparison between tables and databases. This is useful in comparing schema between two databases that may have the same data but different schemas.

Report and Reconciliation Files

A report file which details the comparison result between two tables or two databases is created after every schema comparison. The report file is named `reportPROCID.txt`. If inconsistencies exist, **rs_subcmp** creates a reconciliation script named `reconcilePROCID.sql`. The report file and the reconciliation script are saved in the same directory where you issued **rs_subcmp**.

Schema Types Supported by rs_subcmp

Enumerates the schema types supported by **rs_subcmp**.

Table 25. Schema Types Supported by rs_subcmp

Type	Description
A	All aliases in the database.
D	All defaults in the database.
E	All user-defined datatypes in the database.
G	All groups in the database.

Type	Description
R	All rules in the database.
T	All user tables in the database. Includes table elements such as indexes, keys, constraints, and triggers.
U	All users in the database.
V	All views in the database.
P	All procedures in the database.

Schema Subtypes Supported by rs_subcmp

Enumerates the schema subtypes supported by **rs_subcmp**.

Table 26. Schema Subtypes Supported by rs_subcmp

Type	Description
c	Constraint
d	Bind default
f	Foreign key
g	Grant
i	Index
m	Procedure mode
p	Primary key
r	Bind rule
t	Trigger

New Command Line Parameters

New **rs_subcmp** command line parameters support schema comparison.

Table 27. Command Line Parameters for Schema Comparison

Parameter Name	Description	Valid Values
-x	comparison flag	0 (default) – data comparison. 1 – database schema comparison. 2 – table schema comparison.

Parameter Name	Description	Valid Values
-X	filter flag	If the value starts with “+”, only the schema types are selected for comparison, and subtypes are ignored. Otherwise, the schema types and sub types are both unselected and not used for comparison.
-I	interface file	Interface file location.

New Configuration File Parameters

New configuration file parameters have been introduced to support schema comparison.

Table 28. Configuration File Parameters for Schema Comparison

Item Name	Description	Valid Values
SCHEMAFLAG	comparison flag	0 (default) – data comparison. 1 – database schema comparison. 2 – table schema comparison.
FILTER	filter flag	If the value starts with “+”, only the schema types are selected for comparison, and subtypes are ignored. Otherwise, the schema types and subtypes are both unselected and not used for comparison.
IFILE	interface file	Interface file location.

Schema Comparison Examples

Compares database or table schemas using `rs_subcmp`.

Example 1

Compares all schemas between two databases using the `config.cfg` file:

```
rs_subcmp -f config.cfg
```

The configuration file contains the following:

```
PDS = PASE
RDS = R2ASE
PDB = pubs2
PTABLE = authors
RTABLE = authors
PUSER = sa
RUSER = sa
PPWD =
RPWD =
SCHEMAFLAG = 1
```

Example 2

Compares schema between two databases without a configuration file:

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -usa -Psa_pwd
          -psa_pwd -xl
```

Example 3

Compares schema of two databases excluding index, trigger, and datatype:

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -usa -Psa_pwd
          -psa_pwd -xl -XitD
```

Example 4

Compares all table schemas and user schemas:

```
rs_subcmp -Spds -srds -Dpdb -drdb -Usa -usa -Psa_pwd
          -psa_pwd -xl -X+TU
```

Manual Data Reconciliation

The enhanced **rs_subcmp** command now supports manual reconciliation of inconsistent data through the creation of a reconciliation file. This allows you to verify the reconciliation statements before execution.

Limitation

The reconciliation file's SQL statements cannot contain *text*, *unitext*, or *image* data.

See Chapter 7, "Executable Programs" of the *Replication Server Reference Manual*, for more information on using these parameters with **rs_subcmp**.

New rs_subcmp Parameters

Replication Server 15.0.1 introduces two parameters to support **rs_subcmp** manual reconciliation.

Table 29. New Parameters for rs_subcmp Manual Reconciliation

Parameter	Description
-g	A command line parameter indicating to rs_subcmp to create a reconciliation file.
RECONCILE_FILE	A configuration file parameter. Values are: Y – create a reconciliation file. N (default) – do not create a reconciliation file.

New Features in Replication Manager 15.0.1

Replication Manager supports several Replication Server 15.0.1 features.

Dynamic Configuration Support

The Replication Manager (RM) plug-in has been modified to support the dynamic configuration feature of Replication Server 15.0.1. Versions of Replication Server earlier than 15.0.1 do not support dynamic configuration.

This feature allows the RM to dynamically retrieve all parameters that can be configured for the Replication Server, including the parameters for connections and routes.

Enhanced Properties Dialog Box

To support dynamic configuration, the Parameters tab of the Properties dialog box in RM has been enhanced to accurately display the parameter status information returned by the Replication Server. It includes:

- The Status field replaces the Restart Required field in RM 15.0. Its associated text field now has a wider range of information. The remaining fields and functions of the properties dialog box remain the same.
- The parameter information in the Run Value column may now contain the value “<server default>”. This may be used for parameters in the connections, logical connections, and routes to indicate that a parameter at that component level has not been set, and the Replication Server default value will be used. These types of parameters can have a default setting at the server level, or can have the parameter set for an individual connection, logical connection, or route.
- There may be some parameters for which the legal values are considered nonstandard so the Legal values field may be left blank. This is usually due to a parameter that may contain a textual setting and a numerical setting. In these cases, the Explanation field may contain further information regarding legal values for the parameter.

Heterogeneous Data Server Support

The RM plug-in now supports heterogeneous data servers, which lets you create connections, replication definitions, and subscriptions for non-Sybase data servers, including Oracle, Microsoft SQL Server, and IBM UDB.

Note: This version of RM does not support IBM DB2 data servers or Replication Agents, although Sybase provides a replication solution for IBM DB2 on mainframe.

Non-Sybase Data Server Support

To provide support for heterogeneous data servers, the RM displays the databases, tables, and stored procedures for non-Sybase data servers in a two-tier environment, the same as in the Adaptive Server data servers.

See Chapter 3, “Managing Replication Server with Sybase Central” of the *Replication Server Administration Guide Volume 1*, for more information about two- and three-tier management solutions.

The RM allows you to create, update, and delete both primary and replicate connections. It also allows you to create, update, and delete databases, tables, and function replication definitions and subscriptions. The RM user interface for non-Sybase data servers is identical to the current Adaptive Server interface.

Note: RM supports table, and function replication definition, and subscription, but does not support database replication definition, and subscription for non-Sybase data servers. RM also supports all connections, except logical ones, for non-Sybase data servers.

For non-Sybase data servers, the RM uses Replication Agents and DirectConnect to communicate with the data servers and to act as an interface for the RM. The RM uses an interface server to retrieve the metadata from the data servers. The Replication Agent serves as the interface server for the primary data servers, while DirectConnect serves as the interface server for the replicate data servers.

The interface server is not necessarily the server used to replicate data in the environment. For example, when defining an Oracle server that will be both a primary and a replicate data server, you only identify a Replication Agent as the interface server. This Replication Agent can be used to extract transactions from the Oracle server but a DirectConnect may also be needed to replicate data to the Oracle server. Also, if you are replicating from a Microsoft SQL Server, you can identify a Replication Agent as the interface server, but may require several Replication Agents to replicate data from several databases.

Replication Agents and Mirror Replication Agents Support

In version 15.0.1, the RM provides the ability to initialize the Sybase Replication Agent and displays its performance statistics in a two-tier environment. Additionally, the connection associated with the Replication Agent appears in the tree view hierarchy upon clicking the Replication Agent.

The RM also provides a feature that tests the connection between the primary data server and the Replication Agent, and the connection between the Replication Agent and the Replication Server.

The context menu for the Replication Agent now provides commands to test connections, display statistics, and initialize the Replication Agent. In the Sybase Central object tree, right-click the Replication Agent icon to display the context menu.

Replication Manager Plug-in Enhancements

Several items have been modified in the Replication Manager to provide heterogeneous support.

- The Add Server wizard to add non-Sybase data servers to a two-tier environment.
- The Add Connection wizard to create connections to non-Sybase databases.
- The Replication Agent context menu provides commands to initialize the agent, displays performance statistics, and tests network connections.
- The tree view hierarchy displays the connection associated with the Replication Agent under the actual agent.
- The Create Replication Definition dialog to create replication definitions for tables in a non-Sybase database. You can use user-defined datatypes to define non-Sybase datatypes.

New Features in Replication Server Version 15.0

Replication Server 15.0 introduces several new features and enhancements.

Support for Longer Identifiers

The limit of 30 bytes for selected database and replication object names (identifiers) is extended to 255 bytes for Replication Server version 15.0.

Longer identifiers are supported for these objects:

- Tables.
- Columns.
- Stored procedures.
- Parameters – for Replication Server functions and Adaptive Server stored procedures.
- Functions – for function replication definitions.
- Function strings.
- Replication definitions – including table replication definitions, function replication definitions, and database replication definitions.
- Publications.
- Articles.

All other database and replication object identifiers, login names and passwords, transaction names, subscription names, database names, and server names retain the existing 30-byte limit.

Parameter names for Replication Server functions and Adaptive Server stored procedures are the only identifiers that can begin with the @ character.

- Replication Server function parameter names can be as many as 256 bytes including the @ character.
- Adaptive Server stored procedure parameter names can be as many as 255 bytes including the @ character.

Note: `rs_subcmp` supports long identifiers for both table and column names.

Limitation

The **create function**, **alter function**, and **drop function** commands do not support long identifiers. The name of the function and the parameters of these commands cannot exceed 30 bytes.

New Datatype: bigint

Replication Server 15.0 adds support for the *bigint* datatype, which is a fixed-width, 8-byte datatype.

Like *int*, *smallint*, and *tinyint*, *bigint* is a signed integer datatype. *bigint* can hold whole numbers between -263 and $+(263 - 1)$.

Mixed-Version Issues

To fully support *bigint*, the primary and replicate Replication Server must have a site version of 15.0, and the LTL version must be 700. If the LTL version is less than 700 at connect-source time, RepAgent converts *bigint* columns to *numeric*.

Note: *rs_subcmp* supports the new *bigint* datatype.

Range and Storage Size of Signed Integer Datatypes

Describes the range and storage size of all integer datatypes.

Table 30. Signed Integer Datatypes

Integer Data-types	Range	Storage (in Bytes)
<i>bigint</i>	Whole numbers between -263 and $+(263 - 1)$, inclusive ($-9,233,372,036,854,775,808$ and $+9,233,372,036,854,775,807$)	8
<i>int</i>	Whole numbers between -231 and $+(231 - 1)$, inclusive ($-2,147,483,648$, and $+2,147,483,647$)	4
<i>smallint</i>	Whole numbers between -215 and $+(215 - 1)$, inclusive ($-32,768$ and $+32,767$)	2
<i>tinyint</i>	Positive whole numbers between 0 and 255, inclusive	1

New Unsigned Integer Datatypes

Replication Server version 15.0 supports four new unsigned integer datatypes.

You can use these unsigned datatypes in the same way as their signed equivalents:

- *unsigned tinyint* – *tinyint*
- *unsigned smallint* – *smallint*
- *unsigned int* – *int*
- *unsigned bigint* – *bigint*

Signed integers are whole positive or negative numbers. Unsigned integers are only whole positive numbers. The storage sizes of the signed and unsigned integers datatypes are the same. The table shows the range of positive numbers supported for each unsigned datatype.

Table 31. Unsigned Integer Datatypes

Integer Data-types	Range	Storage (in Bytes)
<i>unsigned bigint</i>	Whole numbers between 0 and 18,446,744, 073, 709,551,615, inclusive	8
<i>unsigned int</i>	Whole numbers between 0 and 4,294,967,295, inclusive	4
<i>unsigned smallint</i>	Whole numbers between 0 and 65535, inclusive	2
<i>unsigned tinyint</i>	Positive whole numbers between 0 and 255, inclusive	1

Mixed-Version Issues

To fully support the unsigned integer datatypes, the primary and replicate Replication Server must have a site version of 15.0, and the LTL version must be 700.

If the LTL version is less than 700 at connect-source time, RepAgent makes these datatype conversions:

- *unsigned bigint* → *numeric*
- *unsigned int* → *numeric*
- *unsigned smallint* → *int*
- *unsigned tinyint* → *tinyint*

Replication definitions created with unsigned integer datatypes are not sent to Replication Servers version 12.5.x and earlier. If a replication definition is subscribed by Replication Server version 12.5.x, it cannot be altered to add unsigned integer columns.

Note: `rs_subcmp` supports the new unsigned integer datatypes.

New Unicode Datatype: *unitext*

Replication Server version 15.0 adds support for the *unitext* datatype. *unitext* is a variable-width, nullable Unicode datatype. Although independent of the *text* datatype, *unitext* mirrors its behavior, and can be used wherever *text* is used.

Like the data in the other Unicode datatypes—*unichar* and *univarchar*—*unitext* data is encoded in UTF-16, which is essentially a 2-byte, fixed-width encoding of Unicode. *unitext* can hold as many as 1,073,741,823 Unicode characters, or the equivalent of 2,147,483,647 bytes. *unitext* has no connection to the default character set ID or the default sort order.

The main advantage of Unicode datatypes is efficiency. The UTF-16 character types are approximately 33% more space efficient than UTF-8 for Asian characters.

Mixed-version Issues

To fully support *unitext*, the primary and replicate Replication Server must have a site version of 15.0, the route version must be 15.0, and the LTL version must be 700. If the LTL version is less than 700 at connect-source time, RepAgent converts *unitext* columns to *image*.

Note: `rs_subcmp` supports the new *unitext* datatype.

Computed Columns Replication

Computed columns allow you to create an expression and place the result of the expression in a table column.

A computed column is:

- Materialized – when its value is computed for each insert or update. Materialized computed columns are stored in the same way as regular columns.
- Virtual – when its value is computed only when referenced in a query. Virtual computed columns are not stored in the table or index page.

A computed column expression is:

- Deterministic – when its value is the same each time it is evaluated.
- Nondeterministic – when its value may be different each time it is evaluated (for example, a date stamp).

Replication Server replicates materialized computed columns in DML statements in the same way it replicates other columns; it does not replicate virtual computed columns.

The replication of computed columns is supported by function strings. In Replication Server version 15.0, the class-level function string `rs_set_dml_on_computed` is applied at the replicate database DSI when a connection is established. It issues **set dml_on_computed "on"** after the **use database** statement. If the replicate Adaptive Server is version 12.5.x or earlier, the command is ignored.

Since Replication Server does not distinguish between computed and regular columns, there are no changes to the syntax for creating or altering replication definitions.

When creating or altering replication definitions for tables containing:

- Deterministic columns – you can choose whether to include those columns in the replication definition. Since deterministic columns always realize the same value, you can create the replication definition without them and allow each replicated insert and update to compute values at the replicate database.
- Nondeterministic columns – you must include nondeterministic computed columns in the replication definition to ensure that the primary and replicate databases remain synchronized.

rs_subcmp Support for Computed Columns

rs_subcmp supports comparison and reconciliation of materialized computed columns; it does not support virtual columns.

If the supported Adaptive Server provides the **set dml_on_computed "on"** command, **rs_subcmp** inserts and updates deterministic and nondeterministic materialized columns as regular columns.

Encrypted Columns Replication

Replication Server 15.0 supports replication of encrypted columns in Adaptive Server 15.0. Similar to Adaptive Server, Replication Server does not support encrypted columns that contain *text* and *image* data.

Replication Server replicates encrypted data as well as the encryption keys. For more information about this feature, see the *New Features Adaptive Server Enterprise 15.0 with Encrypted Columns*.

Note: **rs_subcmp** supports replication of encrypted columns in Adaptive Server.

Partitioned Tables Replication

In Replication Server 15.0, partitioned tables introduced in Adaptive Server® Enterprise 15.0 are replicated in a way that is similar to nonpartitioned tables. The **rs_truncate** system function in LTL, and the **rs_truncate** function and function string in Replication Server have been extended to support partitioned tables.

rs_truncate Changes in LTL

In LTL, the **rs_truncate** function has been extended to include partition names, as shown in the following syntax:

```
distribute command_tags applied [owner=owner_name]
table.rs_truncate [partition_name, [partition_name]...] yielding
```

You must assign a *partition_name* for each partition specified in the **truncate table** partition command.

rs_truncate Function Changes in Replication Server

In Replication Server, the **rs_truncate** function has been extended to accept parameters to support a new **truncate table partition** command, shown as follows:

```
truncate table table_name partition partition_name
```

When a truncate partition command is issued, the RepAgent sends the following LTL:

```
applied [owner=owner_name] table_name.rs_truncate, partition_name
_yd
```

rs_truncate Function String Changes in Replication Server

The partition names are passed as parameters to the **rs_truncate** function. The **rs_truncate** function string accepts position-based function-string parameters. The following is a position-based variable that specifies the parameter position in the function, in the LTL command:

```
?n!param?
```

A sample function string for **rs_truncate** with the position-based variable is as follows:

```
truncate table publishers partition ?1!param?, ?2!param?
```

Example 1

To replicate **truncate table partition** as a delete command, alter the function string in the following way:

```
alter function string publisher.rs_truncate
for rs_sqlserver_function_class
output language
'begin transaction
if (?1!param? = '')          /* NO parameter */
delete publisher
if (?1!param? = 'A')
delete publishers where c1 < 1000
if (?1!param? = 'B')
delete publishers where c1 >= 1000
commit transaction'
```

Example 2

To not to truncate table partitions at the replicate server, alter the function string to do nothing if there is a parameter, in the following way:

```
alter function string publisher.rs_truncate
for rs_sqlserver_function_class
output language
'if (?1!param? = '') delete publisher'
```

Mixed-Version Issues

For the RepAgent to send an **rs_truncate** applied subcommand with parameters, the site version must be 15.0 and the LTL version must be 700. If the LTL version is below 700, RepAgent skips the **rs_truncate** portion of the **distribute** command.

To replicate the **rs_truncate** function with a parameter to the replicate Replication Server, the route version must be 1500. If the route version is lower than 1500, the **rs_truncate** command with the parameter is skipped.

Larger Disk Partitions

Replication Server version 15.0 extends the maximum disk partition size from 2GB to 1TB. The new limit is applicable after the Replication Server site version is updated to 15.0.

Two new RCL commands support larger disk partitions:

- **create partition** – makes a partition available to Replication Server. This command replaces the existing **add partition** command. **add partition** is still supported, for backward compatibility with earlier versions. The syntax and usage of the two commands are identical. The command name has been changed to be consistent with other Replication Server command names.

- **alter partition** – increases the size of a partition. The syntax is:

```
alter partition logical_partition_name [ expand [ size = size ] ]
```

For example, to increase the size of the logical partition *p2* by 50MB, enter:

```
alter partition p2 expand size = 50
```

Larger text and image Size

In Replication Server 15.0, the **rs_subcmp** utility extends the maximum text and image size from 32KB to 2GB.

You can specify the new value as length in kilobytes, by either:

- Using the **-L *text_image_length_in_kilobytes*** parameter to set the new text and image length in kilobytes. For example, to set the new length to 64KB, enter:

```
rs_subcmp -L 64 -f subcmp.cfg
```

- Providing the new value for text and image length to the parameter **TEXT_IMG_LEN** in the configuration file. Start **rs_subcmp** using the modified configuration file:

```
rs_subcmp -f subcmp.cfg
```

subcmp.cfg file includes the following configuration parameter and its value:

```
TEXT_IMG_LEN = 64
```

Note: If the value is specified both in the command line and the configuration file, the value in the command line overwrites the value in the configuration file.

Embedded Replication Server System Database

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

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

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

```
configure replication server
set erssd_ra to <value>
```

Note: Sybase provides ERSSD in SQL Anywhere (SA) as an option, as well as continues to support the traditional RSSD in Adaptive Server Enterprise.

See also

- *The Embedded RSSD* on page 205

New Password Encryption Algorithm

Replication Server 15.0 uses the FIPS-certified Advanced Encryption Standard (AES) algorithm to encrypt new Replication Server user passwords. The AES algorithm uses the 128-bit encryption key and can be obtained from the Certicom Security Builder library.

Mixed-Version Issues

To fully support the new password encryption algorithm, both the Replication Server and the **rs_init** utility must have a site version of 15.0. If the site version is lower than 15.0, an error message appears and encryption is disabled.

Commands to Migrate to the New Algorithm

Lists commands to migrate existing encrypted passwords in the Replication Server configuration file, and the *rs_users* and *rs_maintusers* tables.

Table 32. Commands to Encrypt Passwords in New Algorithm

Task	Command
Migrate existing user passwords to the new algorithm	<pre>alter user user set password password</pre> <p>where:</p> <ul style="list-style-type: none"> <i>user</i> is the login name of the existing user. <i>password</i> is the existing password to encrypt using the new algorithm.
Migrate existing database maintenance user passwords to the new algorithm	<pre>alter connection to data_server.database set password to password</pre> <p>where <i>password</i> is the existing password to encrypt using the new algorithm.</p>
Migrate existing route user passwords to the new algorithm	<pre>alter route to dest_replication_server set password to passwd</pre> <p>where:</p> <ul style="list-style-type: none"> <i>dest_replication_server</i> is the name of the destination Replication Server. <i>passwd</i> is the existing password to encrypt using the new algorithm.
Migrate existing user passwords in the configuration file to the new algorithm	Use rs_init to encrypt the passwords using the new algorithm.

Mixed-Version Support Enhancement

In mixed-version environments, interaction between Replication Servers of different versions is restricted to the capabilities of the oldest version. Information associated with new features may not be available to Replication Servers of earlier versions.

Features in the new version, when sent to downstream Replication Servers of earlier versions, can break the multisite availability (MSA) replication and must be filtered out. In Replication Server 15.0, mixed-version support has been enhanced to provide a mechanism in which the later-version feature data is filtered out by the primary Replication Server before sending it out to the downstream Replication Servers of earlier versions.

New Configuration Parameter

To enable the primary Replication Server to block certain new feature commands not supported by earlier versions of Replication Server, a new Replication Server configuration parameter, **dist_stop_unsupported_cmd**, is available for all server and connection-level commands.

Table 33. Modified Commands for Mixed-Version Support

Replication Server Command	Syntax
configure replication server	<code>configure replication server set dist_stop_unsupported_cmd to [on off]</code>
alter connection	<code>alter connection srv.db set dist_stop_unsupported_cmd to [on off]</code>
alter logical connection	<code>alter logical connection lsrv.ldb set dist_stop_unsupported_cmd to [on off]</code>

By default, **dist_stop_unsupported_cmd** is set to off. When this parameter is set to on, DIST suspends if a command cannot be sent to some destination Replication Servers. Resume the DIST by skipping the entire transaction or by resetting this parameter to off.

When the parameter is set to off, the distributor (DIST) thread skips the newer commands to earlier version Replication Servers.

New Interface for Monitors and Counters

Replication Server 15.0 provides a new, simpler interface for monitoring the counters that provide performance information.

By default, all counters are inactive until you turn them on using either:

- The **admin stats** (**admin statistics**) command, which activates counters for a specific time period, or
- The Replication Server configuration parameter **stats_sampling**, which is a toggle that activates or deactivates counters.

Using **admin stats**, you can specify which counter statistics to report, whether to display those statistics on the screen or save them to the RSSD, and how many seconds to collect the statistics. **admin stats** lets you specify statistics for:

- Individual counters
- Individual modules

- The sysmon counters, which are a set of counters Sybase has identified as those most valuable for monitoring performance
- All counters

Viewing statistics on screen provides a point-in-time benchmark. Saving statistics to the RSSD lets you accumulate data, so you can see changes in statistics over time, and perform averages and other calculated values. You can specify a sampling period and the number of observations during that sampling period.

When you turn on counter activity using **stats_sampling**, the counters stay active until you turn them off.

New **admin stats** options are:

- **admin stats, backlog** – reports the current backlog in the inbound and outbound stable queues.
- **admin stats, { tps | cps | bps }** – reports throughput in transactions per second, commands per second, or bytes per second.
- **admin stats, status** – reports configuration information for counters.

Isolation Level Support

Isolation levels let you control the degree to which data can be accessed by other users during a transaction. With version 15.0, Replication Server decouples isolation levels and serialization methods for the replicate data server, and enables all isolation levels for replicate data servers that Replication Server supports. In versions earlier than 15.0, Replication Server supported only isolation level 3.

Through the use of custom function strings, Replication Server supports all isolation levels the replicate data servers may use. Support is not limited to the ANSI standard only.

Each isolation level specifies the types of actions that are not permitted while concurrent transactions are processing. Higher levels include the restrictions imposed by lower levels.

You can set the isolation level with the database configuration parameter **dsi_isolation_level**. The ANSI standard levels supported by Adaptive Server are:

- 0 – ensures that data written by one transaction represents the actual data.
- 1 – prevents dirty reads and ensures that data written by one transaction represents the actual data.
- 2 – prevents nonrepeatable reads and dirty reads, and ensures that data written by one transaction represents the actual data.
- 3 – prevents phantom rows, nonrepeatable reads, and dirty reads, and ensures that data written by one transaction represents the actual data.

The default value is the current transaction isolation level for the target data server.

New Features in Replication Server Version 15.0

For example, to select isolation level 2 for the Replication Server connection to the TOKYO_DS data server and *pubs2* database, enter:

```
alter connection to TOKYO_DS.pubs2
set dsi_isolation_level to '2'
```

In versions earlier than 15.0, you set isolation level 3 and the serialization method at the same time:

```
alter connection to TOKYO_DS.pubs2
set dsi_serialization_method to 'isolation_level_3'
```

With version 15.0, the equivalent is:

```
alter connection to TOKYO_DS.pubs2
set dsi_serialization_method to 'wait_for_start'
```

```
alter connection to TOKYO_DS.pubs2
set dsi_isolation_level to '3'
```

Bidirectional Replication Support for DDL in MSA

You can configure multisite availability (MSA) to set up a two-way replication of data definition language (DDL) transactions between two Adaptive Server databases. In version 15.0, bidirectional DDL replication support in MSA environments is supported only for non-warm standby databases.

Replication Server 15.0 supports this bidirectional replication using a new configuration parameter **dsi_replication_ddl**. When **dsi_replication_ddl** is set to on, DSI sends set replication off to the replicate database, which instructs it to mark the subsequent DDL transactions available in the system log not to be replicated. Therefore, these DDL transactions are not replicated back to the original database, which enables the DDL transactions replication in bidirectional MSA replication environment.

Setting Up Bidirectional Replication

Learn how to set up a two-way replication of data definition language (DDL) transactions between two Adaptive Server databases.

1. Create a bidirectional MSA replication environment.

For steps, see Chapter 12, “Managing Replicated Objects Using MultiSite Availability,” in the *Replication Server Administration Guide Volume 1*.

2. Grant “set session authorization” privilege to a maintenance user on the destination database, as shown in the following example:

```
grant set session authorization to maint_user
```

3. In the destination database, set **dsi_replication_ddl** to on to enable bidirectional DDL replication, as shown in the following example:

```
alter connection to dataserver.database set dsi_replication_ddl on
```

4. Replicate DDL transactions.

Command Batching for Non-ASE Servers

Replication Server 15.0 allows you to batch commands for non-ASE database servers. By batching commands, you may be able to achieve improved performance in Replication Server.

New Function Strings

Support for command batching to non-ASE servers is achieved through the use of two function strings, **rs_batch_start** and **rs_batch_end**. These function strings store the SQL translation needed for marking the beginning and end of command batches. Use of these function strings is not necessary for Adaptive Server Enterprise or any other data server where the function strings **rs_begin** and **rs_commit** already support the needed functionality.

New DSI Connection Parameter

A DSI connection parameter, **use_batch_markers**, is used to control the processing of the two function strings, **rs_batch_start** and **rs_batch_end**. You can set **use_batch_markers** with **alter connection** and **configure connection** commands. If **use_batch_markers** is set to on the function strings **rs_batch_start** and **rs_batch_end** are executed. The default is off.

New DSI Configuration Parameters

DSI must be correctly configured for batching of commands to the replicate data server. There are three DSI configuration parameters to consider for each connection that will be batching commands:

- **batch**
- **batch_begin**
- **use_batch_markers**

For more information about configuring DSI for non-ASE data servers, see Chapter 5, “Replication Server Issues,” in the *Replication Server Heterogeneous Replication Guide*.

SySAM License Management Changes

The Sybase Software Asset Management (SySAM) implementation has changed for this version of Replication Server.

- Asset management and reporting tools are provided with SySAM 2. These tools allow you to monitor license usage and compliance.
- A single installation method supports all editions of Replication Server.
- SySAM configuration is no longer optional.
- Flexible SySAM configuration options.

New Features in Replication Server Version 15.0

- SySAM licenses are no longer shipped along with order fulfillment. You must obtain license certificates from the Sybase Product Download Center (SPDC).
- SySAM license keys include information about the support plan you purchased. You must update these licenses whenever you renew your support plan.
- Licensing policies are strictly and consistently enforced.
- Replication Server can function under grace periods if cannot obtain a license. These grace periods allow you reasonable time to respond to the issues causing license failure. During the grace period, the Replication Server continues to function normally. However, the Replication Server or any of its features, will shut down at the end of the grace period if licensing issues remain unresolved.
- You can receive real-time e-mail notifications about licensing events.
- Licenses issued from SPDC include information about the host machine where the licenses will be deployed. These licenses cannot be used on another machine without being reissued from SPDC.

Note: Replication Server 15.0 does not require the REP_SSL license, as SSL now comes as a part of the basic REP_SERVER license.

These changes affect the Replication Server installation and configuration process. For pre-installation and SySAM installation information, see the *Replication Server Installation Guide* for your platform. Plan your SySAM deployment before installing Replication Server.

New Features in Replication Manager 15.0

Several new features have been introduced in Replication Manager, a plug-in to Sybase Central.

New User Interface Features

The Replication Manager supports several new interface features that promote ease-of-use and productivity.

Two-Tier Management Solution

Replication Manager can manage replication environments by connecting directly to servers in the environment without communicating through a management server layer. This two-tier management solution lets you manage small, simple replication environments with fewer than ten servers.

To set up a replication environment in Sybase Central, you select the servers contained in the environment from a drop-down list, which is derived from the local interfaces file, then provide a user name and password that the Replication Manager uses to connect to these servers. These passwords are encrypted when stored in the Sybase Central repository.

Three-Tier Management Solution

In a three-tier management solution, RMS server is a middle-management layer that monitors the status of the servers and other components in the replication environment.

If you are managing a large or complex replication environment, you might want to install the Replication Monitoring Services (RMS) server. Replication Manager connects to the servers in the environment through RMS.

The Replication Manager provides the client interface that displays the information provided by RMS.

See also

- *Replication Monitoring Services* on page 193

Replication Manager Plug-in Replaces Replication Server Plug-in

With Replication Server 15.0, the Replication Manager plug-in replaces the Replication Server plug-in as the complete management tool for developing, managing, and monitoring a Sybase Replication Server environment for a two-tier management solution.

In earlier versions, the Sybase Central Replication Server Manager included a Replication Server plug-in and a Replication Server Manager Server (RSM Server) to monitor, analyze, troubleshoot, and administer a replication system.

The Replication Server plug-in included a graphical user interface (GUI) integrated with Sybase Central. The Replication Manager plug-in has an interface that is similar to the Replication Server plug-in, but runs within the Sybase Central, Java Edition framework.

In addition, Replication Manager does not require the RSM Server to manage servers in a replication environment. It communicates directly with the Replication Server and Adaptive Servers in a two-tier management solution.

Online Help

Online help for Replication Manager is now available. The online help contains extensive topic-level help that provides a quick reference for all Replication Manager concepts and tasks.

Select Help from the Sybase Central main menu and then select Replication Manager online help.

Status Visual Monitor

The state of each object appears on the object icon, in the parent object Details list, and on the Properties dialog box for that object. You can monitor the status of servers, connections, routes, and queues.

Detail List

When you select an object in the left pane that contains subcomponents or function components, one or more tabs appear in the right pane of the Sybase Central window with lists of information. For most objects, a single tab called “Details” appears; this tab contains a list of general information about the object.

The Details list includes:

- Subcomponents, which are other replication or database objects that are contained in another object.
- Function components, which are components that invoke a wizard when double-clicked. For example, the Add Connection object is a function component that invokes the Add Connection wizard.

Event Log Pane

Replication Manager displays an event log in a pane at the bottom of the Sybase Central window.

The event log includes:

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

To show or hide the event log, select Event Log from the View menu.

See also

- *Replication Monitoring Services* on page 193

Background Processing

Several tasks performed by Replication Manager can be very time-consuming, such as creating a subscription that also materializes the table. These tasks are now performed in the background, allowing Sybase Central to continue to function.

When you start a time-consuming task, Replication Manager displays a message window indicating that a process is running. You can click Stop Process in this window to stop the process.

Note: The process continues even if the Background Process window is closed.

When a background task completes, the Replication Manager places an event entry in the event log.

Checking the Status of a Background Process

To see the status of a background process, you can open the Background Processes dialog box, which displays a list of all of the currently running background threads.

To access the Background Processes dialog box, select **Search | Background Processes**.

The Background Processes dialog box opens, showing:

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

Replication Manager Logging Enhancement

Replication Manager now uses the Sybase Central message logging feature to provide a log of all commands sent by the Replication Manager to any server.

Because the log may contain passwords (for example, passwords are needed to create a connection, and these passwords are saved in the log), the Replication Manager provides the ability to turn off command logging.

Turning Command Logging On or Off

Enable or disable command logging.

1. Right-click the **Replication Manager** object (at the top of the tree).
2. Select **Properties**.
3. In the Replication Manager properties dialog box:
 - Select **Write SQL Commands to Log** to enable command logging.
 - Unselect **Write SQL Commands to Log** to disable command logging.

Next

Turning on command logging can quickly fill the Sybase Central log, and may cause it to stop responding. If you turn command logging on, monitor the log closely.

Script Editors

Replication Manager provides two script editors, the Replication Command Language (RCL) script editor and the Structured Query Language (SQL) script editor. These editors operate identically, except the RCL script editor highlights RCL keywords, while the SQL script editor highlights SQL keywords.

You can use the script editor to view generated RCL commands, which include syntax to create connection and configuration parameters that can be used to create connections outside the Replication Manager.

With the script editors, you can:

- Select several Replication Server objects and generate RCL for all objects selected.
- Edit and save the generated RCL script.
- Load an RCL script from a file and add it to the current script.
- Resubmit an RCL script to Replication Server.

Accessing the Script Editor

Use the script editor to view generated RCL commands.

1. Select the Replication Server object for which you want to generate RCL.

2. Right-click the object and select **Generate RCL**.

The RCL Script Editor opens, showing the commands needed to create the object.

Replication Server Features Supported

Describes the Replication Server 15.0 features that the Replication Manager also supports.

New Datatypes Supported

Replication Manager supports the new datatypes supported by Replication Server 15.0.

- *bigint*
- unsigned integer datatypes
 - *unsigned tinyint*
 - *unsigned smallint*
 - *unsigned int*
 - *unsigned bigint*
- *unitext*

See also

- *New Features in Replication Server Version 15.0* on page 165

DirectConnect Support

In version 15.0, the Replication Manager manages a component that represents a Sybase DirectConnect data access server. The DirectConnect server acts as an Open Server gateway by converting the Open Client/Server protocol used by Replication Server to the native communication protocol used by the non-Sybase replicate database.

In Sybase Central, the DirectConnect data access server is managed as any other object in the replication environment. The Replication Manager displays the state on the icon of the DirectConnect data access server and on the parent object's Detail list. The state of the server reflects whether the server is available and the state of the back-end data servers.

Limitations

In version 15.0, the Replication Manager neither displays the error log nor sets configuration parameters for DirectConnect.

Replication Support

Replication Server provides replication at the database, table, and stored procedure levels. Replication Manager allows you to create a replication definition for a database, a table, or a stored procedure.

A replication definition for a stored procedure is called a function replication definition. You can create, alter, and delete function replication definitions and function subscriptions.

New Features in Replication Manager 15.0

For more information about creating, altering, and deleting replication definitions and subscriptions, see the Replication Manager online help.

Management of Route Upgrades

A route is a one-way message stream, carrying replication data, from a source Replication Server to a destination Replication Server. With Replication Server 15.0, you can upgrade routes using the Replication Manager. In earlier versions, the RSM Server provided the route upgrade capability.

The Replication Manager displays a set of all routes in the Replication Server that are eligible to be upgraded. You can select one single route at a time for upgrade. Upon selection of a route, the Replication Manager runs the upgrade process in the background and reports any errors or warnings in the event log. The new route upgrade implementation also lets you:

- Cancel a route upgrade process – the ability to cancel a route upgrade process. However, canceling can leave the replicate Replication Server in an unstable state.
- Recover from an unsuccessful upgrade – a recovery procedure to reset the replicate RSSD. An unsuccessful upgrade can result from either an abnormal termination of the upgrade process or a cancellation of the upgrade process.

For more information on managing routes using Replication Manager, see the Replication Manager online help.

Limitations

- You cannot currently upgrade multiple routes at the same time.
- Route upgrade is supported only in an environment domain using the two-tier solution.

Upgrading Routes Using Replication Manager

Upgrade the route of a Replication Server object using Replication Manager.

1. In the object tree, select a Replication Server object.
2. Right-click the Replication Server object and select **Upgrade Route**.
A dialog box lists all the routes that can be upgraded.
3. From the list, select a route and click the **Upgrade** button.

A background process starts, and indicates on completion whether the upgrade is successful or not.

Troubleshooting Replication Problems

An environment may occasionally stop replicating data, when a transaction is formatted incorrectly or when a server generates an error. To troubleshoot the situation, view the Replication Server queue data and exceptions log.

Using the Exceptions Log

Use the exceptions log to troubleshoot a problem with replication.

Perform these steps when a transaction is not processed because of a SQL error.

1. Issue a **resume** command with a **skip transaction** clause to the queue.
2. Right-click the connection and select **View Exceptions**.
The View Exceptions dialog box displays the exceptions log in the Exceptions Log table.
3. View the skipped transaction and the erroneous SQL in the exceptions log.
4. Filter the contents of the exceptions log by selecting these columns in the table: Origin Data Server, Origin Database, and Previous Hours.
5. To see the command language associated with a transaction, select it from the Exceptions Log table.
The Command Editor text box displays the SQL commands in the transaction.
6. Edit the transaction in the text box.
7. To resubmit the transaction to the replicate data server, select **Resubmit**.

Queue Data

Data that passes between servers (Adaptive Server, Replication Server, and so on) is stored in stable queues within Replication Server. The Replication Manager shows queue contents and queue usage statistics.

View Queue Data Dialog Box

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

The View Queue Data dialog box contains the following fields:

- Filter fields, which let you select the type of filters that the Replication Manager uses to display data from the queue. These filters include:
 - Column
 - Column value
 - Segment
 - Starting block
 - Number of blocks displayed
 - Number of rows displayed
 - Whether to start at the first active segment or not
 - Whether to include all data to the end of the segment
 - Whether to include all rows or not
 - Whether you want to show deleted data
 - Whether to view all data to the end of the queue

New Features in Replication Manager 15.0

- General buttons, which let you:
 - Display the queue data with the current filters
 - Close the dialog box
 - Purge the first transaction from the queue
 - Edit transactions
 - Delete transactions
 - Undelete transactions
 - Group transactions, which returns the Queue Data scrolling list display back to grouped transactions
- Queue Data scrolling list, which contains rows of data from the current queue. Each column contains specific information about the command and transaction contained in each row. For example, to sort the queue data by a specific column, select that column name. The Queue Data scrolling list refreshes, sorting the data according to that column. An arrow displays next to the column name to show that you have sorted the data by that column. The columns you can sort by include:
 - Segment
 - Transaction Name
 - Command
 - Origin Site
 - Origin Commit Time
 - Origin User
 - Transaction ID
 - Origin QID

Note: You can only delete, undelete, or purge queue transactions when Replication Server is in standalone mode. For more information, see “Managing Replication Server with Sybase Central” in the *Replication Server Administration Guide Volume 1*.

Viewing Queue Data

Filter and sort the data from a queue as an aid in troubleshooting transactions in the queue.

1. Right-click the queue which data you want to view.
2. Select **View Data**.
The View Queue Data dialog box opens.
3. To filter data shown, select one of the filter fields.
4. To sort the data, select segment, transaction, origin, size, status, commit time, or user.

Option to Hide Connection Status

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

Because the filtering state of the connection status is stored locally by the Replication Manager, different instances of the Replication Manager do not share filtering states. For

example, if you create a connection using one instance of the Replication Manager, and then set the Replication Agent status to “hide” for that connection, another Sybase Central plug-in instance monitoring the same environment does not filter the connection status because the filtering information is available only to the original Replication Manager instance.

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

Connection Status in Warm Standby Environments

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

Hiding Connection Status

Hides the state of the Replication Agent or the DSI thread.

1. Right-click the connection for which to hide the status.
2. Select **Hide Connection Status**.
3. Select one of these options:
 - **Hide the State of the Replication Agent** – hides the state of the Replication Agent thread in the Details list, on the Connection Properties dialog box, and in the rollup status for the Replication Server to which that Replication Agent thread is connected.
 - **Hide the State of the DSI Thread** – hides the state of the DSI thread in the Details list, on the Connection Properties dialog box, and in the rollup status for the Replication Server to which the DSI thread is associated.

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

Warm Standby Wizards

Replication Manager 15.0 introduces three wizards to add logical connections, active database connections, and standby database connections.

To create a warm standby environment, create these components in the specified order:

1. A logical connection
2. A connection to the active database
3. A connection to the standby database

Previously, these steps were only part of the Configure Replication Environment wizard, which lets you build a warm standby environment in one step. With this version of Replication Manager, you can use a separate wizard for each step in the process, which lets you drop and re-create connections as needed.

The three wizards are:

- Add Logical Connection – adds a logical connection whether you are basing this connection on an existing physical connection or not.
- Add Active Database – adds an active database connection to an existing logical connection.
- Add Standby Database – adds a standby database to an existing logical connection that already has an active database connection.

Creating a Logical Connection

Use the the Add Logical Connection wizard to add a logical connection. You can use the wizard whether or not you are basing the logical connection on an existing physical connection.

1. In the tree view, select the **Logical Connection** folder under the Replication Server object.
2. In the Details view, double-click **Add Logical Connection**.
The Add Logical Connection wizard starts.
3. On the Convert Existing Connection wizard page, verify that the Use an Existing Connection as the Active Connection option is not selected.
4. Enter the database name and data server name for the logical connection.
5. Review the summary information for the logical connection.
6. If everything looks correct, click **Finish**. Otherwise, click **Back** to return to an earlier page in the wizard and change the logical connection information.

Replication Manager creates the logical connection object.

Next

You must create the connections to the active and standby databases before you can have a working warm standby environment.

Creating an Active Database Connection

Use the Add Active Database wizard to add an active database connection to an existing logical connection.

1. In the Details view, right-click the logical connection object you created using the Add Logical Connection wizard.
2. Select **Add Active Database**.
The Add Active Database wizard starts.
3. Select the active server and active database.
4. Select the Replication Server that will manage the database connections.
5. Enter the user name and password of the maintenance user.
6. Select the user name and password that the RepAgent will use to connect to Replication Server.

If the RepAgent user does not exist, the wizard creates one for you and gives it a default name and password. Accept the defaults, or enter your own values.

7. Select **Enable all objects in the active database** to enable database objects.
8. Review the summary information about the replication environment.
9. If everything looks correct, click **Finish**. Otherwise, click **Back** to return to an earlier page in the wizard and change the replication environment information.

Replication Manager creates the active database connection.

Creating a Standby Database Connection

Use the Add Standby Database wizard to add a standby database to an existing logical connection that already has an active database connection.

1. In the Details view, right-click the logical connection object you created using the Add Logical Connection wizard.
2. Select **Add Standby Database**.
The Add Standby Database wizard starts.
3. Select the standby server and standby database.
4. Select the Replication Server that will manage the database connections.
5. Enter the user name and password of the maintenance user.
6. Select the user name and password that the RepAgent will use to connect to Replication Server.

If the RepAgent user does not exist, the wizard creates one for you and gives it a default name and password. Accept the defaults, or enter your own values.

7. Select the materialization method.
8. Select one of these options:
 - **Initialize Standby Database with Dump Load**, which initializes the standby database with the current data dump from the active database.
 - **Use Dump Marker in Transaction Log**, which replicates transactions that are executed between the time the active database is enabled and the time the data is dumped.
9. Review the summary information about the replication environment.
10. If everything looks correct, click **Finish**. Otherwise, click **Back** to return to an earlier page in the wizard and change the replication environment information.

Replication Manager creates the standby database connection.

Next

Before you use your warm standby environment, resume the DSI thread on the replicate database connection, if necessary.

Thread Management

Replication Manager displays the state of the threads in the Replication Server.

If the thread is directly related to another Replication Server component, such as a connection, route, or queue, then the thread component is represented in Sybase Central by the related component and its features. For example, the Replication Agent thread (RepAgent) or DSI threads are represented by the associated connection component. The DSI EXEC thread, however, is not represented by another component.

Thread Context Menus

The Thread context menus give you access to the menus of a related component, if a related component exists for a particular thread.

For example, if you right-click a DIST thread, the context menu displays Connection and Copy. If you select Connection, a context submenu displays, showing you the commands from the connection context menu.

Note: Some thread objects are not related to a component and, therefore, do not give access to a submenu. For example, if you right-click a DSI EXEC thread, the context menu displays only the Copy command.

The following table shows the mapping of thread objects to related components:

Thread	Related Component
DSI DIST REP AGENT	Connection
SQM SQT	Queue
RSI	Route
USER	User

Details List and Other Thread Information

When you select the Threads folder in the left pane, the Details list and tab appear, as well as several other tabs. These other tabs give you access to additional information about each type of thread.

- DIST – displays information about distributor threads.
- DSI – displays information about DSI threads.
- RSI – displays information about RSI threads.
- SQM – displays information about SQM threads.
- SQT – displays information about SQT threads.

See the *Replication Server Administration Guide Volume 1* for more information about Replication Server threads.

Copying Thread Information

Copy information about a thread from any of the thread information lists to the system clipboard.

The columns of data in each list are separated by tabs so that you can paste the columns directly into a columnar format such as a spreadsheet.

1. Right-click the thread in the right pane and select **Copy**.
2. Select Copy.
3. Go to the destination file and paste the information as you would any text.

If you select the DIST thread number 19 for a Replication Server called “myRepServer,” then select Copy from the menu and paste that information into a file, the results look similar to:

```
19 Awaiting Wakeup 102 myRepServer.emb2 102 P Normal 0 1 0 21787  
43856 0 0 0 14
```

New Features in Replication Manager 15.0

Replication Monitoring Services

Replication Server 15.0 introduces a new component called the Replication Monitoring Services (RMS). RMS is the new middle-management monitoring layer that replaces the existing Replication Server Manager and provides monitoring services for large and complex replication environments.

RMS monitors the servers and components in a replication environment, provides the ability to control the flow of data in the replication environment, and sets the configuration parameters. RMS functionality is available through the Replication Manager plug-in to Sybase Central and the command line API. Replication Manager provides commands to create, modify, or delete replication objects, while RMS provides an API to monitor, manage, and configure the replication environment.

RMS is applicable only for a three-tier management solution.

Software Requirements and Compatibilities

Replication Monitoring Services requires:

- JRE version 1.4.2.03
- Sybase Unified Agent Framework (UAF) version 1.5.0.97
- jConnect™ for JDBC™ version 6.0

Installation Notes

You can use InstallShield to install the Replication Monitoring Services component at the same time you install Replication Server.

If you specify a typical or full installation when you install Replication Server 15.0, InstallShield automatically installs the Replication Monitoring Services component along with Sybase Central and Replication Manager.

Three-Tier Management Solution

A three-tier management solution is for large and complex replication environments consisting of ten or more Replication Servers. RMS is applicable only for a three-tier management solution.

The Replication Manager connects to the servers in the environment through RMS. RMS provides the monitoring capabilities for the replication environment.

RMS monitors the status of servers and other components in the replication environment, and the Replication Manager plug-in provides the client interface that displays the status information provided by the RMS.

RMS and RSM Feature Comparison

Compares the features supported in the Replication Server Manager (RSM), a component in the earlier versions, and in the Replication Monitoring Services (RMS), the new component in version 15.0.

Table 34. Difference in Features Supported in RSM and RMS

Feature	Replication Server Manager	Replication Monitoring Services
Allows you to manage, monitor, and configure replication system components	X	X
Monitors the availability of servers and the state of all connections and routes	X	X
Manages a warm standby environment	X	X
Supports multisite availability		X
Supports the Embedded RSSD	X	X
Provides support for SSL and network-based security (DCE/Kerberos) security between servers in a replication environment	X	
Runs on Microsoft Windows platforms as well as all UNIX platforms supported by Replication Server		X
Provides a server-centric view of the replication environment	X	X
Enables administration of a logical group of servers		X
Sets configuration parameters of Replication Servers, Replication Agents, the Adaptive Server Enterprise Replication Agent thread (RepAgent), connections, and routes	X	X
Monitors the latency and the state and performance of a replication path	X	X
Provides commands to create, alter, and delete replication objects	X	
Provides logging and tracing of server commands	X	X
Generates a rollup status for Replication Servers and Adaptive Servers	X	X

Feature	Replication Server Manager	Replication Monitoring Services
Executes user-defined scripts for events in the replication environment	X	X

Supported Replication Environment Servers

You can use the Replication Monitoring Services (RMS) to monitor servers in your replication environment.

The servers supported in RMS are:

- Adaptive Server Enterprise
- SQL Anywhere and IQ
- Replication Agent
- Mirror Replication Agent
- DirectConnect
- Open Server
- Replication Server
- Remote RMS servers

Starting RMS

Start the Sybase Unified Agent Framework (UAF) to start Replication Monitoring Services (RMS). RMS is a Java application built using the Sybase UAF.

1. Navigate to the Replication Server installation directory, %SYBASE% on Windows, or \$SYBASE on UNIX.
2. Set the environment variables by executing **SYBASE.bat** (Windows) or by sourcing **SYBASE.csh** (UNIX).
3. Change to the %SYBASE_UA%\bin directory (Windows) or the \$SYBASE_UA/bin directory (UNIX).
4. In Windows, execute **agent.bat** or add this command to run Unified Agent (UA) in the background. Similarly on UNIX, execute **agent** or add this command to run UA in the background.

Stopping RMS

Stop the Sybase Unified Agent Framework (UAF) to stop Replication Monitoring Services (RMS). RMS is a Java application built using the Sybase UAF.

1. Navigate to the Replication Server installation directory, %SYBASE% on Windows, or \$SYBASE on UNIX.
2. Set the environment variables by executing **SYBASE.bat** (Windows) or by sourcing **SYBASE.csh** (UNIX).
3. Stop RMS.
 - If an Adaptive Server Unified Agent is running, enter one of these commands:
 - `shutdown [-port rmi-port] [-U username]
[-P password]`
 - `shutdown [-port rmi-port] [-user username]
[-password password]`

where:

- *rmi-port* value is 9999.
- *username* and *password* are the values configured for the UA.
- If RMS is the only Unified Agent running, enter **shutdown.bat** (Windows) or **shutdown** (UNIX).

Connecting to RMS in Sybase Central

Add a Replication Monitoring Services (RMS) object in Sybase Central to which you can connect.

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

The RMS server is added to the object tree in Sybase Central.

Replication Environment Management Using RMS

To monitor a replication environment, you must set up a Replication Monitoring Services (RMS) domain. This domain consists of the servers in your environment you want to monitor using the RMS. In a three-tier management solution, the Replication Manager connects to the RMS, which then connects to the various servers in your replication environment.

To set up the RMS domain and monitor servers in this domain, you can either:

- Use the Replication Manager plug-in graphical interface, or
- Use command line API

In a three-tier replication environment, you can perform some monitoring through Replication Manager, and additional monitoring through the RMS API at the command line.

The following topics provide information on how to monitor RMS using the Replication Manager plug-in and the API. For those tasks for which you can use both interfaces, only the Replication Manager plug-in is described in detail, while the name of API is identified. For those tasks that can be performed using only the API, detailed API information is given.

Adding a Server

Add servers to the Replication Monitoring Services (RMS) server folder in Sybase Central for monitoring.

Prerequisites

Before you add a server, have this information available:

- The server's name.
- The type of server you are adding.
- The user name and password used to administer the server. The login must have System Administrator privileges on the server.
- Replication Server only – the user name and password of the RSSD primary user.

Task

The servers that you add for monitoring in a three-tier solution can be from anywhere in your network. This allows you to monitor replication systems that are distributed worldwide.

You can use Replication Manager wizard to add a server to a RMS domain. Depending on the type of server you are adding (Adaptive Server, Replication Server, RepAgent, Open Server, Sybase Replication Agents, DirectConnect), the wizard prompts for different information.

1. Select the RMS.

Replication Monitoring Services

2. Select **File | New | Servers**, or double-click the **Add Servers** icon in the right pane of the main window.
3. Enter the required information in the wizard dialog boxes.

The servers are added to the Sybase Central viewer under the RMS server in the object tree.

Dropping a Server

Drop servers from the Replication Monitoring Services (RMS) domain.

1. Select the server to drop.
2. Either:
 - Click the **Delete** icon in the toolbar, or,
 - Right-click the selected server and select **Delete**.

Sybase Central removes the server from the RMS server folder. However, the server is not actually removed from your replication system, and the server name may still appear in the dialog boxes because there are routes or database connections associated with it.

Viewing Monitored Objects in Sybase Central

View monitored objects in Replication Manager the same as in a two-tier environment.

1. In the object tree, double-click or expand the **RMS** icon to view monitored servers and components, such as connections, routes, queues, and threads.
Under RMS, you can view
2. Select a particular replication object, such as the Routes folder, to view the list of created objects.

You can manage these replication objects using Replication Manager.

Setting Configuration Parameters for Monitored Replication Objects

Set the configuration properties for these replication objects monitored by Replication Monitoring Services (RMS): Replication Server, Replication Agent thread, remote RMS server, database connections and logical connections, and routes.

1. Select the object and choose **File | Properties**.
The property sheet for the object opens.
2. Select the **Parameters** tab in the Properties dialog box.
The Parameters page displays a list of all configuration parameters for the selected server or component.
3. Select the parameter to modify and click **Edit Parameter**.
4. In the Edit dialog, enter the new parameter and click **OK**.
You can modify any other parameter in the list.

5. When you are finished, click **OK**.

Management of a Logical Group of Servers

Replication Monitoring Services (RMS) enables you to define a set of servers as a logical group and monitor the group as a single entity.

You can have many logical groups in your replication environment. The servers in a group can belong to different logical groups. However, all servers in one logical group must be of the same type. For example, you can have one logical group of all Replication Servers and another for all Adaptive Servers. When you issue any command to a group, it affects all the servers contained in this logical group.

You can delete logical groups. When you delete a logical group, only the group is deleted, not the servers within the group.

RMS also returns a rollup status for each group or each server in a group. Rollup status shows the lowest status reported, for example, if any server in a group is not UP, then the group status is reported as SUSPECT.

Suspending or Resuming Components in the Replication Environment

Suspend or resume these monitored components in the replication environment: RepAgent in Adaptive Server, Replication Agent and DSI connection in Replication Server, routes, queues (**resume** only), and replication in a Replication Agent.

In Sybase Central, select the monitored connection or route object from the RMS Server folder.

- To suspend a monitored connection or route, select **Suspend** from the context menu.
- To resume a monitored connection or route, select **Resume** from the context menu.

Shutting Down Monitored Servers

Shut down any monitored Replication Server, Replication Agent, or Mirror Replication Agents using Replication Monitoring Services (RMS).

1. In Sybase Central, select the monitored server or RMS in the object tree and select **Shutdown** from the context menu.
2. To shut down a group of servers, select the logical group folder and select **Shutdown All** from the context menu.

Server Rollup Status

The RMS status shows the lowest status reported, for example, if the status of any server in the list is not UP, then the status for the RMS is reported as SUSPECT.

For example, the status of the Adaptive Server reflects the status of its Replication Agent threads. If any of the monitored Replication Agent threads in the Adaptive Server are not UP, then the state of the Adaptive Server is set to SUSPECT.

For more information about RMS server and component states, see Appendix C, “RMS Server and Component States,” in the *Replication Server Reference Manual*.

Latency and Heartbeat Information Generation

The RMS heartbeat feature uses the stored procedure **rs_ticket** to generate latency information, which is the amount of time it takes for a transaction 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.

At a specified interval, RMS executes **rs_ticket** at the primary database. The generated latency information is stored in a table in the replicate database. You can use RMS to set up the heartbeat process to retrieve the latency information from the replicate database.

For detailed information about **rs_ticket**, see “Adaptive Server Stored Procedures” in the *Replication Server Reference Manual*.

Creating an Event Trigger for a Replication Server

Create event triggers to monitor server and component status changes.

Replication Monitoring Services (RMS) monitors the replication environment. When something happens in your environment, server and component status changes. These changes appear in the event log and you are notified by the event triggers that you may have created. RMS executes a script when the specified event occurs. For example, a user can set up a script to be notified with an e-mail message when a connection suspends. You can create an event trigger for any server or component that the RMS monitors.

1. In the object tree, select the Replication Server.
2. On the right side of the desktop, select the event log pane.
3. Double-click the **Add Server Event Trigger** icon.
4. Select the status change that will trigger the event.
5. As an option, enter a **Wait before executing** value.
This notifies RMS to wait for the event to change before executing the trigger.
6. To execute a trigger at each monitoring interval, rather than only once, select **Execute at Each Interval**.
7. Enter the name of the script for RMS to execute when the event occurs.
8. Click **OK**.
The new event appears in the Event Log pane.

Related API Commands

In addition to the Replication Manager plug-in, you can also use the Replication Monitoring Services (RMS) API to monitor your replication environment.

Task	Related API
Adding a server	addserver – adds a server to monitor.
Dropping a server	dropserver – drops a monitored server.
Viewing monitored objects	N/A
Setting configuration parameters	<ul style="list-style-type: none"> • configure server – returns configuration parameter information for a Replication Server or Replication Agent, or sets the value of a specified configuration parameter. configure server also retrieves and sets RMS-specific parameters. • configure component – returns configuration parameters for a component or sets the value of the configuration parameter specified. • configure RMS – returns the configuration parameter information for the RMS, or sets the value of a specified RMS configuration parameter
Managing a logical group of servers	<ul style="list-style-type: none"> • create group – defines a logical group of servers. • delete group – deletes a logical group that was added using the create group command. • get group – returns a result set that contains either a list of the groups and a rollup status for each group or for each server.
Suspending or resuming components	<ul style="list-style-type: none"> • suspend component – suspends a component in a specified server. • suspend replication – suspends replication in a Replication Agent. • resume component – resumes a component in a specified server. • resume replication – resumes replication in a Replication Agent.
Shutting down monitored servers	shutdown server – shuts down one or more, or group of monitored Replication Servers or RMS.
Viewing rollup status information and various server states	<ul style="list-style-type: none"> • get servers – returns a list of servers that are monitored by the RMS, and the status of the RMS environment. • get status description – retrieves the list of status descriptions for a server or component.

Replication Monitoring Services

Task	Related API
Monitoring heartbeat	<ul style="list-style-type: none">• get heartbeat – retrieves a list of the heartbeat processes that have been defined in the RMS.• get heartbeat ticket – retrieves a set of tickets from the <i>rms_ticket_history</i> table, for the heartbeat process and date and time range specified.• start heartbeat – sets up and starts a heartbeat process from a specified primary connection to a specified replicate connection.• stop heartbeat – stops the heartbeat process between the primary and replicate databases. Optionally truncates the <i>rms_ticket_history</i> table.
Managing event triggers	<ul style="list-style-type: none">• add event trigger – sets up a trigger, such as a process or a script, that is executed by the RMS when a specific event occurs.• drop event trigger – removes a trigger that the RMS is monitoring.• get triggers – displays information about the triggers that are monitored by the RMS.

For detailed information about these APIs, see Chapter 9, “Replication Monitoring Services API,” in the *Replication Server Reference Manual*.

New Features in Replication Server Version 12.6

Replication Server version 12.6 and 12.5 EBFs have introduced several new features.

Multisite Availability

Multisite availability (MSA) extends Replication Server replication capabilities and can make the process of setting up a replication system both faster and easier.

Some of the features that MSA provides include:

- A simple replication methodology that requires only one replication definition for the primary database and only one subscription for each subscribing database.
- A replication filtering strategy that lets you choose whether or not to replicate individual tables, transactions, functions, system stored procedures, and data definition language (DDL).
- Replication of DDL to any replicate database—including non-warm standby databases.
- Replication to multiple replicate sites—for standby as well as nonstandby databases.

You can overlay MSA scenarios onto your existing replication structure. The procedures for implementing MSA are similar to those you already use to replicate to warm standby or replicate databases.

Database Replication

When you use table and function replication, you describe each piece of data that is to be replicated using individual table and function replication definitions and subscriptions. This methodology allows you to transform data and provides fine-grained control over the information being entered in the replicate database. However, you must mark each table or function to be replicated, create a replication definition for each replicated table or function, and create subscriptions for each replication definition at each replicate database.

MSA lets you identify specific database objects: tables, functions, transactions, DDL, and system stored procedures in a single replication definition. You can choose to replicate the entire database; or you can choose to replicate—or not replicate—specific tables, functions, transactions, DDL, and system stored procedures in that database. If you do not need to replicate partial tables, MSA can provide replication while affording the advantages of simple setup and maintenance.

When the Replicate is a Warm Standby Database

In the non-MSA warm standby scenario, changes to the primary database are copied directly to the warm standby database without alteration. This methodology allows replication of

New Features in Replication Server Version 12.6

DDL. To change or qualify the data sent, you must add table and function replication definitions. Each primary database can have one, and only one, standby database.

MSA provides all the features of traditional Sybase warm standby. In addition, MSA:

- Enables replication to multiple standby databases
- Provides the option to replicate or not replicate specific database objects

MSA supports the use of logical connections.

Symmetric Multiprocessors Support

Replication Server 12.6 lets you run Replication Server on either symmetric multiprocessor (SMP) or single-processor platforms. Replication Server multithreaded architecture supports both hardware configurations.

On a single processor platform, Replication Server threads run serially. On a multiprocessor platform, Replication Server threads can run in parallel, thereby improving performance and efficiency.

Replication Server support for multiple processors is based on Open Server support for multiple processors. Both servers use the POSIX thread library on UNIX platforms and the WIN32 thread library on Windows NT platforms. For detailed information about Open Server support for multiple processing machines, see the *Open Server Server-Library/C Reference Manual*.

When Replication Server is in single-processor mode, a server-wide mutual exclusion lock (mutex) enforces serial thread execution. Serial thread execution safeguards global data, server code, and system routines, ensuring that they remain thread-safe.

When Replication Server is in multiprocessor mode, the server-wide mutex is disengaged and individual threads use a combination of thread management techniques to ensure that global data, server code, and system routines remain secure.

Enabling SMP

To enable SMP on a multiprocessor machine, use **configure replication server** with the **smp_enable** option. For example:

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

Commands and Procedures for Monitoring Thread Status

You can verify Replication Server thread status using these commands and procedures:

- **admin who** – provides information on all Replication Server threads
- **admin who_is_up** or **admin who_is_down** – lists Replication Server threads that are running, or not running.

- **sp_help_rep_agent** – provides information on the RepAgent thread and the RepAgent User thread. This is an Adaptive Server stored procedure.

Performance Monitoring

Replication Server provides monitors and counters for monitoring performance. See Chapter 15, “Using Counters to Monitor Performance,” in the *Replication Server Administration Guide Volume 2*.

See the *Replication Server Administration Guide* for more information about increasing Replication Server performance using SMP.

The Embedded RSSD

Replication Server can run either on an Adaptive Server Replication Server System Database (RSSD) or on an Embedded RSSD (ERSSD). ERSSD is designed for users who do not want to manage the Replication Server RSSD in Adaptive Server.

Replication Server is easier to install and manage with ERSSD. If you select embedded RSSD when you install Replication Server, ERSSD is automatically installed, configured, and started in the background. It is self-maintained. Backup procedures are automatic and pre-configured.

Limitations

Currently, you cannot create a route originating from Replication Server with ERSSD. Nor can you migrate between RSSD and ERSSD.

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

Performance Enhancements

Replication Server 12.6 includes new performance enhancements.

See Chapter 16, “Performance Tuning,” in the *Replication Server Administration Guide Volume 1* for detailed information about these enhancements.

Better Management of Empty Transactions

Transactions that contain only a **begin** and a **commit** statement can degrade the performance of warm standby connections. To enhancement performance, these transactions are now deleted from the inbound queue as they are read.

To further enhance performance, Sybase recommends that you also tune your application to eliminate as many of these empty transactions as possible.

Internal Commit Control for Parallel Processing

Replication Server must maintain transaction commit order and resolve commit consistency deadlocks to resolve conflicting updates when using parallel processing.

Replication Server introduces a new method to maintain commit control using the function string **rs_dsi_check_thread_lock**. Replication Server uses **rs_dsi_check_thread_lock** to check whether the current DSI executor thread is blocking another replicate database process. This new method handles commit control within Replication Server thus requiring less network I/O than other methods, and may result in the rollback of only one transaction instead of many.

New database connection parameters used with internal commit control are:

- **dsi_commit_check_locks_intrvl**
- **dsi_commit_check_locks_max**
- **dsi_commit_control**

New Replication Server Configuration Parameters

The new Replication Server configuration parameters fine-tune Replication Server performance.

- **sqt_init_read_delay** – the amount of time an SQT thread sleeps while waiting for an SQM read before checking for new instructions in its command queue.
- **sqt_max_read_delay** – the maximum amount of time an SQT thread sleeps while waiting for an SQM read before checking for new instructions in its command queue.

New Database Configuration Parameters

The new database connection configuration parameters fine-tune Replication Server performance.

- **dsi_commit_check_locks_intrvl** – specifies the number of milliseconds (ms) the DSI executor thread waits between executions of the *rs_dsi_check_thread_lock* function string. Used with parallel DSI.
- **dsi_commit_check_locks_max** – specifies the maximum number of times a DSI executor thread checks whether it is blocking other transactions in the replicate database before rolling back its transaction and retrying it. Used with parallel DSI.
- **dsi_commit_control** – specifies whether commit control processing is handled internally by Replication Server using internal tables (on) or externally using the *rs_threads* system table (off). Used with parallel DSI.

Changed Database Configuration Parameters

The database connection parameters **dsi_serialization_method** and **dsi_partitioning_rule** have changed.

dsi_serialization_method changes

dsi_serialization_method specifies how and when parallel DSI threads can start. It includes these new or changed options:

- **no_wait** – specifies that a transaction can start as soon as it is ready—without regard to the state of other transactions.
- **wait_for_start** – specifies that a transaction can start as soon as the transaction scheduled to commit immediately before it has started. Replaces the **none** option. When used with the **origin** partitioning parameter, this option replaces the **single_transaction_per_origin** serialization method.
- **isolation_level_3** – is the same as **wait_for_start**, except that DSIs will specify isolation level 3 when connecting to the replicate database.
- **wait_for_commit** – specifies that a transaction can start only when the previous transaction is ready to commit.
- **none** – maintained for backward compatibility. Replaced by **wait_for_start**.
- **single_transaction_per_origin** – maintained for backward compatibility. Replaced by **dsi_serialization_method** set to **wait_for_start** and **dsi_partitioning_rule** set to **origin**.

dsi_partitioning_rule changes

dsi_partitioning_rule specifies the partitioning rules the DSI uses to partition transactions among available parallel DSI threads. It includes these new options:

- **origin** – specifies that transactions with the same origin must be serialized when applied to the replicate database.
- **origin_sessid** – specifies that transactions with the same origin *and* the same process ID (SPID in Adaptive Server) must be serialized when applied to the replicate database. The LTL version must be 600 or later.

Sybase recommends that you try setting **dsi_partitioning_rule** to **origin_sessid, time** as this setting may provide the most efficient partitioning.

date and time Datatypes

Replication Server 12.6 introduces the `date` and `time` datatypes. These datatypes extend the existing `datetime` and `smalldatetime` datatypes, providing `date` and `time` columns to replicate and standby databases. Both are fixed-width 4-byte datatypes that support **rs_subcmp**, and mixed-version environments.

Replication Server version 12.6 with `date` and `time` datatype support is backward-compatible with earlier versions of Adaptive Server. However, earlier versions of Adaptive

Server do not recognize `date` and `time`, and thus can send only `datetime` and `smalldatetime` data.

The new columns generated by `date` and `time` datatypes allow you to replicate date and time data to both standby and replicate databases. These columns can be part of the primary key in a replication definition, and are searchable columns in a replication definition. You can use `date` and `time` columns in the **where** clause of **define subscription**, **create subscription**, or **create article**. In the same way, the `date` and `time` columns are searchable parameters in a function replication definition, again used in the **where** clause of **define subscription**, **create subscription**, or **create article**.

Table 35. Range and Storage Needs for date and time Datatypes

Datatype	Range	Storage Needed
<code>date</code>	January 1, 0001 to December 31, 9999	4
<code>time</code>	12:00:00 AM to 11:59:59.999 PM	4
<code>smalldatetime</code>	January 1, 1900 to June 6, 2079; 12:00:00AM to 11:59:59.999 PM	4
<code>datetime</code>	January 1, 1753 to December 31, 9999; 12:00:00AM to 11:59:59.999 PM	8

Support for Sending Encrypted Passwords

Replication Server 12.6 supports the **-X** option in **isql** that sends encrypted passwords through the network when making a client connection.

To ensure that all Replication Server client connections—except the first connection to the RSSD—send encrypted passwords, set the Replication Server configuration parameter **send_enc_password** to **on**. For example, enter:

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

To ensure that all Replication Server client connections, *including* the first connection to the RSSD, send encrypted passwords, set the configuration parameter **RS_enc_pw** to **on** in the *rs_name.cfg* file using a text editor.

If **RS_enc_pw** is **on**, all Replication Server connections to the RSSD send encrypted passwords, even if **send_enc_password** is **off**.

New Bulk Materialization Method

Replication Server 12.6 supports a new bulk materialization method for copying or moving a database from a source Adaptive Server to a destination Adaptive Server without shutting down the source Adaptive Server.

The Adaptive Server **quiesce database ... to *manifest_file*** and **mount** commands let you quiesce the server and copy or move the database.

To use this bulk materialization method, both the source and destination database servers must be Adaptive Server version 12.5.1 or later.

You can use **quiesce database ... to *manifest_file*** to generate all the data storage information and then use **mount** to mount the data to a new database—see Chapter 10, “Managing Subscriptions,” in the *Replication Server Administration Guide*. You can use **quiesce database ... to *manifest_file*** and **mount** when you add a warm standby database—see “Managing Warm Standby,” in the *Replication Server Administration Guide Volume 1*.

For information about **mount** and **unmount**, see “Database Mount and Unmount,” in the *Adaptive Server Enterprise System Administration Guide*.

Chinese Character Set (GB18030) Support

Replication Server supports all character sets supported by Adaptive Server Enterprise. Accordingly, Replication Server 12.6 supports the Chinese character set (GB18030).

Obtaining Help and Additional Information

Use the Sybase Getting Started CD, Product Manuals site (SyBooks™ Online), 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 Manuals 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 Manuals 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. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.
4. Specify a time frame and click **Go**. You see a list of EBF/Maintenance releases.

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 **Edit Roles** to add the “Technical Support Contact” role to your MySybase profile.

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

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

64-bit support 90–92

A

Adaptive Server
 resynchronizing replicate database 45
 support, in Replication Server 15.5 86
 support, in Replication Server 15.6 63

Adaptive Servers
 communication through Replication Manager 180
 shared-disk cluster support 129

admin config 150
 admin config command 96
 admin who
 enhancement 83

Advanced Services Option 72
 alter connection command 96

B

Background Processes dialog box 181
 bigdatetime and bigtime datatype support in Replication Manager 93
 bigdatetime, replication support 86
 bigtime, replication support 86
 block size
 changing 40

bulk copy-in support
 connection parameters 96
 connection parameters, checking value of 96
 connection parameters, setting value of 96
 Data Server Interface (DSI), implementation in 95
 multi-statement transactions, support for 96

bulk insert
 See bulk copy-in support

bulk materialization 209

C

cascading connection, in Replication Server gateway 107
 changing replication definitions, enhancements to process 77

character set
 Chinese (GB18030) 209

Chinese character set support 209

commands
 admin config 96, 102
 alter connection 96, 99, 101
 alter replication definition 103
 configure replication server 96, 99
 connect 107
 create connection 101
 create replication definition 102
 create route 172
 disconnect 109
 show connection 108
 show server 108
 sysadmin dump_tran 133
 sysadmin issue_ticket 84
 sysadmin sqm_unzap_tran 133
 sysadmin sqm_zap_tran 132

commit control
 internal 206

compilation and bulk apply in RTL 20

configuration for replicate Sybase IQ 26

configuration overview 45

configuration parameters
 dist_sqt_max_cache_size 91
 dsi_bulk_copy 95, 96
 dsi_bulk_threshold 96
 dsi_non_blocking_commit 99
 dsi_row_count_validation 60
 dsi_sqt_max_cache_size 91
 dynamic_sql 153
 dynamic_sql_cache_management 153
 dynamic_sql_cache_size 153
 exec_cmds_per_timeslice 85
 init_sqm_write_delay 85
 init_sqm_write_max_delay 85
 memory_limit 85, 92
 setting with Replication Manager 198
 smp_enable 85
 sqt_max_cache_size 85, 91
 stats_reset_rssd 152
 sts_full_cache 85

configure replication server command 96, 99

Index

- configuring database resynchronization 45
 - applying dump to a database to be resynchronized 48
 - instructing Replication Server to skip transactions 46
 - monitoring DSI thread information 48
 - obtaining a dump of the database 48
 - sending resync database marker to Replication Server 46
 - sending the dump database marker to Replication Server 48
- connection profiles 123
 - Sybase IQ 26
- connection profiles for Sybase IQ 26
- connection to Sybase IQ
 - creating 26
- connectivity for replicate Sybase IQ 24
- conventions
 - style 1
 - syntax 1
- counter statistics
 - optionally keeping in RSSD 152
- counters 135
- creating
 - connection to Sybase IQ 26

D

- Data Server Interface 95, 96
- data transfer, incremental 88
- database generation numbers, resetting 84
- database permissions for replicate Sybase IQ 25
- database resynchronization 79
- database resynchronization scenarios 49
 - resynchronizing both the primary and replicate databases from the same dump 53
 - resynchronizing if there no resync database marker support 52
 - resynchronizing replicate databases directly from a primary database 49
 - resynchronizing the active and standby databases in a warm standby application 54
 - resynchronizing using a third-party dump utility 51
- database support, real-time loading 19, 69
- datatype
 - date, time 207
- datatypes
 - bigdatetime 86

- bigtime 86
- image 64
- Java 64
- opaque 138, 146
- timestamp 147
- date, datatype 207
- default parameter values, changes 85
- deferred name resolution, replication support 87
- delaying replication 79
- deleting exceptions 56
 - dates 57
 - range of transaction IDs 56
 - user or destination site 59
- dialog boxes
 - Background Processes 181
- direct I/O file access 143
- DIST status recording 139
- dist_sqt_max_cache_size 91
- distributor thread read efficiency, enhanced 74
- DSI 95
 - DSI efficiency enhanced 73
- dsi_bulk_copy connection parameter 95, 96
 - checking value of 96
 - setting value of 96
 - See also bulk copy-in support
- dsi_bulk_threshold connection parameter 96
 - checking value of 96
 - setting value of 96
 - See also bulk copy-in support
- dsi_bulk_threshold in RTL 28
- dsi_command_convert in RTL 28
- dsi_compile_enable in RTL 27
- dsi_compile_retry_threshold configuration parameter, to enable enhanced retry mechanism 38
- dsi_compile_retry_threshold in RTL 28
- dsi_max_cmds in RTL 28
- dsi_max_cmds_in_batch configuration parameter 70
- dsi_max_xacts_in_group configuration parameter 70
- dsi_quoted_identifier 101
- dsi_row_count_validation configuration parameter 60
- dsi_serialization_method configuration method 70
- dsi_sqt_max_cache 91
- dump database 48
- dump database marker, sending 48
- dump of database, applying 48

- dump of database, obtaining a 48
- dynamic parameters
 - configuring 149
- dynamic SQL
 - configuration parameters setup 153
 - enhancement 127, 145
- dynamic SQL enhancements 74
- dynamic SQL parameters 149
- dynamic SQL, extending
 - replicate_minimal_columns parameter to connections 75
- dynamic SQL, optimizing statement execution 74
- dynamic SQL, using with replicate minimal columns clause 75

E

- Embedded Replication Server System Database
 - 172, 205
 - create route 172
 - limitations 205
 - routing 172
- empty transactions 205
- encrypted passwords
 - extended support 134
 - sending 208
- enhanced distributor thread read efficiency 74
- enhanced DSI efficiency 73
- enhanced memory allocation 74
- enhanced RepAgent executor thread efficiency 73
- enhanced text update 140
- enhancements
 - admin who 83
 - configuration and tuning 149
 - dump transaction 138
 - dynamic SQL 127, 145
 - error handling 80
 - for Replication Server performance 95, 152
 - function replication 128, 145
 - locales directory 133
 - log_first_tran 132
 - monitor and counter 129, 150
 - release area 133
 - Replication Manager plug-in 163
 - resume connection 132
 - rs_subcmp 154
 - SQM performance 141
 - stable queue management 131
 - sysadmin dump_queue 131
 - sysadmin sqt_dump_queue 131

- error class, for Sybase IQ 26
- error handling
 - enhancement 80
- ERSSD
 - See Embedded Replication Server System Database
- Event Log pane 181
- example for RTL replication 29
- exceptions log
 - deleting transactions by originating user or originating or destination site 59
 - deleting transactions by range of transaction dates 57
 - deleting transactions by range of transaction IDs 56
- exceptions, deleting 56
- exec_cmds_per_timeslice 85

F

- failback system, by delaying replication 79
- fstr_cache_size 92
- function replication
 - enhancement 128
- function string
 - rs_dsi_check_thread_lock 70, 206
 - rs_non_blocking_commit 100
 - rs_non_blocking_commit_flush 100
 - rs_set_quoted_identifier 103
- function string efficiency improvements, extending
 - none parameter of function string commands 75
- function-string class for Sybase IQ 26

H

- hash algorithm
 - command line parameters 155
 - configuration file parameters 155
- heterogeneous parallel DSI 69
- High Volume Adaptive Replication 72
- HVAR 72
- HVAR, retry mechanism enhanced 38

I

- image datatype, bulk copy-in replication support
 - 64
- IMDB 63, 88, 89

Index

- in-memory databases 63, 88
- increasing queue block size 39, 74
- incremental data transfer, support for 88
- init_sqm_write_delay 85
- init_sqm_write_max_delay 85
- injecting rs_ticket markers 84
- interfaces file 24
- interfaces file, creating for replication to Sybase IQ 30
- internal commit control 206
- intrusions and impacts, replication into Sybase IQ 23
- intrusions into Sybase IQ, from temporary worktables 24
- J**
- Java datatype, bulk copy-in replication support 64
- Java JRE
 - compatibility with Replication Monitoring Services 193
- jConnect for JDBC
 - compatibility with Replication Monitoring Services 193
- L**
- license
 - obtaining 18
- licenses
 - types of 17
- licenses, types of 67
- licensing 17
- limitations
 - dynamic SQL 127
 - function replication 129
 - LOB datatypes 136
 - opaque datatype 138
 - partial update 137
- LOB datatypes
 - partial update 137
 - support 146
- locales directory
 - changes 133
- locking schema, RSSD 86
- log
 - event 181
 - exceptions 56, 57, 59
- M**
- maintenance user
 - granting authority 25
- manual data reconciliation 159
- master database
 - supported DDL commands 154
- md_source__memory_pool 92
- memory allocation, enhanced 74
- memory_limit 85, 92
- migrating from Sybase IQ replication staging solution to RTL 36
- minimal DML logging 89
- mixed-version environment restrictions, with version 15.5 and later 90
- monitoring DSI, for resynchronizing database 48
- monitoring of status 180
- mount command 209
- multiprocessors
 - enabling 204
 - monitoring 204
- multisite availability (MSA) 203
- multithreaded architecture 204
- N**
- net-change database in RTL, displaying 35
- new features
 - Replication Manager 15.0 179
 - Replication Manager 15.0.1 161
 - Replication Manager 15.1 145
 - Replication Manager 15.5 93
 - Replication Server 15.2 95
 - Replication Server 15.5 67
 - Replication Server 15.6 17
 - Replication Server 15.6 ESD #1 5
- no resync database marker support
 - resynchronizing databases 52
- non-ASE error class support
 - altering error classes 122
 - creating error classes 121
 - default non-ASE error classes 121
- non-ASE replicate support
 - connection profiles 123
 - listing connection profiles 125
 - simplified installation, configuration 123
 - using connection profiles 123
- non-blocking commit
 - ASE delayed commit feature 98
 - configuring 99
 - dsi_non_blocking_commit 99
 - non-ASE databases, support for 101
 - Oracle, support for 101
 - rs_non_blocking_commit 100

- rs_non_blocking_commit_flush 100
- none parameter, extending scope in alter function string, create function string 75

O

- opaque datatypes 146
- operating system support, in Replication Server 15.5 90
- Oracle data server
 - rs_lastcommit table 23
- Oracle to Sybase IQ replication 19
- Oracle, replicating from 19
- Oracle, trigger execution 71

P

- parallel DSI parameter
 - dsi_max_cmds_in_batch 70
 - dsi_max_xacts_in_group 70
 - dsi_serialization_method 70
- parallel processing 204
 - empty transactions 205
 - internal commit control 206
- parameters, changes to default values 85
- password encryption
 - extended support 134
- performance enhancement
 - High Volume Adaptive Replication 72
- performance enhancements
 - Advanced Services Option 72
 - Enhanced DSI efficiency 73
 - High Volume Adaptive Replication 72
- performance enhancements, in Replication Server 15.5 72
- performance enhancements, in Replication Server 15.6 38
- permissions, for replicate Sybase IQ 25
- platform support, real-time loading 19, 69
- POSIX thread library 204
- product editions, types of 17, 67
- profiles
 - connection 26

Q

- queue block size
 - changing 40
 - example, simple replication system 41

- example, with intermediate route 43
- recommendations 39
- restrictions 40
- queue block size, increasing 39, 74
- queue_dump_buffer_size 92
- quiesce database command 209
- quoted identifiers
 - alter replication definition, changes to 103
 - create replication definition, changes to 102
 - dsi_quoted_identifier 101
 - embedded double quote characters 101
 - quoted parameter 102
 - rs_set_quoted_identifier 103

R

- RCL command
 - admin stats 150
 - admin stats, status 151
- real-time loading
 - database support 19, 69
 - platform support 19, 69
- real-time loading (RTL) replication to Sybase IQ 19, 68
- Real-Time Loading Edition 19, 68
- reference implementation 81
- referential constraints in RTL 33
- relaxed-durability databases 63, 88
- RepAgent executor thread efficiency, enhanced 73
- replicate databases
 - Sybase IQ 23
- replicate minimal columns clause, using with dynamic SQL 75
- replicate_minimal_columns parameter, extending to connections using dynamic SQL 75
- replicating
 - data, large batch of 95
 - master database 153
- replication definition change request process enhancements 77
- Replication features
 - supported by Replication Manager 183
- Replication Manager
 - new features 15.0 179
 - new features 15.0.1 161
 - new features 15.1 145
 - new features 15.5 93
 - support of Replication Server/Replication features 183
 - using to set configuration parameters 198

Index

- Replication Monitoring Services
 - adding servers 197
 - compatibilities 193
 - connecting to 196
 - dropping servers 198
 - monitoring replication environment 197
 - software requirements 193
 - viewing monitored objects 198
- Replication Server
 - communication through Replication Manager 180
 - dynamic configuration 149
 - interfaces file 24
 - new features 15.2 95
 - new features 15.5 67
 - new features 15.6 17
 - new features 15.6 ESD #1 5
 - Sybase IQ replicate database 24
- Replication Server and Sybase IQ InfoPrimer data flow 5, 6
- Replication Server features
 - supported by Replication Manager 183
- Replication Server gateway
 - cascading connection 107
 - connection dropping 109
 - connections, tracking 108
 - enable, to 107
 - limitations 107
 - product version requirements 108, 109
- Replication Server System Database (RSSD) 205
- replication threshold setting, in SQL statement replication 88
- replication, scheduling tasks 78
- reserved words, new 86
- resetting database generation numbers 84
- resume connection, with skip to resync marker 46
- resync marker, sending 46
- resync marker, with a purge instruction 47
- resync marker, with init command 47
- resync marker, without any option 47
- resynchronizing Adaptive Server databases
 - Adaptive Server and RepAgent versions supported 45
 - introduction 45
- resynchronizing database 45
 - applying dump of database 48
 - configuration 45
 - monitoring DSI 48
 - obtaining database dump 48
 - resuming connection with skip to resync parameter 46
 - resync marker, sending 46
 - scenarios 49
 - scenarios, no resync database marker support 52
 - scenarios, warm standby 54
 - sending dump database marker 48
 - skip to resync parameter 46
 - skipping transactions 46
- resynchronizing databases 79
- retry mechanism, enhanced for HVAR and RTL 38
- route upgrade 62
- row count validation
 - disabling 60
 - displaying table names 61
 - enhancements 60, 61
- row count validation enhancements 80
- row count verification, in SQL statement replication 118
- rs_autoc_ignore system function 15
- rs_autoc_off system function 14
- rs_autoc_on system function 13
- rs_delexception
 - delete transactions by range of IDs and dates 56
- rs_delexception_date stored procedure 57
- rs_delexception_id stored procedure 56
- rs_delexception_range stored procedure 59
- rs_dsi_check_thread_lock function string 206
- rs_lastcommit table
 - in Oracle database 23
- rs_status system table 12
- rs_subcmp
 - parameters for manual reconciliation 159
 - schema subtypes 157
 - schema types 156
 - string options 155
- rs_ticket
 - version 2 134
- rs_ticket markers, injecting 84
- RSSD locking schema, changes 86
- RTL 19, 68
 - admin config command 34
 - advantages 19
 - backward compatibility 36
 - compilation and bulk apply 20
 - compilation examples 20
 - compilation rules 20

- configuring for replication to Sybase IQ 28
 - database and platform support 19
 - displaying database-level configuration
 - parameters 34
 - displaying information 34
 - displaying net-change database 35
 - displaying table references 35
 - displaying table-level configuration
 - parameters 34
 - dsi_bulk_threshold 28
 - dsi_command_convert 28
 - dsi_compile_enable 27
 - dsi_compile_retry_threshold 28
 - dsi_max_cmds 28
 - enabling for replication to Sybase IQ 27
 - limitations 21
 - migrating from the staging solution 36
 - mixed-version support 36
 - noncompilable commands, tables 22
 - referential constraints 22, 33
 - replication scenario 29
 - rs_helprep stored procedure 35
 - system table support 29
- RTL, retry mechanism enhanced 38
- ## S
- sampling data
 - optionally keeping in RSSD 152
 - scenario for RTL replication 29
 - scenarios, database resynchronization 49
 - scenarios, database resynchronization, no resync
 - database marker support 52
 - scenarios, database resynchronization, warm
 - standby 54
 - scheduling replication tasks 78
 - schema comparison 156
 - command line parameters 157
 - configuration file parameters 158
 - seamless upgrade 62
 - segment preallocation 142
 - select command
 - variables and 109
 - send_enc_password configuration parameter 208
 - serialization method
 - wait_after_commit 69
 - setting configuration parameters
 - using Replication Manager 198
 - skip to resync marker, sending to Replication Server
 - from RepAgent 46
 - skip to resync parameter 46
 - smp_enable 85
 - software compatibilities of Replication Monitoring
 - Services 193
 - software requirements for Replication Monitoring
 - Services 193
 - SQL statement replication
 - autocorrection 117
 - configuring warm standby 119, 120
 - database level 113
 - database replication definition 115
 - replicate SQLDML clause 115
 - restrictions 117
 - row count verification 118
 - RSSD modifications 120
 - session level 114
 - set repmode 114
 - setting replication threshold 88
 - sp_setrepdbmode 113
 - sp_setrepdefmode 114
 - table level 114
 - table replication definition 116
 - WS_SQLDML_REPLICATION parameter
 - 119, 120
 - sqt_max_cache_size 85, 91, 92
 - sre_reserve 92
 - stable queue caching 141
 - stats_reset_rssd 152
 - status monitoring using RMS 180
 - sts_cachesize 92
 - sts_full_cache 85
 - sub-capacity licensing 17
 - support
 - Adaptive Server integer identity 140
 - direct I/O 143
 - dynamic configuration 161
 - for Adaptive Server shared-disk cluster 129
 - for heterogeneous data servers 161
 - for non-Sybase data servers 162
 - for Replication Agents and Mirror Replication
 - Agents 162
 - LOB datatypes 136, 146
 - opaque datatypes 146
 - timestamp datatypes 137, 147
 - See also bulk copy-in support
 - Sybase IQ
 - configuring RTL 28
 - connections profiles 26
 - enabling RTL 27

Index

- error class and function-string class 26
- intrusions, system tables 23
- intrusions, temporary worktables 24
- replicate database configuration 26
- replicate database connectivity 24
- replicate database permissions 25
- replication intrusions and impacts 23
- RTL compilation and bulk apply 20
- staging solution, migrating from 36
- Sybase IQ, replicating to 19, 68
- symmetric multiprocessors (SMP) 204
- system tables
 - rs_status 12

T

- tables
 - rs_lastcommit, in Oracle database 23
- time, datatype 207
- timestamp datatypes 147
- transactions
 - empty 205
- trigger execution, for Oracle 71

U

- upgrade routes 62
- upgrading Replication Server seamlessly 62

- usability and process enhancements, in Replication Server 15.5 76
- usability and process enhancements, in Replication Server 15.6 45
- user interface features, Replication Manager 179

V

- version support
 - resynchronizing Adaptive Server 45
- visual monitoring of status 180

W

- warm standby
 - adding active database connection 188
 - adding logical connection 188
 - adding standby database connection 189
 - heterogeneous 71
 - Oracle 71
 - resynchronizing databases 54
- WIN32 thread library 204
- wizards
 - Add Active Database 188
 - Add Logical Connection 188
 - Add Standby Database 189