SYBASE®
An SAP® Company

Reference Manual

Replication Agent™ for Oracle 15.7.1 ESD #2

Linux, Microsoft Windows, and UNIX

DOCUMENT ID: DC01847-01-1571-01
LAST REVISED: October 2012

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

Contents

Conventions

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

Style conventions

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

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

Run:

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

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

Syntax conventions

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

Case-sensitivity

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

Terminology

Replication Agent™ is a generic term used to describe Replication Agent for Oracle for Linux, Unix and Windows.

Command Reference

Learn about Replication Agent commands including targets, syntax, options, examples, and command usage.

Replication Agent Commands Table

This table lists the commands that Replication Agent supports.

Table 1. Replication Agent Commands

Command Name	Description
log_system_name on page 8	Returns the path to the Replication Agent system log file.
lr_dump_marker on page 9	Returns or sets the system change number (SCN), indicating the point at which the primary database was last dumped.
pdb_capabilities on page 9	Returns a list of Replication Agent capabilities.
pdb_date on page 10	Returns the current date and time from the primary data server.
pdb_execute_sql on page 10	Executes the specified SQL statement in the current database.
pdb_gen_id on page 11	Returns the current value of the database generation ID; updates the value of the database generation ID.
pdb_get_columns on page 12	Returns a list of all the columns in the specified table.
pdb_get_databases on page 14	Returns a list of all the databases in the primary data server.
pdb_get_primary_keys on page 15	Returns a list of all the columns that make up the primary keys in the specified table.
pdb_get_procedure_parms on page 16	Returns a list of the properties for the specified procedure.
pdb_get_procedures on page 17	Returns a list of all the procedures in the specified database.

Command Name	Description
pdb_get_sql_database on page 18	Returns the name of the database specified for SQL statement execution.
pdb_get_tables on page 19	Returns a list of all the tables in the specified database.
pdb_ownerfilter on page 20	Returns a list of owners whose objects will be filtered for initialization; adds or removes owners to or from the list.
pdb_send_osuser_list on page 22	Returns and modifies a list of database users whose primary database user names will be replaced with the corresponding operating system user name in LTL sent to Replication Server®.
pdb_set_sql_database on page 24	Specifies the database to be used for SQL statement execution.
pdb_setrepcol on page 25	Returns replication marking status; enables or disables replication for all marked columns or a specified column.
pdb_setrepddl on page 29	Returns DDL replication status; enables or disables replication for DDL statements.
pdb_setrepproc on page 37	Changes and reports stored procedure marking status.
pdb_setrepseq on page 45	Changes and reports sequence replication marking status.
pdb_setreptable on page 47	Changes and reports table replication marking status.
pdb_skip_op on page 57	Returns, adds, or removes record identifiers from a list of records to skip in processing.
pdb_thread_filter on page 59	Filters all activity on an Oracle instance redo log thread or threads during replication, and displays a list of threads being filtered.
pdb_truncate_xlog on page 60	Truncates the Replication Agent primary database transaction log.
pdb_version on page 61	Returns the type and version of the primary data server.
pdb_xlog on page 61	Returns names of transaction log objects; creates Replication Agent system objects in the primary database; removes Replication Agent system objects from the primary database.

Command Name	Description
quiesce on page 65	Stops current Log Reader activity after all data remaining in the transaction log and Replication Agent internal queues is processed and puts Replication Agent in Admin state.
ra_admin on page 66	Returns the names of Replication Agent system objects; creates Replication Agent system objects in the primary database; or removes Replication Agent system objects from the primary database.
ra_config on page 70	Returns help information for configuration parameters; sets the value of a configuration parameter.
ra_date on page 73	Returns the current date and time from the Replication Agent server.
ra_downgrade on page 73	Prepares Replication Agent to downgrade to an earlier version.
ra_downgrade_accept on page 74	Completes the version downgrade process initiated by the instance from which Replication Agent is being downgraded.
ra_downgrade_prepare on page 75	Prepares Replication Agent for a version downgrade.
ra_dump on page 76	Records a dump marker in the primary database transaction log.
ra_dumptran on page 77	Returns information for use in troubleshooting a specific database transaction.
ra_finalize_upgrade on page 81	Finalizes the upgrade of instances from a previous version and prevents downgrade to the previous version.
ra_help on page 82	Returns help information for Replication Agent commands.
ra_helparchive on page 83	Displays a list of metadata for all managed archive logs, for a specific redo log thread, or for archive logs for a specific redo log thread.
ra_helparticle on page 83	Returns information about articles from the RASD.
ra_helpdb on page 85	Returns information about the primary database from the RASD.
ra_helpdevice on page 85	Returns information about primary database log devices from the RASD.

Command Name	Description
ra_helpfield on page 87	Returns information about fields (columns in tables, or input parameters in stored procedures) from the RASD.
ra_helplocator on page 89	Returns LTM locator field values.
ra_helpop on page 90	Returns information for use in troubleshooting a specific database transaction log operation.
ra_helptran on page 92	Returns a list of all open transactions.
ra_helpuser on page 93	Returns information about primary database users from the RASD.
ra_license on page 94	Returns license information for Replication Agent and its licensed features.
ra_locator on page 95	Returns and changes the current value of the LTM Locator stored by Replication Agent.
ra_maintid on page 97	Returns the maintenance user for the Replication Agent connection.
ra_marker on page 98	Records a marker in the primary database transaction log.
ra_migrate on page 99	Performs any necessary migration and downgrade tasks between releases of Replication Agent.
ra_purge_first_open on page 100	Removes the first open transaction from the list of open transactions.
ra_regenerate_keys on page 100	Regenerates the value of the **instance_rand** configuration property and the **instance_password_key** entry in the RASD encryption keys table
ra_set_autocorrection on page 101	Enables or disables autocorrection for marked tables.
ra_set_login on page 103	Sets the Replication Agent admin user login and password.
ra_statistics on page 104	Returns statistics for either a specified Replication Agent component or all components, and resets statistics for all components.
ra_statrack on page 111	Starts and stops the statistics tracking thread.
ra_statrack_list on page 111	Adds or removes a group of statistics from the tracking list, replaces the tracking list, and displays a list of statistics currently being tracked.

Command Name	Description
ra_status on page 112	Returns the current Replication Agent state.
ra_truncatearticles on page 113	Truncates older versions of primary database articles in the system data repository in the RASD.
ra_truncateddlfilters on page 114	Truncates old lists of DDL commands that are filtered in the RASD.
ra_truncateusers on page 114	Truncates older versions of primary database users in the system data repository in the RASD.
ra_updatedevices on page 115	Updates the log device repository in the RASD.
ra_updateusers on page 117	Reloads user information from the primary database to the RASD.
ra_version on page 117	Returns the Replication Agent version.
ra_version_all on page 118	Returns Replication Agent, primary data server, Replication Server, and communications driver versions.
rasd_backup on page 119	Backs up the Replication Agent System Database (RASD).
rasd_helpbackup on page 119	Displays a list of RASD backups.
rasd_removebackup on page 120	Removes RASD backups.
rasd_restore on page 121	Restores the Replication Agent System Database (RASD).
rasd_trunc_schedule on page 122	Returns a list of the repository truncation weekly schedule; also adds or removes a specific schedule.
resume on page 124	Starts replication for the current active log and puts Replication Agent in Replicating state.
rs_create_repdef on page 126	Creates a replication definition at Replication Server for a marked table and procedure, or for all marked tables and procedures.
rs_drop_repdef on page 128	A replication definition at the configured Replication Server for a table and procedure is dropped.
rs_ticket on page 129	Supports Replication Server **rs_ticket** processing by placing an **rs_ticket** marker in the primary database transaction log.
shutdown on page 130	Shuts down Replication Agent.

Command Name	Description
suspend on page 131	Immediately stops all Log Reader activity, drops connections, and puts Replication Agent in Admin state.
test_connection on page 132	Tests Replication Agent connectivity.
trace on page 135	Returns current trace flag settings; changes a specified trace flag.

log_system_name

Returns the full path of the Replication Agent instance log file.

Syntax

```
log_system_name
```

Usage

- When you create a Replication Agent instance, a log directory is created automatically as part of the instance directory structure. The default value of the **log_directory** parameter points to that directory.
- The default path of the Replication Agent log directory on Microsoft Windows is:

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

  The default path of the Replication Agent log directory on Linux and UNIX is:

  ```
  $SYBASE/RAX-15_5/inst_name/log/
  ```

  where:
  - *%SYBASE%* or *$SYBASE* is the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.
- If you specify a valid directory path as the value of the **log_directory** parameter, the Replication Agent instance places its system log file in the directory you specify.

  If you change the value of the **log_directory** parameter with the **ra_config** command, the new value is recorded in the configuration file immediately, but you must shut down and restart the Replication Agent instance to make the new value take effect.

  See the log_directory parameter for more information.
- The **log_system_name** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *trace* on page 135
- *ra_config* on page 70

lr_dump_marker

Returns or sets the system change number (SCN), indicating the last committed transaction in the primary database dump. This value is sent to Replication Server when Replication Agent encounters a log record with an SCN greater than or equal to this value.

Syntax

```
lr_dump_marker [scn]
```

Parameters

- **scn –** The SCN indicating the point at which the primary database was last dumped. This is the **dump database** marker and denotes the oldest committed transaction in the dump.

Usage

- **lr_dump_marker** invoked with no option returns the SCN marking the point at which the primary database was last dumped. If no SCN has yet been specified, **lr_dump_marker** returns 0.
- To set an SCN dump point, invoke **lr_dump_marker** with a valid SCN marking the point at which the primary database was last dumped. If you made the dump with the Oracle Recovery Manager (RMAN) utility, you can obtain this SCN by using the RMAN **list backup** command.
- The SCN set with the **lr_dump_marker** command takes effect when Replication Agent is in the Replicating (Resynchronization) state.

pdb_capabilities

Returns a list of Replication Agent capabilities, which is used by the replication management tools.

Syntax

```
pdb_capabilities
```

Usage

- When **pdb_capabilities** is invoked, it returns a list of the capabilities of the Replication Agent instance.
- The purpose of the **pdb_capabilities** command is to support the replication management tools.
- The **pdb_capabilities** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

pdb_date

Returns the current date and time from the primary data server.

Syntax

```
pdb_date
```

Usage

- When **pdb_date** is invoked, it returns the current date and time from the primary data server in the form of a Sybase `datetime` datatype, as follows:

  ```
  Current PDB Date
  ------------------------------
  Jan 11 2010 12:09:47.310
  (1 row affected)
  ```

- The **pdb_date** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_date* on page 73

pdb_execute_sql

Executes a SQL statement in the current database at the primary data server.

Syntax

```
pdb_execute_sql statement
```

Parameters

- **statement –** A string in the form of a SQL statement enclosed in double quotes.

Usage

- The Replication Agent instance executes the specified SQL statement against the "current" database.

  The current database is either:

  - The default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter, or
  - The database specified in the **pdb_set_sql_database** command (to which the Replication Agent instance is currently connected).

- To set or change the current database, use the **pdb_set_sql_database** command.
- To find the name of the current database, use the **pdb_get_sql_database** command.

Note: If the **pdb_set_sql_database** command has not been invoked to set or change the current database, the **pdb_get_sql_database** command returns the name of the default current database.

- The SQL statement specified in the **pdb_execute_sql** command must be a single SQL command enclosed in double quotes. For example:

  ```
  pdb_execute_sql "select * from Authors"
  ```

  The string is passed directly to the database for execution. No command to terminate is required and no syntax or other validation is performed.
- Any results returned from execution of the SQL statement are passed to the Replication Agent administrative client, by way of the Replication Agent administration port.
- The **pdb_execute_sql** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_sql_database* on page 18
- *pdb_set_sql_database* on page 24

pdb_gen_id

Returns the current value of the database generation ID, or updates the value of the database generation ID.

Syntax

```
pdb_gen_id [number]
```

Parameters

- **number –** The value of the new database generation ID to be used when the database generation ID is updated. It must be a number between 0 and 32767.

Examples

- **Example 1 –**

  ```
  pdb_gen_id
  ```

  This command returns the current value of the database generation ID.
- **Example 2 –**

  ```
  pdb_gen_id 10
  ```

  This command updates the database generation ID to the value 10.

Usage

- When **pdb_gen_id** is invoked with no option, it returns the current value of the database generation ID stored in the RASD.
- When **pdb_gen_id** is invoked with the **number** option, it updates the value of the database generation ID in the RASD. Changing the database generation ID takes effect immediately.
- The database generation ID is the first 2 bytes of the origin queue ID. The database generation ID is used by Replication Server to support recovery operations, which may require Replication Agent to re-send transactions.

  During recovery, if Replication Agent must re-send operations that Replication Server has already processed, you can change the database generation ID to prevent Replication Server from recognizing the operations as already processed.
- For more information about the origin queue ID, see ra_helplocator, or refer to the section for your specific primary data server in the *Replication Agent Primary Database Guide*.
- If the RASD does not exist, the **pdb_gen_id** command returns an error.
- The **pdb_gen_id** command with parameters is valid when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *ra_helplocator* on page 89
- *ra_locator* on page 95

pdb_get_columns

Returns a list of columns in tables in the current database at the primary data server.

Syntax

```
pdb_get_columns [ownername, tablename[, colname]]
```

Parameters

- **ownername –** The user name of the owner of the table specified in the *tablename* option. This option can be delimited with quote characters to specify character case.
- **tablename –** The name of the table in the current database for which information is returned. This option can be delimited with quote characters to specify character case.
- **colname –** The name of the column for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

- **Example 1 –**

  ```
  pdb_get_columns
  ```

This command returns a list of all of the columns in all of the user tables in the current database.

- **Example 2 –**

```
pdb_get_columns bob, authors
```

This command returns a list of all of the columns in the table *authors*, owned by the user "bob" in the current database.

- **Example 3 –**

```
pdb_get_columns bob, authors, au_fname
```

This command returns information about the column *au_fname* in the table *authors*, owned by the user "bob" in the current database.

Usage

Note: Results from these commands are taken from the Replication Agent System database (RASD).

- When **pdb_get_columns** is invoked with no option, it returns a result set that lists all of the columns in all of the user tables in the current database.
- When **pdb_get_columns** is invoked with the *ownername* and *tablename* options, it returns a result set that lists all of the columns in the specified table with the specified owner in the current database.
- When **pdb_get_columns** is invoked with the *ownername*, *tablename*, and *colname* options, it returns a result set with information about the specified column in the specified table with the specified owner in the current database.
- The **pdb_get_columns** command accepts the % wildcard character in the *ownername*, *tablename*, and *colname* options.
- The current database is either:
  - The default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter, or
  - The database specified in the **pdb_set_sql_database** command (to which the Replication Agent instance is currently connected).
- To set or change the current database, use the **pdb_set_sql_database** command.

  Note: If the **pdb_set_sql_database** command has not been invoked to set or change the current database, the **pdb_get_columns** command returns information from the current database.
- To find the name of the current database, use the **pdb_get_sql_database** command.
- The **pdb_get_columns** command returns 0 rows if the specified table (with the specified owner) does not exist in the current database or if the specified column does not exist in the specified table.

- The **pdb_get_columns** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_databases* on page 14
- *pdb_get_primary_keys* on page 15
- *pdb_get_procedure_parms* on page 16
- *pdb_get_procedures* on page 17
- *pdb_get_tables* on page 19

pdb_get_databases

Returns a list of all user databases in the primary data server.

Note: The Oracle data server does not support multiple user databases. The **pdb_get_databases** command returns the name of the database instance.

Syntax

```
pdb_get_databases
```

Usage

- When **pdb_get_databases** is invoked, it returns a result set that lists all of the user databases in the primary data server.

  Note: Depending on the type of system database, the result set may or may not include the user database in the primary data server. See the section for your specific primary data server in the *Replication Agent Primary Database Guide*.
- The **pdb_get_databases** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_columns* on page 12
- *pdb_get_primary_keys* on page 15
- *pdb_get_procedure_parms* on page 16
- *pdb_get_procedures* on page 17
- *pdb_get_tables* on page 19

pdb_get_primary_keys

Returns a list of primary key columns in a specified table in the current database at the primary data server.

Syntax

```
pdb_get_primary_keys ownername, tablename
```

Parameters

- **ownername –** The user name of the owner of the table specified in *tablename*. This option can be delimited with quote characters to specify character case.
- **tablename –** The name of the table in the current database for which primary key column information is returned. This option can be delimited with quote characters to specify character case.

Usage

Note: Results from these commands are from the Replication Agent System database (RASD).

- When **pdb_get_primary_keys** is invoked, it returns a result set that lists all of the columns that are defined as primary keys in the specified table with the specified owner in the current database.
- The **pdb_get_primary_keys** command accepts the % wildcard character in the *ownername* option, but not in the *tablename* option.
- The current database is the default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter.
- To find the name of the current database, use the **pdb_get_sql_database** command.
- The **pdb_get_primary_keys** command returns 0 rows if the specified table with the specified owner does not exist in the current database.
- The **pdb_get_primary_keys** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_columns* on page 12
- *pdb_get_databases* on page 14
- *pdb_get_procedure_parms* on page 16
- *pdb_get_procedures* on page 17
- *pdb_get_tables* on page 19

pdb_get_procedure_parms

Returns a list of input parameters for procedures in the current database at the primary data server.

Syntax

```
pdb_get_procedure_parms [ownername, procname [, paramname]]
```

Parameters

- **ownername –** The user name of the owner of the procedure specified in *procname*. This option can be delimited with quote characters to specify character case.
- **procname –** The name of the procedure in the current database for which information is returned. This option can be delimited with quote characters to specify character case.
- **paramname –** The name of the input parameter for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

- **Example 1 –**

  ```
  pdb_get_procedure_parms
  ```

  This command returns a list of all of the input parameters for all of the procedures in the current database.

- **Example 2 –**

  ```
  pdb_get_procedure_parms bob, sp_foo
  ```

  This command returns a list of all of the input parameters for the procedure named *sp_foo*, owned by the user “bob” in the current database.

- **Example 3 –**

  ```
  pdb_get_procedure_parms bob, sp_foo, foo_count
  ```

  This command returns information about the input parameter *foo_count* for the procedure *sp_foo*, owned by the user “bob” in the current database.

Usage

Note: Results from these commands are from the Replication Agent System database (RASD).

- When **pdb_get_procedure_parms** is invoked with no option, it returns a result set that lists all of the input parameters for all the procedures in the current database.

- When **pdb_get_procedure_parms** is invoked with the *ownername* and *procname* options, it returns a result set that lists all of the input parameters for the specified procedure with the specified owner in the current database.
- When **pdb_get_procedure_parms** is invoked with the *ownername*, *procname*, and *paramname* options, it returns a result set with information about the specified input parameter for the specified procedure with the specified owner in the current database.
- The **pdb_get_procedure_parms** command accepts the % wildcard character in both the *ownername* and *procname* options.
- The current database is the default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter.
- To find the name of the current database, use the **pdb_get_sql_database** command.
- The **pdb_get_procedure_parms** command returns 0 rows if the specified procedure (with the specified owner) does not exist in the current database.
- The **pdb_get_procedure_parms** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_columns* on page 12
- *pdb_get_databases* on page 14
- *pdb_get_primary_keys* on page 15
- *pdb_get_procedures* on page 17
- *pdb_get_tables* on page 19

pdb_get_procedures

Returns a list of procedures in the current database at the primary data server.

Syntax

```
pdb_get_procedures [ownername, procname]
```

Parameters

- **ownername –** The user name of the owner of the procedure specified in *procname*. This option can be delimited with quote characters to specify character case.
- **procname –** The name of the procedure in the current database for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

- **Example 1 –**

  ```
  pdb_get_procedures
  ```

  This command returns a list of all of the procedures in the current database.

- **Example 2 –**

```
pdb_get_procedures bob, sp_foo
```

This command returns information about the procedure named *sp_foo*, owned by the user "bob" in the current database.

Usage

Note: Results from these commands are from the Replication Agent System database (RASD).

- When **pdb_get_procedures** is invoked with no option, it returns a result set that lists all of the procedures in the current database.
- When **pdb_get_procedures** is invoked with the *ownername* and *procname* options, it returns a result set with information about the specified procedure with the specified owner in the current database.
- The **pdb_get_procedures** command accepts the % wildcard character in both the *ownername* and *procname* options.
- The current database is the default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter.
- To find the name of the current database, use the **pdb_get_sql_database** command.
- The **pdb_get_procedures** command returns 0 rows if the specified procedure (with the specified owner) does not exist in the current database.
- The **pdb_get_procedures** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_columns* on page 12
- *pdb_get_databases* on page 14
- *pdb_get_primary_keys* on page 15
- *pdb_get_procedure_parms* on page 16
- *pdb_get_tables* on page 19

pdb_get_sql_database

Returns the name of the current database, if any.

Syntax

```
pdb_get_sql_database
```

Usage

- When **pdb_get_sql_database** is invoked, it returns the name of the current database.

- If the **pdb_set_sql_database** command has not been invoked to set the current database, it returns the default current database.
- The current database is the default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter.
- The **pdb_get_sql_database** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_execute_sql* on page 10
- *pdb_set_sql_database* on page 24

pdb_get_tables

Returns a list of user tables in the current database at the primary data server.

Syntax

```
pdb_get_tables [ownername, tablename]
```

Parameters

- **ownername –** The user name of the owner of the table specified in *tablename*. This option can be delimited with quote characters to specify character case.
- **tablename –** The name of the table in the current database for which information is returned. This option can be delimited with quote characters to specify character case.

Examples

- **Example 1 –**

  ```
  pdb_get_tables
  ```

  This command returns a list of all of the user tables in the current database.

- **Example 2 –**

  ```
  pdb_get_tables bob, authors
  ```

  This command returns information about the table *authors*, owned by the user "bob" in the current database.

Usage

Note: Results from these commands are from the Replication Agent System database (RASD).

- When **pdb_get_tables** is invoked with no option, it returns a result set that lists all of the user tables in the current database.

Note: System tables may or may not be returned by some primary data servers when the **pdb_get_tables** command is invoked.

- When **pdb_get_tables** is invoked with the *ownername* and *tablename* options, it returns a result set with information about the specified table with the specified owner in the current database.
- The **pdb_get_tables** command accepts the % wildcard character in both the *ownername* and *tablename* options.
- The current database is the default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter.
- To find the name of the current database, use the **pdb_get_sql_database** command.
- The **pdb_get_tables** command returns 0 rows if the specified table (with the specified owner) does not exist in the current database.
- The **pdb_get_tables** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_get_columns* on page 12
- *pdb_get_databases* on page 14
- *pdb_get_primary_keys* on page 15
- *pdb_get_procedure_parms* on page 16
- *pdb_get_procedures* on page 17

pdb_ownerfilter

Returns a list of the owners whose objects will be filtered for initialization; adds or removes owners to or from the list.

Syntax

```
pdb_ownerfilter [ {add | remove}, owner ]
```

Parameters

- **add –** The **add** keyword filters out any objects that are owned by the owner you specify. Any objects that are owned by this owner cannot be marked for initialization.
- **remove –** The **remove** keyword removes the filter for the owner you specify. Any objects that are owned by this owner can be marked for initialization. You cannot remove the "SYS" owner.
- **owner –** The name of the owner that is used for filtering.

  The *owner* option can be delimited with quote characters to specify the character case.

  If mixed case (uppercase and lowercase) is required, the name must be delimited. This parameter can be delimited with quotes to specify the character case. For example:

```
"Owner", "oWnEr"
```

Examples

- **Example 1 –**

  ```
  pdb_ownerfilter
  ```

  This command returns a list of all owners whose objects will be filtered for initialization.

- **Example 2 –**

  ```
  pdb_ownerfilter add, SYSTEM
  ```

  This command adds the "system" user to the list of owners whose objects will be filtered for replication.

- **Example 3 –**

  ```
  pdb_ownerfilter remove, SYSTEM
  ```

  This command removes the "system" user from the list of owners whose objects will be filtered for initialization.

Usage

- **pdb_ownerfilter** can be used to limit the number of objects that are loaded into the Replication Agent System Database during initialization (see **pdb_xlog init**). When **pdb_xlog init** is processed, the objects and owners in the **pdb_ownerfilter** list will not be loaded. You can reduce the size of the RASD and reduce the time to perform initialization by adding owners to the list whose objects are not be replicated, or for owners where the majority of objects are not to be replicated.

  Note: Any object marked for replication (using commands **pdb_setreptable**, **pdb_setrepproc**), is loaded into the RASD, even if the owner is not on the list. This list affects initialization processing, but not replication (replication occurs based on marking status, not owner filtering).

  The default owners are: CTXSYS, DBSNMP, DMSYS, IX, DSSYS, EXFSYS, HR, MDSYS, OE, ODM, ODM_MTR, OLADDBA, OLAPSYS, ORDPLUGINS, ORDSYS, OSE$HTTP$ADMIN, OUTLN, PM, PERFSTAT, QS, QS_ADM, QS_CBADM, QS_CS, QS_ES, QS_OS, QS_WS, RMAN, REPADMIN, SCOTT, SH, SYS, SYSMAN, SYSTEM, TRACESVR, TSMSYS, WKPROXY, WKSYS, WMSYS, XDB, FLOWS_030000, FLOWS_030100, WK_TEST, FLOWS_FILES, APEX_030200, ORDDATA, OWBSYS, and APPQOSSYS.
- When **pdb_ownerfilter** is invoked, its function is determined by the keywords and options you specify.
- When multiple keywords and options are specified, each must be separated by a comma. Blank space before or after a comma is optional. For example:

```
pdb_ownerfilter add, system
```

- When **pdb_ownerfilter** is invoked with no keyword, it returns a list of users whose objects will be filtered.
- The **pdb_ownerfilter** command is valid only when the Replication Agent instance is in the Admin or Replication Down state.
- You cannot remove the "SYS" owner.
- After initialization you can replicate any object with **pdb_setreptable** and **pdb_setrepproc**, except for the following objects which cannot be replicated at any time:
  - Objects that are owned by "SYS" owner.
  - Any system table whose name begins with V$.
  - Any system procedure or package whose name begins with DBMS.

See also

- *pdb_setrepproc* on page 37
- *pdb_setreptable* on page 47
- *pdb_xlog* on page 61
- *ra_admin* on page 66
- *ra_config* on page 70

pdb_send_osuser_list

Returns and modifies a list of database users whose primary database user names will be replaced with the corresponding operating system user name in the LTL sent to Replication Server.

Syntax

```
pdb_send_osuser_list [ { add | remove}, { user | all } ]
```

Parameters

- **add –** Use the **add** keyword to add primary database user names to the list of users whose primary database user names will be replaced with operating system user names in LTL sent to Replication Server. To add one user name, follow the **add** keyword with the *user* parameter:

  ```
  pdb_send_osuser_list add, user
  ```

  To add all valid primary database user names to the list, follow the **add** keyword with the **all** keyword:

  ```
  pdb_send_osuser_list add, all
  ```

- **remove –** Use the **remove** keyword to remove primary database user names from the list of users whose primary database user names will be replaced with operating system user names in LTL sent to Replication Server. To remove one user name, follow the **remove** keyword with the *user* parameter:

```
pdb_send_osuser_list remove, user
```

To remove all user names from the list, follow the **remove** keyword with the **all** keyword:

```
pdb_send_osuser_list remove, all
```

To display a list of all user names in the list of users whose primary database user names will be replaced with operating system user names in LTL sent to Replication Server, use the **pdb_send_osuser_list** command alone:

pdb_send_osuser_list

Examples

- **Example 1 –**

  ```
  pdb_send_osuser_list add, dbuser1
  ```

  This command adds the primary database user name dbuser1 to the list. If the operating system user name corresponding to dbuser1 is osuser1, the LTL that Replication Agent sends to Replication Server will contain the user name osuser1 instead of dbuser1.

- **Example 2 –**

  ```
  pdb_send_osuser_list add, all
  ```

  This command adds all valid primary database user names to the list. The LTL that Replication Agent sends to Replication Server will contain the operating system user names corresponding to all valid primary database user names in the list.

- **Example 3 –**

  ```
  pdb_send_osuser_list remove, dbuser1
  ```

  This command removes the primary database user name dbuser1 from the list. If the operating system user name corresponding to dbuser1 is osuser1, the LTL that Replication Agent sends to Replication Server will contain the primary database user name dbuser1 instead of osuser1.

- **Example 4 –**

  ```
  pdb_send_osuser_list remove, all
  ```

  This command removes all primary database user names from the list. The LTL that Replication Agent sends to Replication Server will contain primary database user names, not operating system user names.

- **Example 5 –**

  ```
  pdb_send_osuser_list
  ```

  This command lists all database users whose primary database user names will be replaced with a corresponding operating system user name in the LTL sent to Replication Server.

Usage

- The **pdb_send_osuser_list** command with parameters is valid only when the Replication Agent instance is in the Admin or Replication Down state.
- The **pdb_send_osuser_list** command with parameters affects only database users who are logged in to the primary database at the time **pdb_send_osuser_list** is invoked.

pdb_set_sql_database

Sets the current database to be used for SQL statement execution.

Syntax

```
pdb_set_sql_database database
```

Parameters

- **database –** The name of the database in the primary data server against which Replication Agent can execute SQL statements (queries). To specify character case, delimit this parameter with quote characters.

Usage

- When **pdb_set_sql_database** is invoked, it sets the "current" database, in which Replication Agent can execute SQL queries.

  Note: The **pdb_set_sql_database** command has no effect in Oracle, but it is included to provide continuity with other Replication Agents that support database servers with multiple databases.
- Replication Agent does not validate the database name you specify with **pdb_set_sql_database**.

  If you specify an invalid database name, no error is returned until one of the following Replication Agent commands is invoked:
  - **pdb_execute_sql**
  - **pdb_get_columns**
  - **pdb_get_primary_keys**
  - **pdb_get_procedure_parms**
  - **pdb_get_procedures**
  - **pdb_get_tables**
- To find the name of the current database, use **pdb_get_sql_database**.

  Note: If the **pdb_set_sql_database** command has not been invoked to set the current database, the **pdb_get_sql_database** command returns the default current database, which is the primary database specified in the Replication Agent **pds_database_name** configuration parameter.

- The **pdb_set_sql_database** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_execute_sql* on page 10
- *pdb_get_sql_database* on page 18

pdb_setrepcol

Returns LOB column replication status; enables or disables replication for LOB columns within marked tables.

Syntax

```
pdb_setrepcol [
   {
    enable
   |
    disable
   |
    tablename[, colname[, { enable | disable[, force] } ] ]
   |
    all, { enable | disable[, force] }
   }
]
```

Parameters

- **tablename –** The name of the user table in the primary database that contains the column specified in the *colname* option.

  The *tablename* option can be owner-qualified (include the owner name), with each element separated by a period. For example:

  ```
  owner.table
  ```

  The *tablename* option can be delimited with quote characters to specify the character case.

  If mixed case (uppercase and lowercase) is required, the name must be delimited. For example:

  ```
  "Owner".table
  ```

  ```
  "Owner"."Table"
  ```

  Each mixed-case element of the *tablename* option must be delimited separately, as shown in the previous example.

  Note: If you must use an object name case that does not match the value of the **ltl_character_case** parameter, the object name must be delimited.

  If an object name contains any non-alphanumeric characters, such as spaces or periods, it must be delimited with quote characters. For example:

```
"table name"
```

```
owner."table name"
```

If an object name contains a period, it must be both owner-qualified and delimited with quote characters. For example:

```
owner."table.name"
```

```
"table.owner"."table.name"
```

- **colname –** The name of a LOB column in the user table specified in the *tablename* option.

  The *colname* option can be delimited with quote characters to specify the character case.

  If mixed character case (both uppercase and lowercase) is required, the name must be delimited. For example:

  ```
  "Colname"
  ```

  ```
  "COLname"
  ```

  Note: If you must use a column name case that does not match the value of the **ltl_character_case** parameter, the column name must be delimited. See ltl_character_case for more information.

- **all –** A keyword that refers to all LOB columns in marked tables in the primary database. By using the **all** keyword, you can apply an enable or disable operation to all LOB columns in marked tables.
- **enable –** A keyword that refers to enabling replication for LOB columns.
- **disable –** A keyword that refers to disabling replication for LOB columns.
- **force –** A keyword that refers to forcing replication to be disabled for LOB columns.

  When the **force** keyword follows the **disable** keyword, the **pdb_setrepcol** command immediately disables replication for the specified LOB column. When the **force** keyword follows the **disable** keyword and the **all** keyword, the **pdb_setrepcol** command immediately disables replication for all marked LOB columns in marked tables in the primary database.

Examples

- **Example 1 –**

  ```
  pdb_setrepcol
  ```

  This command returns replication information for all enabled LOB columns in marked tables in the primary database.

- **Example 2 –**

  ```
  pdb_setrepcol authors
  ```

  This command returns replication information for all LOB columns defined for the table *authors* in the primary database.

- **Example 3 –**

  ```
  pdb_setrepcol authors, picture
  ```

  This command returns replication information for the column called “picture” in the table *authors* in the primary database.

- **Example 4 –**

  ```
  pdb_setrepcol authors, picture, enable
  ```

  This command enables replication for the column *picture* in the table *authors* in the primary database.

- **Example 5 –**

  ```
  pdb_setrepcol all, disable
  ```

  This command disables replication for all LOB columns in all marked tables in the primary database.

Usage

- If a column is renamed or dropped and a new column with the original name is created, you must explicitly enable or disable replication from this new column because the new column has no replication status related information from the original column. The marking information is maintained internally based on column number, not column name.
- When **pdb_setrepcol** is invoked, its function is determined by the keywords and options you specify.
- When multiple keywords or options are specified, each must be separated by a comma. Blank space before or after a comma is optional. For example:

  ```
  pdb_setrepcol all, disable
  ```

- When you specify a column name in the **pdb_setrepcol** command, you must use the name of a valid LOB column.
- You cannot specify the following items as a table name in the **pdb_setrepcol** command:
  - Primary database system tables
  - Aliases or synonyms
  - Views
  - Replication Agent transaction log objects
- If a column name in the primary database is the same as a keyword, it can be identified by adding the string **col=** to the beginning of the column name. For example:

  ```
  pdb_setrepcol tablename, col=enable, disable
  ```

- If you enable LOB column replication with the **pdb_setrepcol** command, do not configure Replication Agent to convert `date` or `time` datatypes in the primary database.
- When **pdb_setrepcol** is invoked with either no option or a single option, it returns information about the enabled status of LOB columns in the primary database.
  - If **pdb_setrepcol** is invoked with no option, it returns a list of all LOB columns for which replication is enabled in the primary database.

Note: Invoking the **pdb_setrepcol** command with no option produces the same result as invoking the **pdb_setrepcol** command with the **enable** keyword.

- If **pdb_setrepcol** is invoked with a table name, it returns information about the enabled status of all the LOB columns in the specified primary table.
- If **pdb_setrepcol** is invoked with the **enable** keyword, it returns a list of all LOB columns for which replication is enabled in the primary database.
- If **pdb_setrepcol** is invoked with the **disable** keyword, it returns a list of all LOB columns for which replication is disabled in the primary database.

For LOB columns listed as disabled, transactions are not captured for replication.

- When **pdb_setrepcol** is invoked with a valid primary table name and valid LOB column name, with no keywords, it returns information about the enabled status of the specified LOB column in the specified table in the primary database.
- When **pdb_setrepcol** is invoked with the **all** keyword, the operation specified by the following keyword (**enable** or **disable**) is applied to all LOB columns in marked tables in the primary database.
  - If **pdb_setrepcol** is invoked with the **all** keyword and the **enable** keyword, it enables replication for all LOB columns in marked tables in the primary database.
  - If **pdb_setrepcol** is invoked with the **all** keyword and the **disable** keyword, it disables replication for all LOB columns in marked tables in the primary database.
- When **pdb_setrepcol** is invoked with a valid primary table name and valid LOB column name followed by one or more keywords, the operation specified by the keyword (**enable** or **disable**) is applied to the specified LOB column in the specified primary table.
  - If **pdb_setrepcol** is invoked with a table name and LOB column name and the **enable** keyword, it enables replication for the specified LOB column in the primary database.
  - If **pdb_setrepcol** is invoked with a table name and LOB column name and the **disable** keyword, it disables replication for the specified LOB column in the primary database.

  If the table name and LOB column name combination you specify does not exist in the primary database, the **pdb_setrepcol** command returns an error.
- If the Replication Agent transaction log does not exist in the RASD is not initialized, the **pdb_setrepcol** command returns an error.

See also

- *pdb_setrepproc* on page 37
- *pdb_setreptable* on page 47
- *ra_config* on page 70
- *ltl_character_case* on page 163

pdb_setrepddl

Returns DDL replication status and enables or disables replication for DDL statements.

Syntax

```
pdb_setrepddl [
   {   tablename
     | procname
     | sequence_name
     | objects, all
     | user, { all | user } ]
     | stmt, { all | ddl_statement | ddl_statement_keyword }
     | owner, { all | ownername }
    } ]
   [
        { enable[, override] | disable[, override] | default }
      | { enable, { all | marked | unmarked } }
    ]
```

Parameters

- **override –** To enable the replication of DDL statements and override any existing filtering rules, follow the **pdb_setrepddl** command with the **enable** and **override** keywords:

  ```
  pdb_setrepddl enable, override
  ```

  To disable the replication of DDL statements and override any existing filtering rules, follow the **pdb_setrepddl** command with the **disable** and **override** keywords:

  ```
  pdb_setrepddl enable, override
  ```

- **tablename –** The name of a user table in the primary database. To enable or disable the replication of DDL involving a table, use the *tablename* parameter:

  ```
  pdb_setrepddl tablename [ , { enable[ , override] | disable |
  default } ]
  ```

  To list the current filter setting for a table, enter the *tablename* parameter alone:

  ```
  pdb_setrepddl tablename
  ```

- **procname –** The name of a procedure in the primary database. To enable or disable the replication of DDL involving a procedure name, use the *procname* parameter:

  ```
  pdb_setrepddl procname [ , { enable[, override] | disable |
  default } ]
  ```

  To list the current filter setting for a procedure, enter the *procname* parameter alone:

  ```
  pdb_setrepddl procname
  ```

- **sequence_name –** The name of a user sequence in the primary database. To enable or disable the replication of DDL involving a sequence, use the *sequence_name* parameter:

  ```
  pdb_setrepddl sequence_name [ , { enable[, override] | disable |
  default } ]
  ```

To list the current filter setting for a sequence, enter the *sequence_name* parameter alone:

```
pdb_setrepddl sequence_name
```

- **objects, all –** The **objects** keyword must be used with the keyword **all** and allows you to enable or disable the replication of DDL statements for all objects:

  ```
  pdb_setrepddl objects, all [ , { enable[, override] | disable |
  default } ]
  ```

  To list all objects for which DDL statements are filtered, follow the **pdb_setrepddl** command with the **objects, all** keywords:

  ```
  pdb_setrepddl objects, all
  ```

- **user –** The **user** keyword allows you to enable or disable the replication of DDL statements executed by primary database users. To enable or disable the replication of DDL from a specified user, use the *user* parameter.

  ```
  pdb_setrepddl user, user [ , { enable[, override] | disable |
  default } ]
  ```

  To list the current filter setting for a user, follow the **user** keyword with the *user* parameter:

  ```
  pdb_setrepddl user, user
  ```

  To list database users whose DDL statements will be filtered from replication, enter the **user** keyword alone:

  ```
  pdb_setrepddl user
  ```

  To enable or disable the replication of DDL statements for all users, follow the **user** keyword with the **all** keyword:

  ```
  pdb_setrepddl user, all [ , { enable[, override] | disable |
  default } ]
  ```

- **stmt –** The **stmt** keyword allows you to enable or disable the replication of DDL statements. To enable or disable the replication of DDL for a particular statement, use the *ddl_statement* parameter, which contains a string in the form of a DDL statement enclosed in single or double quotes:

  ```
  pdb_setrepddl stmt, ddl_statement [ , { enable[, override] |
  disable | default } ]
  ```

  To list the current filter setting for a particular DDL statement, follow the **stmt** keyword with the *ddl_statement* parameter:

  ```
  pdb_setrepddl stmt, ddl_statement
  ```

  To enable or disable the replication of DDL for a entire set of statements, use the *ddl_statement_keyword* parameter, which contains a string in the form of a DDL statement keyword:

  ```
  pdb_setrepddl stmt, ddl_statement_keyword [ , { enable[, override]
  | disable | default } ]
  ```

Table 2. The ddl_statement_keyword Parameter

Keyword Value	DDL Statements Filtered
cluster	**alter cluster**, **create cluster**, **drop cluster**, **truncate cluster**
context	**alter context**, **drop context**
dimension	**alter dimension**, **create dimension**, **drop dimension**
directory	**alter directory**, **drop directory**
function	**alter function**, **create function**, **drop function**
index	**alter index**, **create index**, **drop index**
indextype	**alter indextype**, **create indextype**, **drop indextype**
java	**alter java**, **create java**, **drop java**
library	**alter library**, **drop library**
materialized_view	**alter materialized view**, **create materialized view**, **drop materialized view**
operator	**alter operator**, **create operator**, **drop operator**
outline	**alter outline**, **create outline**, **drop outline**
package	**alter package**, **create package**, **drop package**
package body	**alter package body**, **create package body**, **drop package body**
point	**alter restore point**, **drop restore point**
procedure	**alter procedure**, **create procedure**, **drop procedure**
profile	**alter profile**, **create profile**, **drop profile**
role	**alter role**, **create role**, **drop role**
sequence	**alter sequence**, **create sequence**, **drop sequence**
synonym	**alter (public) synonym**, **drop (public) synonym**
table	**alter table**, **create table**, **drop table**
trigger	**alter trigger**, **create trigger**, **drop trigger**
type	**alter type**, **create type**, **drop type**
type body	**alter type body**, **create type body**, **drop type body**
user	**alter user**, **create user**, **drop user**
view	**alter view**, **create view**, **drop view**

To enable or disable the replication of all DDL statements, follow the **stmt** keyword with the **all** keyword:

```
pdb_setrepddl stmt, all [ , { enable[, override] | disable |
default } ]
```

To list DDL statements that will be filtered from replication, enter the **stmt** keyword alone:

```
pdb_setrepddl stmt
```

- **owner –** The **owner** keyword allows you to enable or disable the replication of DDL statements affecting an object owned by a particular user. To enable or disable the replication of DDL statements affecting objects owned by a particular user, use the *ownername* parameter to specify the user:

  ```
  pdb_setrepddl owner, ownername [ , { enable[, override] | disable
  | default } ]
  ```

  To list the current filter setting for an object owner, follow the **owner** keyword with the *ownername* parameter:

  ```
  pdb_setrepddl owner, ownername
  ```

  To list the object owners for which DDL statements will be filtered from replication, enter the **owner** keyword alone:

  ```
  pdb_setrepddl owner
  ```

  To enable or disable the replication of DDL statements affecting objects for all owners, follow the **owner** keyword with the **all** keyword:

  ```
  pdb_setrepddl owner, all [ , { enable[, override] | disable |
  default } ]
  ```

- **enable –** For Replication Agent for Oracle, the **enable** keyword allows you to enable the replication of DDL statements as specified by other keywords and parameters in the **pdb_setrepddl** command. To override any previous filtering of DDL statements, follow the **enable** keyword with the **override** keyword.

  For multiple Replication Agents and Replication Server Multi-Path Replication™:

  - **all** enables the replication of all DDL for the invoking Replication Agent instance within the Replication Agent group. For example:

    ```
    pdb_setrepddl enable, all
    ```

  - **marked** enables the replication of DDL only for objects that have been marked by the invoking Replication Agent instance within the Replication Agent group.

    ```
    pdb_setrepddl enable, marked
    ```

  - **unmarked** enables the replication of DDL for objects that have not been marked by any Replication Agent instance within the Replication Agent group.

    ```
    pdb_setrepddl enable, unmarked
    ```

- **disable –** The **disable** keyword allows you to disable the replication of DDL statements as specified by other keywords and parameters in the **pdb_setrepddl** command.
- **default –** If replication is not explicitly enabled or disabled for a particular owner, object, or DDL statement, the **default** keyword results in DDL statement filtering being enabled or disabled according to the following hierarchy:

1. Object: Any DDL filtering rules for an object will be observed, but not rules involving the object owner, statements affecting the object, or the user.
2. Owner: Any DDL filtering rules for an object owner will be observed, but not rules involving statements affecting the object or the user.
3. Statement: Any DDL filtering rules for DDL statements will be observed, but not rules involving the user.
4. User: Any DDL filtering rules involving the user will be observed.

- **enable (for multiple Replication Agents) –** For Replication Agent for Oracle, the **enable** keyword allows you to enable the replication of DDL statements as specified by other keywords and parameters in the **pdb_setrepddl** command. To override any previous filtering of DDL statements, follow the **enable** keyword with the **override** keyword.

Examples

- **Example 1 –**

```
pdb_setrepddl stmt
```

  This command lists DDL statements that are filtered from replication.

- **Example 2 –**

```
pdb_setrepddl stmt, 'create index'
```

  This command lists the current filter setting for the **create index** statement.

- **Example 3 –**

```
pdb_setrepddl stmt, 'create index', disable
```

  This command causes Replication Agent to filter the **create index** statement from replication.

- **Example 4 –**

```
pdb_setrepddl stmt, index, disable
```

  This command causes Replication Agent to filter the **alter index**, **create index**, and **drop index** statements from replication.

- **Example 5 –**

```
pdb_setrepddl owner
```

  This command lists the object owners for which DDL statements will be filtered from replication.

- **Example 6 –**

```
pdb_setrepddl owner, myuser
```

  This command lists the current filter setting for an object owner.

- **Example 7 –**

  ```
  pdb_setrepddl owner, myuser, disable
  ```

  This command causes Replication Agent to filter DDL statements affecting objects owned by the user named *myuser*.

- **Example 8 –**

  ```
  pdb_setrepddl myuser.mytable
  ```

  This command lists the current filter setting for the table named *myuser.mytable*.

- **Example 9 –**

  ```
  pdb_setrepddl myuser.mytable, disable
  ```

  This command causes Replication Agent to filter all DDL statements that affect the table named *myuser.mytable*.

- **Example 10 –**

  ```
  pdb_setrepddl owner, myuser, disable
  ```

  ```
  pdb_setrepddl myuser.mytable, enable
  ```

  These commands cause Replication Agent to filter all DDL statements affecting objects owned by the user named myuser except the table named *myuser.mytable*.

- **Example 11 –**

  ```
  pdb_setrepddl owner, all, disable
  ```

  ```
  pdb_setrepddl owner, myuser, enable
  ```

  These commands cause Replication Agent to replicate only DDL statements affecting objects owned by the user named *myuser*.

- **Example 12 –**

  ```
  pdb_setrepddl user, myuser, disable
  ```

  This command causes Replication Agent to filter DDL from the user named *myuser*.

- **Example 13 –**

  ```
  pdb_setrepddl enable, override
  ```

  This command enables DDL replication and overrides any existing filter settings.

- **Example 14 –**

  ```
  pdb_setrepddl owner, myuser, enable, override
  ```

  This command enables replication of DDL from the user named *myuser* and overrides any existing filter settings.

- **Example 15 –**

  ```
  pdb_setrepddl
  ```

This command returns the current DDL replication status for the primary database.

- **Example 16 –**

```
pdb_setrepddl enable
```

This command enables replication of DDL commands issued into the primary database after this point in time.

- **Example 17 –**

```
pdb_setrepddl disable
```

This command disables replication of DDL commands issued into the primary database after this point in time.

Usage

- In addition to enabling DDL replication using **pdb_setrepddl** command, you must set the Replication Agent **ddl_username** and **ddl_password** parameters.
- A database replication definition that enables DDL is required for DDL replication. If the **use_rssd** configuration parameter is set to true, the database replication definition must exist in the RSSD of the primary Replication Server before the Replication Agent **resume** command is invoked.
- Only DDL statements that have identical syntax in the primary and replicate databases can be replicated. Replication Agent sends DDL statements using the syntax of the statements in the primary database.
- If Replication Agent has not been initialized, the **pdb_setrepddl** command returns an error.
- The **pdb_setrepddl** command can only use the **enable** and **disable** options when the Replication Agent instance is in the Admin or Replication Down state.
- When using the **all**, **marked**, and **unmarked** keywords for DDL replication and multiple Replication Agents:
  - You can also enable the replication of all DDL by omitting the **all** keyword:

    ```
    pdb_setrepddl enable
    ```
  - The **marked** and **unmarked** keywords do not override each other. For example, to enable the replication of both DDL for objects that have not been marked by any Replication Agent instance within the Replication Agent group and DDL for objects that have been marked by the invoking Replication Agent instance:

    ```
    pdb_setrepddl enable, unmarked
    pdb_setrepddl enable, marked
    ```

    However, **marked** and **unmarked** will override previous use of the **all** keyword, and the **all** keyword will override previous use of the **marked** and **unmarked** keywords.

Oracle Privileges for DDL Replication:

Note: Issuing **GRANT ALL PRIVILEGES TO DDLUSER** turns the DDL user into a superuser, like the SYS or SYSTEM user.

For Oracle 10g and 11g, grant user permission to the DDL user to execute:

- **GRANT ALTER ANY INDEX**
- **GRANT ALTER ANY INDEXTYPE**
- **GRANT ALTER ANY PROCEDURE**
- **GRANT ALTER ANY TABLE**
- **GRANT ALTER ANY TRIGGER**
- **GRANT ALTER ANY TYPE**
- **GRANT ALTER SESSION**
- **GRANT BECOME USER**
- **GRANT CREATE ANY INDEX**
- **GRANT CREATE ANY INDEXTYPE**
- **GRANT CREATE ANY PROCEDURE**
- **GRANT CREATE ANY SYNONYM**
- **GRANT CREATE ANY TABLE**
- **GRANT CREATE ANY TRIGGER**
- **GRANT CREATE ANY TYPE**
- **GRANT CREATE ANY VIEW**
- **GRANT CREATE INDEXTYPE**
- **GRANT CREATE MATERIALIZED VIEW**
- **GRANT CREATE PROCEDURE**
- **GRANT CREATE PUBLIC SYNONYM**
- **GRANT CREATE SYNONYM**
- **GRANT CREATE TABLE**
- **GRANT CREATE TRIGGER**
- **GRANT CREATE TYPE**
- **GRANT CREATE VIEW**
- **GRANT DELETE ANY TABLE**
- **GRANT DROP ANY INDEX**
- **GRANT DROP ANY INDEXTYPE**
- **GRANT DROP ANY MATERIALIZED VIEW**
- **GRANT DROP ANY PROCEDURE**
- **GRANT DROP ANY SYNONYM**
- **GRANT DROP ANY TABLE**
- **GRANT DROP ANY TRIGGER**
- **GRANT DROP ANY TYPE**
- **GRANT DROP ANY VIEW**
- **GRANT DROP PUBLIC SYNONYM**
- **GRANT INSERT ANY TABLE**
- **GRANT SELECT ANY TABLE**

- **GRANT UPDATE ANY TABLE**

Revoke user permission from the DDL user to execute:

- **GRANT ALTER DATABASE**
- **GRANT ALTER ROLLBACK SEGMENT**
- **GRANT ALTER SYSTEM**
- **GRANT ALTER TABLESPACE**
- **GRANT ANALYZE ANY**
- **GRANT AUDIT ANY**
- **GRANT AUDIT SYSTEM**
- **GRANT CREATE DATABASE LINK**
- **GRANT CREATE ROLLBACK SEGMENT**
- **GRANT CREATE TABLESPACE**
- **GRANT DROP PUBLIC DATABASE LINK**
- **GRANT DROP ROLLBACK SEGMENT**
- **GRANT DROP TABLESPACE**
- **GRANT LOCK ANY TABLE**

See also

- *ddl_password* on page 151

pdb_setrepproc

Returns stored procedure replication marking status; marks specified procedures for replication; unmarks all marked procedures or a specified procedure; enables or disables replication for all marked procedures or a specified procedure.

Syntax

```
pdb_setrepproc
  [ { procname[, repname,] { mark
                           | unmark[, force]
                           | enable
                           | disable }
    | all, { unmark[, force]
           | enable
           | disable }
  } ]
```

To return stored procedure replication marking status:

```
pdb_setrepproc [ {procname|mark|unmark|enable|disable } ]
```

To unmark, enable, or disable all marked stored procedures:

```
pdb_setrepproc all, {unmark[, force]|enable|disable}
```

To mark a specified stored procedure for replication with a replicated name:

```
pdb_setrepproc procname, [repname,] mark
```

To unmark, enable, or disable a specified stored procedure:

```
pdb_setrepproc procname, {unmark[, force]|enable|disable}
```

Parameters

- **procname –** The name of a user stored procedure in the primary database.

  The *procname* option can be delimited with quote characters to specify the character case.

  If mixed character case (both uppercase and lowercase) is required, the name must be delimited. For example:

  ```
  "Proc"
  ```

  Note: If you must use an object name case that does not match the setting of the **ltl_character_case** parameter, the object name must be delimited.

  If an object name contains any non-alphanumeric characters, such as spaces, periods, and so forth, it must be delimited with quote characters. For example:

  ```
  "proc name"
  ```

  ```
  "proc.name"
  ```

  If an object name contains a period, it must be both owner-qualified and delimited with quote characters. For example:

  ```
  owner."proc.name"
  ```

  ```
  "proc.owner"."proc.name"
  ```

- **repname –** The name of the stored procedure specified in a function replication definition for the primary stored procedure.

  The *repname* option can be delimited with quote characters to specify character case. See the previous description of the *procname* option for details.

  By specifying a replicated name, stored procedure invocations can be replicated to a stored procedure invocation in the replicate database that has a different stored procedure name from the primary database.

  Note: The replicated name you specify with the **pdb_setrepproc** command must match the name specified by a **with primary function named** clause in a Replication Server function replication definition for the primary database connection. Replication Agent cannot validate the function replication definition, but if it does not exist, function replication from the primary database fails.

- **all –** A keyword that refers to all user stored procedures in the primary database. By using the **all** keyword, you can mark all user stored procedures, or apply an unmark, enable, or disable operation to all marked stored procedures.
- **mark –** A keyword that refers to marking user stored procedures for replication. The Replication Agent must have **CREATE ANY PROCEDURE** permission to mark procedures for replication.

Note: You must specify an owner when using the **mark** keyword.

- **unmark –** A keyword that refers to unmarking marked stored procedures.
- **force –** A keyword that refers to the unmark operation. When the **force** keyword follows the **unmark** keyword, the **pdb_setrepproc** command immediately unmarks the specified stored procedure in the primary database, without first checking the enable status of the stored procedure. When the **force** keyword follows the **unmark** keyword and the **all** keyword, the **pdb_setrepproc** command immediately removes replication marking from all marked stored procedures in the primary database, regardless of their enable status.

  The **force** keyword also forces complete execution of the unmarking script, even if errors occur during the unmarking process. Normally, when errors occur during script execution, the script terminates immediately without completing.

  Note: Read the "Usage" section that follows to better understand how scripts are used in Oracle procedure marking and unmarking.

  The **force** keyword can be useful when a previous script execution failed and left the unmarking operation incomplete. When errors occur during a forced script execution, the **pdb_setrepproc** command returns this message:

  ```
  Errors were encountered and ignored during FORCED script
  execution. See error log for details.
  ```

- **enable –** A keyword that refers to enabling replication for marked stored procedures.
- **disable –** A keyword that refers to disabling replication for marked stored procedures.

Examples

- **Example 1 –**

  ```
  pdb_setrepproc
  ```

  This command returns replication marking information for all marked stored procedures in the primary database.

- **Example 2 –**

  ```
  pdb_setrepproc authors
  ```

  This command returns replication marking information for the user stored procedure named "authors" in the primary database.

- **Example 3 –**

  ```
  pdb_setrepproc authors, mark
  ```

  This command marks the user stored procedure named "authors" in the primary database.

- **Example 4 –**

  ```
  pdb_setrepproc authors, enable
  ```

This command enables replication for the marked stored procedure named “authors” in the primary database.

- **Example 5 –**

```
pdb_setrepproc all, unmark
```

This command unmarks all marked stored procedures in the primary database.

Usage

- If a marked procedure is renamed or dropped and a new procedure with the original name is created, you must explicitly mark the new procedure because the new procedure has no marking-related information from the original procedure. The marking information is maintained internally by object id, not object name.
- How you use the **pdb_setrepproc** command depends on the type of replication definition that you have created at Replication Server. If you have created a database replication definition with no function replication definition, the replicate procedure in the **pdb_setrepproc** command refers to the procedure in the replicate database. However, if you have created a function replication definition, the replicate procedure in the **pdb_setrepproc** command refers to the name of the function replication definition, and it is the function replication definition that must map to the procedure in the replicate database.

  If no function replication definition exists and will not be added prior to replication, but only a database replication definition exists, use these commands to mark a procedure for replication:
  - When the procedure in the replicate database has the same name as the procedure in the primary database:

    ```
    pdb_setrepproc pdb_proc, mark
    ```

    where:

    pdb_proc is the name of the procedure in the primary database that you want to mark for replication.
  - When the procedure in the replicate database has the different name than the procedure in the primary database:

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

    where:

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

    rep_proc is the name of the procedure in the replicate database.

  If a function replication definition exists or will be added prior to replication, regardless of whether or not a database replication definition exists, use these commands to mark a procedure for replication:
  - When the function replication definition has the same name as the procedure in the primary database:

```
pdb_setrepproc pdb_proc, mark
```

where:

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

- If the procedure in the replicate database also has the same name as the function replication definition, there is no need to use the “deliver as” clause in the replication definition in the primary Replication Server. For example:

  ```
  create function replication definition pdb_proc with primary
  at data_server.
  database …
  ```

- If the procedure in the replicate database has a different name than the name of function replication definition, the function replication definition must map to the procedure in the replicate database. For example:

  ```
  create function replication definition pdb_proc with primary
  at data_server.database deliver as ‘rep_proc’ …
  ```

- When the name of the function replication definition is different than the procedure in the primary database:

  ```
  pdb_setrepproc pdb_proc , rdpri_proc, mark
  ```

  where:

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

  rdpri_proc is the name of the function replication definition.

  - If the procedure in the replicate database also has the same name as the function replication definition, there is no need to use the “deliver as” clause in the replication definition in the primary Replication Server. For example:

    ```
    create function replication definition rdpri_proc with
    primary at data_server.database …
    ```

  - If the procedure in the replicate database has a different name from the function replication definition, the function replication definition must map to the procedure in the replicate database. For example:

    ```
    create function replication definition rdpri_proc with
    primary at data_server.database deliver as 'rep_proc' …
    ```

- When multiple keywords and options are specified, each must be separated by a comma. Blank space before or after a comma is optional. For example:

  ```
  pdb_setrepproc all, unmark, force
  ```

- When you specify a stored procedure name in the **pdb_setrepproc** command, you must use the name of a valid user stored procedure.
- You cannot specify these items as a stored procedure name in the **pdb_setrepproc** command:
  - System procedures
  - Replication Agent transaction log procedures

- If a stored procedure name in the primary database is the same as a keyword, it can be identified by adding the string `proc=` to the beginning of the stored procedure name. For example:

  ```
  pdb_setrepproc proc=unmark, mark
  ```
- When you use the **unmark** keyword to remove replication marking from a stored procedure, Replication Agent verifies that replication is disabled for that stored procedure and there are no pending (unprocessed) operations for that stored procedure in the transaction log. If replication is not disabled for that procedure, or if there is a pending operation for that procedure in the transaction log, **pdb_setrepproc** returns an error.
- When **pdb_setrepproc** is invoked with either no option or a single option, it returns marking information about the stored procedures in the primary database.
  - If **pdb_setrepproc** is invoked with no option, it returns a list of all marked procedures in the primary database.

    Note: Invoking the **pdb_setrepproc** command with no option produces the same result as invoking the **pdb_setrepproc** command with only the **mark** keyword.
  - If **pdb_setrepproc** is invoked with a procedure name, it returns complete marking information about the specified procedure.
  - If **pdb_setrepproc** is invoked with the **mark** keyword, it returns a list of all marked procedures in the primary database.
  - If **pdb_setrepproc** is invoked with the **unmark** keyword, it returns a list of all unmarked procedures in the primary database.
  - If **pdb_setrepproc** is invoked with the **enable** keyword, it returns a list of all marked procedures in the primary database, for which replication is currently enabled.
  - If **pdb_setrepproc** is invoked with the **disable** keyword, it returns a list of all marked procedures in the primary database, for which replication is currently disabled.

  Stored procedures marked for replication are recorded in the RASD. All other user procedures are considered unmarked.

  Note: The Replication Agent system procedures are not included in the list of unmarked procedures. Also not included are any synonyms or aliases for these procedures.

  For procedures listed as unmarked or disabled, their invocations are not captured for replication.
- When **pdb_setrepproc** is invoked with the **all** keyword and an action keyword (**unmark**, **enable**, or **disable**), the action specified is applied to either all user stored procedures in the primary database, or to all marked procedures in the primary database.
  - If **pdb_setrepproc** is invoked with the **all** and **unmark** keywords, it removes replication marking from all marked procedures in the primary database.

    You can specify the **force** keyword after the **unmark** keyword to force immediate unmarking of all marked procedures, including procedures for which replication is still enabled.

  - If **pdb_setrepproc** is invoked with the **all** and **enable** keywords, it enables replication for all marked procedures in the primary database.
  - If **pdb_setrepproc** is invoked with the **all** and **disable** keywords, it disables replication for all marked procedures in the primary database.
- When **pdb_setrepproc** is invoked with a valid user stored procedure name and followed by an action keyword (**mark**, **unmark**, **enable**, or **disable**), the action specified is applied to the specified procedure.
  - If **pdb_setrepproc** is invoked with a procedure name and the **mark** keyword, it marks the specified procedure in the primary database for replication.
  - If **pdb_setrepproc** is invoked with a procedure name and the **unmark** keyword, it removes replication marking from the specified procedure in the primary database.
  - If **pdb_setrepproc** is invoked with a procedure name and the **enable** keyword, it enables replication for the specified marked procedure in the primary database.
  - If **pdb_setrepproc** is invoked with a procedure name and the **disable** keyword, it disables replication for the specified marked procedure in the primary database.

  Note: Use [**mark** | **unmark**] instead of [**enable** | **disable**] since the results are the same.
- If you specify a stored procedure name that does not exist in the primary database, the **pdb_setrepproc** command returns an error.
- When **pdb_setrepproc** is invoked with a procedure name and a replicated name, followed by the **mark** keyword, the primary procedure is marked for replication with the specified replicated name.

  If the primary procedure name you specify does not exist in the primary database, the **pdb_setrepproc** command returns an error.

  By specifying a replicated name, procedure invocations can be replicated to a procedure in the replicate database that has a different name from the primary procedure.

  Note: The replicated name you specify with the **pdb_setrepproc** command must match the name of a Replication Server function replication definition for the primary database connection. Replication Agent cannot validate the function replication definition, but if it does not exist, function replication from the primary database fails.
- If RASD is not initialized, the **pdb_setrepproc** command returns an error.

For Oracle:

To support stored procedure replication in Oracle, a stored procedure that is marked for replication must be modified. The modification is required to record the stored procedures execution in the Oracle transaction log. As a result of the modifications, consider this behavior when marking and unmarking stored procedures in Oracle:

- You must disable DDL replication before marking or unmarking a procedure, and re-enable it after marking or unmarking to prevent modifications from replicating to standby.
- Marking and unmarking a stored procedure for replication requires that Replication Agent drop, and then re-create the procedure. However, Replication Agent sets all the same privileges on the re-created procedure as those defined on the original procedure.

Note: Do not remove or alter Replication Agent comments in a marked stored procedure.

- When **pdb_setrepproc** is invoked to mark a procedure for replication, Replication Agent:
  - Modifies the user procedure to add code that captures input parameter values and generates Replication Agent transaction log records.
  - Generates a SQL script that creates the procedures required for the Replication Agent transaction log in the primary database.
  - Saves the generated script in a file called `partmark.sql` in the `RAX-15_5\inst_name\scripts\procname` directory, where `inst_name` is the name of the Replication Agent instance, and `procname` is the name of the stored procedure being marked. This script cannot be manually executed—it is for informational purposes only.

    Note: If the value of the **pdb_auto_run_scripts** configuration parameter is false, the `partmark.sql` script is saved but not executed automatically. You cannot manually run the script. To complete marking the procedure, you must first set **pdb_auto_run_scripts** to true, then re-run the **pdb_setrepproc** command.
  - Executes the script to mark the stored procedure and create the transaction log objects in the primary database (if the value of the **pdb_auto_run_scripts** configuration parameter is true).
  - After the script completes successfully, moves the `partmark.sql` file to the `RAX-15_5\inst_name\scripts\procname\installed` directory.
  - If the `mark` script fails, it is stored in a file (`partmark.sql`) in the `RAX-15_5\inst_name\scripts\procname` directory, the stored procedure is not marked, and transaction log objects are not created. You can examine the script by viewing the `mark.sql` file.
- When **pdb_setrepproc** is invoked to unmark a marked stored procedure, Replication Agent:
  - Modifies the user procedure to remove Replication Agent code that captures input parameter values and generates transaction log records.
  - Generates a SQL script that removes the tables and procedures required for the transaction log in the primary database.
  - Saves the generated script in a file called `partunmark.sql` in the `RAX-15_5\inst_name\scripts\procname` directory, where `inst_name` is the name of the Replication Agent instance and `procname` is the name of the stored procedure being unmarked. For Oracle, this script named `partunmark.sql` because it cannot be manually executed—it is for informational purposes only.

    Note: If the value of the **pdb_auto_run_scripts** configuration parameter is false, the `partunmark.sql` script is saved but not executed automatically. You cannot manually run the script. To complete unmarking the procedure, you must first set **pdb_auto_run_scripts** to true, then re-run the **pdb_setrepproc** command.

- Executes the script to unmark the stored procedure and remove the transaction log objects in the primary database (if the value of the **pdb_auto_run_scripts** configuration parameter is true).
- After the script completes successfully, moves the `partunmark.sql` file to the `RAX-15_5\inst_name\scripts\procname\installed` directory.
- If the `unmark` script fails, it is stored in a file (`partunmark.sql`) in the `RAX-15_5\inst_name\procname\scripts` directory and the stored procedure is not unmarked and the transaction log objects are not removed. You can examine the script by viewing the `partunmark.sql` file.

  When the `unmark` script execution encounters a fatal error on any database object, the **pdb_setrepproc** command returns this message:

  ```
  Could not unmark the following objects: ...
  See error log for details.
  ```
- The **pdb_setrepproc** command is used in replicating Oracle stored procedures that have an argument of type `Boolean`. See *Replication Agent Primary Database Guide > Replication Agent for Oracle > Stored Procedure Replication with BOOLEAN Arguments*.

See also

- *pdb_setrepcol* on page 25
- *pdb_setreptable* on page 47
- *ra_config* on page 70

pdb_setrepseq

Returns the sequence replication marking status; marks specified sequence for replication; unmarks all marked sequences or a specified sequence; enables or disables replication for all marked sequences or a specified sequence.

Syntax

```
pdb_setrepseq
  [ { sequence_name, [repname,] { mark
                                | unmark[, force]
                                | enable
                                | disable }
    | all, { mark
           | unmark[, force]
           | enable
           | disable }
  } ]
```

To return sequence replication marking status:

```
pdb_setrepseq [ { sequence_name|mark|unmark|enable|disable } ]
```

To unmark, enable, or disable all marked sequences:

```
pdb_setrepseq all, {unmark[, force] |enable|disable}
```

To mark, unmark, enable, or disable a specified sequence:

```
pdb_setrepseq sequence_name, {mark|unmark[, force] |enable|disable}
```

To mark a specified sequence for replication with a replicated name:

```
pdb_setrepseq sequence_name, repname, mark
```

Parameters

- **sequence_name –** The name of a user sequence in the primary database. The *sequence_name* option can be delimited with quote characters to specify the character case. If mixed character case (both uppercase and lowercase) is required, the name must be delimited. For example:

  ```
  "Sequence"
  ```

  The *sequence_name* parameter can be owner-qualified to include the primary sequence owner name, with each element separated by a period. For example:

  ```
  owner.sequence
  ```

  Note: If you must use an object name case that does not match the setting of the **ltl_character_case** parameter, the object name must be delimited. If an object name contains any non-alphanumeric characters, such as spaces and periods, it must be delimited with quote characters. For example, "sequence name" or owner."sequence name".

- **repname –** The replicated name of the sequence to be updated at the replicate site, if desired to be different than the sequence name at the primary site. The *repname* option can be delimited with quote characters to specify character case. See the previous description of the *sequence_name* parameter for details. By specifying a replicated name, sequence updates can be replicated to a sequence in the replicate database that has a different sequence name from the primary database. The *repname* option can be owner-qualified to include the replicate sequence owner name, with each element separated by a period. For example:

  ```
  repowner.repname
  ```

- **all –** A keyword that refers to all user sequences in the primary database. By using the all keyword, you can unmark all user sequences, or apply an enable or disable operation to all marked sequences.
- **mark –** A keyword that refers to marking user sequences for replication.
- **unmark –** A keyword that refers to unmarking user sequences for replication.
- **force –** A keyword that refers to the unmark operation. When the **force** keyword follows the **unmark** keyword, the **pdb_setrepseq** command immediately unmarks the specified sequence in the primary database, without first checking the enable status of the sequence. When the **force** keyword follows the **unmark** keyword and the **all** keyword, the **pdb_setrepseq** command immediately removes replication marking from all marked sequences in the primary database, regardless of their enable status
- **enable –** A keyword that refers to enabling replication for marked sequences.

- **disable –** A keyword that refers to disabling replication for marked sequences.

Usage

- When **pdb_setrepseq** is invoked, its function is determined by the keywords and options you specify.
- When multiple keywords and options are specified, each must be separated by a comma. Blank space before or after a comma is optional. For example:

```
pdb_setrepseq all, unmark, force
```

- When you specify a sequence in the **pdb_setrepseq** command, you must use the name of a valid user sequence.

pdb_setreptable

Returns replication marking status; marks all user tables or a specified table for replication; unmarks all marked tables or a specified table; or enables or disables replication for all marked tables or a specified table.

Syntax

```
pdb_setreptable
  [ { tablename[, [repname,] { mark [ { , immediate
                                     | , owner[, force] } ]
                               | unmark[, force]
                               | enable
                               | disable}]
    | all, { mark
           | unmark[, force]
           | enable
           | disable }
  } ]
```

To return replication marking status:

```
pdb_setreptable tablename
```

To mark all user tables:

```
pdb_setreptable all, mark
```

To unmark, enable, or disable all marked tables:

```
pdb_setreptable all, {unmark[, force]|enable|disable}
```

To mark, unmark, enable, or disable a specified table:

```
pdb_setreptable tablename, {mark[, owner][, force] |
 unmark[, force] |enable|disable}
```

To mark a specified table for replication with a replicated name:

```
pdb_setreptable tablename, repname, mark[, owner][, force]
```

To mark a specified table for replication immediately for any occurrence, that may or may not be marked:

```
pdb_setreptable tablename, mark[, immediate]
```

To return a list of all marked tables:

```
pdb_setreptable
```

Parameters

- **tablename –** The name of a valid user table in the primary database. Replication Agent returns complete marking information about the specified primary table. You cannot specify a system table, a view, or a Replication Agent transaction log table as a primary table.

  The *tablename* parameter can be owner-qualified to include the primary table owner name, with each element separated by a period. For example:

  ```
  owner.table
  ```

  This parameter can be delimited with quote characters to specify the character case.

  If mixed character case (both uppercase and lowercase) is required, the name must be delimited. For example:

  ```
  "Owner".table
  ```

  ```
  "Owner"."Table"
  ```

  Each mixed case element of the *tablename* option must be delimited separately, as shown in the previous example.

  If an object name contains any non-alphanumeric characters, such as spaces or periods, it must be delimited with quote characters. For example:

  ```
  "table name"
  ```

  ```
  owner."table name"
  ```

  If an object name contains a period, it must be both owner-qualified and delimited with quote characters. For example:

  ```
  owner."table.name"
  ```

  ```
  "table.owner"."table.name"
  ```

- **repname –** The name of the table specified in the replication definition for a primary table.

  Note: The replicated name you specify with the **pdb_setreptable** command must match a table name specified by a **with primary table named** clause in a Replication Server replication definition for the primary database connection. Replication Agent cannot validate the replication definition, but if it does not exist, or if the **with primary table named** clause does not match the replicated name specified with **pdb_setreptable**, replication from the primary table will fail.

  The *repname* option can be owner-qualified to include the replicate table owner name, with each element separated by a period. For example:

```
repowner.reptable
```

The *repname* option can also be delimited with quote characters to specify the character case. See the previous description of the *tablename* option for details.

Note: If the replicate table name contains a period (for example, *table.name*), without owner qualification, you must set the value of the Replication Agent **use_rssd** parameter to **true**.

- **all –** A keyword that refers to all tables in the primary database. By using the **all** keyword, you can mark all user tables, or apply an unmark, enable, or disable operation to all marked tables.
- **mark –** A keyword that refers to marking a table. Replication Agent returns a list of all marked tables in the primary database.
- **owner –** A keyword that refers to the mark operation.

  The owner keyword turns on the **SEND OWNER** mode. When you specify the owner of a table in a replication definition, you must always use the **owner** keyword if you want to enable the **SEND OWNER** mode.

  owner mode sets a flag in the LTL telling Replication Server that any table level Replication definition must be owner qualified to match this table.

  If the **owner** mode is set, the replication definition must be owner qualified. If the **owner** mode is not set, the replication definition must not be owner qualified.
- **unmark –** A keyword that refers to unmarking a marked table. Replication Agent returns a list of all unmarked tables in the primary database.
- **force –** A keyword used with the **unmark** operation or **mark** operation
  - When the **force** keyword follows the **unmark** keyword, the **pdb_setreptable** command immediately removes replication marking for the specified table in the primary database, without first checking the enable status of the table. When the **force** keyword follows the **unmark** keyword and the **all** keyword, **pdb_setreptable** immediately removes replication marking from all marked tables in the primary database, regardless of their enable status.

    The **force** keyword also forces complete execution of the unmarking script, even if errors occur during the unmarking process. Normally, when errors occur during script execution, the script terminates immediately without completing. The **force** keyword can be useful when a previous script execution failed and left the unmarking operation incomplete.

    When errors occur during a forced script execution, the **pdb_setreptable** command returns this message:

    ```
    Errors were encountered and ignored during FORCED script
    execution. See error log for details.
    ```
  - When the **force** keyword follows the **mark** keyword, the **pdb_setreptable** command allows a table that contains one or more columns with unsupported datatypes to be marked for replication. No data for the unsupported columns is sent to Replication Server. As a result, any replicate table must have a suitable default value defined for the

unsupported columns, since no data is received by the replicate database to be inserted into the unsupported columns.

The **force** keyword cannot be used in combination with the **all** keyword. Tables with unsupported datatypes must be individually marked using the **pdb_setreptable** command and the **force** keyword (they will never be automatically marked, or marked by default if they have columns with unsupported datatypes).

In addition, tables with unsupported datatypes are not automatically marked when the **pdb_automark_tables** configuration parameter is **true**. Tables with unsupported datatypes must be individually marked using the **pdb_setreptable** command and the **mark** and **force** keywords. For a list of supported and unsupported datatypes, see the *Replication Agent Primary Database Guide*.

Note: If a replication definition is created using the command **rs_create_repdef**, for a table that was marked using the **force** keyword, only columns with supported datatypes are listed in the replication definition. Any column with an unsupported datatype is excluded from the replication definition.

- **enable –** A keyword that refers to enabling replication for marked tables. Replication Agent returns a list of all marked tables in the primary database for which replication is enabled.
- **disable –** A keyword that refers to disabling replication for marked tables. Replication Agent returns a list of all marked tables in the primary database for which replication is disabled.
- **immediate –** A keyword that allows a table to be immediately marked for any occurrence.

Examples

- **Example 1 –**

```
pdb_setreptable authors
```

This command returns replication marking information for the table named “authors” in the primary database.

- **Example 2 –**

```
pdb_setreptable mark
```

This command returns replication marking information for all marked tables in the primary database.

- **Example 3 –**

```
pdb_setreptable disable
```

This command returns replication marking information for all marked tables for which replication has been disabled in the primary database.

- **Example 4 –**

```
pdb_setreptable all, unmark, force
```

This command forces unmarking for all marked tables in the primary database.

- **Example 5 –**

```
pdb_setreptable all, enable
```

This command enables replication for all marked tables in the primary database.

- **Example 6 –**

```
pdb_setreptable authors, mark
```

This command marks for replication the table named “authors” in the primary database. The primary table name in the replication definition must be **authors**.

- **Example 7 –**

```
pdb_setreptable authors, mark, owner
```

This command marks for replication the table named “authors” in the primary database so that the **OWNER_MODE** is enabled in the LTL. Therefore, any table replication definition created for this table must also be owner qualified.

- **Example 8 –**

```
pdb_setreptable ptable, rtable, mark, owner
```

The primary table name in the replication definition must be:

```
powner.rtable
```

- **Example 9 –**

```
pdb_setreptable ptable, rowner.rtable, mark,owner
```

The primary table name in the replication definition must be:

```
rowner.rtable
```

- **Example 10 –**

```
pdb_setreptable ptable, rowner.rtable, mark,owner
```

The primary table name in the replication definition must be:

```
rowner.rtable
```

- **Example 11 –**

```
pdb_setreptable authors, enable
```

This command enables replication for the marked table “authors” in the primary database.

- **Example 12 –**

```
pdb_setreptable table=mark, enable
```

This command enables replication for the marked table named “mark” in the primary database.

- **Example 13 –**

```
pdb_setreptable authors, unmark, force
```

  This command forces unmarking for the marked table “authors” in the primary database.

- **Example 14 –**

```
pdb_setreptable authors, mark, force
```

  This command forces table “authors” to be marked, even if it contains columns with unsupported datatypes. The columns with unsupported datatypes will not be replicated.

Usage

Using No Options

- When **pdb_setreptable** is invoked with no option, Replication Agent returns a list of all marked tables in the primary database.
- Tables marked for replication are listed in the marked objects table. All other user tables are considered unmarked.

  Note: The Replication Agent transaction log tables and shadow tables are not included in the list of unmarked tables. Also not included are any synonyms, views, or aliases of these database objects.

  For tables listed as unmarked or disabled, transactions will not be captured for replication.

Marking Tables

- When a table is marked for replication and the **owner** mode is set to on, the replication definition must contain the owner name in the **with primary table named** clause, or the **with all tables named** clause. If the **owner** mode setting and the existence of the owner name in the replication definition do not match, the replication definition is not used.

  For example:

  - Issuing **pdb_setreptable** with the **owner** mode set to on:

```
pdb_setreptable “mytable”, mark, owner
```

    causes the **rs_create_repdef** command to generate this replication definition for the primary and replicate database, which Replication Server expects to receive:

```
create replication definition ra$0xda_”mytable”
```

```
with primary at ora102.dco
with primary table named “qa4user”.”mytable”
with replicate table named “qa4user”.”mytable”
.
.
.
```

  - Issuing **pdb_setreptable** with the **owner** mode set to off:

```
pdb_setreptable “mytable”, mark
```

causes the **rs_create_repdef** command to generate this replication definition for the primary and replicate database, which Replication Server expects to receive:

```
create replication definition ra$0xda_”mytable”

with primary at ora102.dco
with primary table named ”mytable”

with replicate table named “qa4user”.”mytable”
.
.
.
```

- When a marked table is renamed or dropped and a new table with the original name is created, you must explicitly mark the new table because the new table has no marking-related information from the original table. The marking information is maintained internally by object ID, not table name.
- If you create a new table using a table name that was previously marked you must mark the new table by executing the **pdb_setreptable** command with the **mark** option, even if you did not **unmark** the previous table.

Unmarking Tables

- When you use the **unmark** keyword to remove replication marking from a primary table, Replication Agent verifies that replication is disabled for that table and checks to make sure that there are no pending (unprocessed) operations for that table in the transaction log. If replication is not disabled, or there is a pending operation for that table in the transaction log, **pdb_setreptable** returns an error.
- When you use the **unmark** keyword to remove replication marking from primary tables, you can also specify the **force** keyword to immediately remove replication marking from primary tables, without regard to whether replication is disabled.

Keywords

- When multiple keywords and options are specified, each must be separated by a comma. Blank space before or after a comma is optional. For example:

  ```
  pdb_setreptable all, unmark, force
  ```
- If a table name in the primary database is the same as a keyword, it can be identified by adding the **table=***string* to the beginning of the name. For example:

  ```
  pdb_setreptable table=unmark, mark
  ```

  This is true for both primary table names and replicated names.

Action Keywords

- When **pdb_setreptable** is invoked with a valid user table name, followed by an action keyword (**mark**, **unmark**, **enable**, or **disable**), the action specified is applied to the specified table.
- If **pdb_setreptable** is invoked with a table name and the **mark** keyword, it marks the specified table in the primary database for replication.

Note: When an individual table is marked, the owner filter list is not checked. This allows users to mark a table that has an owner in the owner filter list.

- If **pdb_setreptable** is invoked with a table name and the **unmark** keyword, it removes replication marking from the specified table in the primary database.

  You can specify the **force** keyword after the **unmark** keyword to force immediate unmarking of the specified table, to unmark a table for which replication is still enabled, or to force the script execution to ignore errors and continue an unmarking operation that failed previously.
- If **pdb_setreptable** is invoked with a table name and the **enable** keyword, it enables replication for the specified marked table in the primary database.
- If **pdb_setreptable** is invoked with a table name and the **disable** keyword, it disables replication for the specified marked table in the primary database.

  If the disable script execution encounters a fatal error on any database object, the **pdb_setreptable** command returns this message:

```
Could not disable the following objects: ...
See error log for details.
```

The all Keyword

- When **pdb_setreptable** is invoked with the **all** keyword and an action keyword (**mark**, **unmark**, **enable**, or **disable**), the action specified is applied to either all tables in the primary database, or all marked tables in the primary database.
- If **pdb_setreptable** is invoked with the **all** and **mark** keywords, all user tables in the primary database are marked for replication.

Note: Tables owned by users contained in the owner filter list will not be marked. However, you will be able to mark any individual table.

- If **pdb_setreptable** is invoked with the **all** and **unmark** keywords, it removes replication marking from all marked tables in the primary database.

  You can specify the **force** keyword after the **unmark** keyword to force immediate unmarking of all marked tables, or to unmark tables for which replication is still enabled, or to force the script execution to ignore errors and continue an unmarking operation that failed previously.
- If **pdb_setreptable** is invoked with the **all** and **enable** keywords, it enables replication for all marked tables in the primary database.
- If **pdb_setreptable** is invoked with the **all** and **disable** keywords, it disables replication for all marked tables in the primary database.

Replication Definition Types

- How you use the **pdb_setreptable** command depends on the type of replication definition that you have created at Replication Server. If you have created a database replication definition with no table replication definition, then the replicate procedure in the **pdb_setreptable** command refers to the table in the replicate database. However, if you have created a table replication definition, then the replicate table in the **pdb_setreptable**

command refers to the name of the table replication definition, and it is the table replication definition that must map to the table in the replicate database.

- If no table replication definition exists and will not be added prior to replication, but only a database replication definition exists, use these commands to mark a table for replication.
  - When the table in the replicate database has the same name as the table in the primary database, use:

    ```
    pdb_setreptable pdb_table, mark
    ```

    where *pdb_table* is the name of the table in the primary database that you want to mark for replication.
  - When the table in the replicate database has the different name than the table in the primary database, use:

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

    where *rep_table* is the name of the table in the replicate database.
- If a table replication definition exists or will be added prior to replication, regardless of whether or not a database replication definition exists, use these commands to mark a table for replication:
  - When the primary table in the table definition has the same name as the table in the primary database:

    ```
    pdb_setreptable pdb_table, mark
    ```

    If the table in the replicate database also has the same name as the table replication definition, then you can use the **with all tables named** clause in the replication definition in the primary Replication Server. For example:

    ```
    create replication definition my_table_repdef with primary at
    data_server.database
    with all tables named pdb_table …
    ```

    If the table in the replicate database has a different name than the primary table in the table replication definition, then the table replication definition must map to the table in the replicate database. For example:

    ```
    create replication definition my_table_repdef with primary at
    data_server.database
    with primary table named pdb_table
    with replicate table name rep_table …
    ```
  - When the name of the table replication definition is different than the table in the primary database, use:

    ```
    pdb_setreptable pdb_table, rdpri_table, mark
    ```

    where *rdpri_table* is the name of the primary table in the replication definition.
    If the table in the replicate database also has the same name as the primary table in the table replication definition, then you can use the **with all tables named** clause in the replication definition in the primary Replication Server. For example:

    ```
    create replication definition my_table_repdef
    with primary at data_server.database
    with all tables named rdpri_table …
    ```

If the table in the replicate database has a different name from the primary table in the table replication definition, then the table replication definition must map to the table in the replicate database. For example:

```
create replication definition my_table_repdef
with primary at data_server.database

with primary table named rdpri_table
with replicate table name rep_table …
```

Replicated Names and the owner Keyword

- When **pdb_setreptable** is invoked with a primary table name and a replicated name, followed by the **mark** keyword, the primary table is marked for replication with the specified replicated name.

  By specifying a replicated name, transactions can be replicated to a table in the replicate database that has a different name from the primary table.

  Note: The replicated name you specify with the **pdb_setreptable** command must match a table name specified by a **with all tables named** clause in a Replication Server replication definition for the primary database connection. Replication Agent cannot validate the replication definition, but if it does not exist, or if the **with all tables named** clause does not match the replicated name specified with **pdb_setreptable**, replication from the primary table will fail.

- You can also specify the **owner** keyword after the **mark** keyword so that when operations against the primary table are replicated, the primary table owner name will be attached to the replicate table name in the form `owner.tablename`.

  Note: If you want to use an owner-qualified replicate table name with the replicate owner's name, use the **owner** keyword with the **pdb_setreptable** command. If you specify an unqualified replicate table name, the primary table owner name is sent with the replicate table name in the LTL.

Unsupported Datatypes

- If a table contains a column with a datatype that is not supported for replication, the **pdb_setreptable** command using the **mark** keyword may fail with an error similar to:

  ```
  Command <pdb_setreptable> failed - Table <MYTABLE> could not be
  marked because:The table contains an
  unsupported data type.
  ```

  To force the table to be marked, excluding the unsupported datatype columns from replication, add the **force** keyword to the **pdb_setreptable** command.

Errors

- If the Replication Agent transaction log does not exist in the RASD is not initialized, the **pdb_setreptable** command returns an error.

- If the table name you specify does not exist in the primary database, the **pdb_setreptable** command returns an error.
- If the primary table name you specify does not exist in the primary database, the **pdb_setreptable** command returns an error.
- If the enable script execution encounters a fatal error on any database object, the **pdb_setreptable** command returns this message:

```
Could not enable the following objects: ...
See error log for details.
```

Spaces in column names

- To replicate a table that contains column names that have spaces, you must set **structured_tokens** to **true**.

Aliases

- If you specify an alias or synonym as a primary table in the **pdb_setreptable** command, the actual table that the alias or synonym refers to is acted upon. The actual table name is the table name sent to the primary Replication Server.

See also

- *pdb_setrepcol* on page 25
- *pdb_setrepproc* on page 37
- *ra_config* on page 70

pdb_skip_op

Returns, adds to, or removes operations from a list of operations to skip during processing. The format of the record locator is database-specific.

Syntax

```
pdb_skip_op [
 {
   add , { locator | { scn, thread, rba | lsn } }
 |
   remove, { all | locator | { scn, thread, rba | lsn } }
 }
]
```

Parameters

- **add –** Adds a specified ID to the list of identifiers of records to skip.
- **remove –** Removes a specified ID from the list of identifiers to skip.
- **locator –** The locator keyword from the list of LTM locators to identify the operations to skip.
- **scn –** The system change number (SCN) keyword identifies a specified log record to skip.

- **thread –** The thread keyword of the redo log thread of the operation to skip.
- **rba –** The record byte address (RBA) keyword of the log record to skip.
- **all –** Allows you to add or remove all IDs in the list of identifiers to skip.

Examples

- **Example 1 –**

  ```
  pdb_skip_op
  ```

  This command with no parameters returns a list of the identifiers for the records you want to skip.

- **Example 2 –**

  ```
  pdb_skip_op add, id
  ```

  This command adds an ID to the list of identifiers you want to skip.

- **Example 3 –**

  ```
  pdb_skip_op remove, id
  ```

  This command removes an ID from the list of identifiers you want to skip.

- **Example 4 –**

  ```
  pdb_skip_op remove, all
  ```

  This command removes all the IDs on the list of identifiers you want to skip.

- **Example 5 –**

  ```
  pdb_skip_op add, locator
  ```

  This command adds an operation, referred to by its location, to the list of identifiers that you want to skip.

Usage

- The **pdb_skip_op** command allows you to skip problem records, thereby avoiding having to reinitialize Replication Agent.
- Skipped records are written to the system log as a warning message.
- The **pdb_skip_op** command is valid when Replication Agent is in Admin or Replication Down state.
- The format of the identifier is database specific:
  - For Oracle, the identifier contains the system change number (SCN), redo log thread, and record byte address (RBA). It has the following form:

    ```
    wrap.base.sub, thread, lsn.blknum.blkoffset
    ```

    where:

- **wrap** is the SCN wrap number.
- **base** is the SCN base number.
- **sub** is the SCN subindex.
- **thread** is the redo log thread number that the operation occurred on.
- **lsn** is the RBA log sequence number.
- **blknum** is the RBA block number.
- **blkoffset** is the RBA offset into the block where this record resides.

The values must be specified in the **pdb_skip_op** command together, as shown above, enclosed in quotes, with each item separated by a period.

For example:

```
'0000.012345678.00', '1', '0012.0000444.0000123'
```

All values can be described as hexadecimal by prefixing the identifier with an "0x" as follows:

```
'0x000c.00001bc.000007b'
```

or:

```
'0x000.00BC614E'
```

See also

- *ra_helplocator* on page 89
- *ra_locator* on page 95

pdb_thread_filter

(Oracle RAC only) Filters all activity on an Oracle instance redo log thread or threads during replication, and displays a list of threads being filtered.

Syntax

```
pdb_thread_filter [
   add, thread_id
 |
   remove, { thread_id | all }
]
```

Parameters

- **add –** adds the specified thread ID to the list of threads being filtered.
- **remove –** removes the specified thread ID or all thread IDs from the list of threads being filtered.
- ***thread_id* –** is the thread ID to add to or remove from the list of threads being filtered.
- **all –** removes all threads from the list of threads being filtered.

Examples

- **Example 1 –**

  ```
  pdb_thread_filter add, 1
  ```

  Filters activity on thread 1.

- **Example 2 –**

  ```
  pdb_thread_filter remove, 1
  ```

  Removes thread 1 from the list of threads being filtered.

- **Example 3 –**

  ```
  pdb_thread_filter remove, all
  ```

  Removes all threads from the list of threads being filtered.

- **Example 4 –**

  ```
  pdb_thread_filter
  ```

  Displays a list of threads being filtered.

Usage

- Incorrect use of **pdb_thread_filter** may result in loss of data.
- Use **pdb_thread_filter** only under the direction of Sybase Technical Support when a thread being filtered may not be running.

pdb_truncate_xlog

Truncates the Replication Agent primary database transaction log on demand.

- The behavior of **pdb_truncate_xlog** changes based on the value of the configuration parameter **pdb_include_archives**:
  - When **pdb_include_archives** is **false**, triggers the archive process to archive any online redo logs that have already been processed by Replication Agent.
  - When **pdb_include_archives** is **true**, removes old archive redo log files from the path specified by **pdb_archive_path**.

Note: Truncation of the old archive log files from the **pdb_archive_path** directory is performed only if the **pdb_archive_remove** parameter is set to **true**.

For more information on how Replication Agent affects each type of database when **pdb_truncate_xlog** is executed, see the *Replication Agent Primary Database Guide*.

Syntax

```
pdb_truncate_xlog
```

Usage

- When **pdb_truncate_xlog** is invoked, Replication Agent immediately truncates the primary database transaction log based on the most recent truncation point received from the primary Replication Server. The truncation point is part of the information contained in the LTM Locator.
- To update the LTM Locator from the primary Replication Server, use the **ra_locator** command.
- The **pdb_truncate_xlog** command is asynchronous and it does not return success or failure (unless an immediate error occurs). You must examine the Replication Agent system log to determine success or failure of the **pdb_truncate_xlog** command.
- If the Replication Agent primary database log does not exist or if a connection failure occurs, the **pdb_truncate_xlog** command returns an error message.
- You can use the **ra_config** command to specify the type of automatic truncation you want. You can use the **pdb_truncate_xlog** command to truncate the transaction log if automatic truncation is not sufficient to manage the size of the transaction log.
- The **pdb_truncate_xlog** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_config* on page 70
- *ra_locator* on page 95

pdb_version

Returns the type and version of the primary data server.

Syntax

```
pdb_version
```

Usage

The actual results returned vary depending on the type of primary data server.

See also

- *ra_version* on page 117
- *ra_version_all* on page 118

pdb_xlog

Returns the names of Replication Agent system objects; creates Replication Agent system objects in the primary database; or removes Replication Agent system objects from the primary database.

Note: Use **ra_admin** and **ra_locator** instead of **pdb_xlog**, which has been deprecated.

For Oracle, **pdb_xlog** verifies permissions are valid for Replication Agent to obtain system data from the primary database. It also checks the condition of the primary database to determine if archiving is turned on or off, and then loads the RASD with system data from the primary database.

Syntax

```
pdb_xlog [{ init | create | remove } [, force ] | move_truncpt ]
```

Parameters

- **init –** the keyword for creating Replication Agent system objects in the primary database.
- **create –** the keyword for creating a transaction log. Deprecated; use the **init** keyword instead.
- **remove –** the keyword for removing a transaction log.
- **force –** a keyword that refers to the **remove** or the **init** operation.
- **move_truncpt –** a keyword that moves the truncation point.

Examples

- **Example 1 –**

  ```
  pdb_xlog init
  ```

  This command initializes Replication Agent, creating any required transaction log base components.

  ```
  pdb_xlog init, force
  ```

  This command re-initializes Replication Agent, creating or re-loading any required transaction log base components.

  ```
  pdb_xlog remove
  ```

  This command removes any Replication Agent transaction log base components.

  ```
  pdb_xlog remove, force
  ```

  This command removes any Replication Agent transaction log base components and ignores any individual errors that occur during removal.

  ```
  pdb_xlog move_truncpt
  ```

  This command moves the transaction log truncation point to the end of the current transaction log.

Usage

- When you invoke **pdb_xlog** with no option, it returns the actual names (not synonyms or aliases) of all Replication Agent system objects in the primary database. If you have

initialized Replication Agent, it returns the name of the component and the primary database instance name.

See the section for your specific primary data server in the *Replication Agent Primary Database Guide* for more information on Replication Agent object names.

- If you invoke **pdb_xlog** with no option, and the Replication Agent system objects do not exist in the primary database, or the RASD has not been initialized, the command returns no information.
- If you invoke **pdb_xlog** with the **init** keyword, the truncation point is established at the end of the primary database transaction log.
- If you invoke **pdb_xlog** with the **init**, **force** keywords, the truncation point is moved to the end of the log if Replication Agent is not already initialized. However, if Replication Agent is already initialized, the truncation point is not moved.

  Note: Use **pdb_xlog init** with the **force** keyword only when advised by Sybase Technical support.
- If you invoke **pdb_xlog** with the **move_truncpt** keyword, the truncation point is moved to the end of the log without change or modification to any Replication Agent components. (for Oracle, this is the end of the current online redo log.) The **move_truncpt** option has no effect if Replication Agent has not been initialized.

  Note: To prevent Replication Server from requesting a log starting point that occurs earlier in the log than the location established by the **move_truncpt** option, Replication Server's LTM locator value for the primary connection must be zeroed. Execute Replication Server System Database (RSSD) command **rs_zeroltm** against the primary database connection to zero the LTM locator.

  If you move the secondary truncation point to the end of the primary database transaction log using **pdb_xlog move_truncpt**, you risk skipping over any DDL commands record in the log. The DDL commands might have been used by Replication Agent to update information stored within the Replication Agent System Database (RASD). If the RASD contents are incorrect due to skipping processing of some log records, you may force all of the schema information in the RASD to be refreshed using command **pdb_xlog init, force**. If only the schema for a single object stored in the RASD is of concern, you can unmark and remark just that single object, which forces the schema of the object to be reread into the RASD.
- When you invoke **pdb_xlog** with the **init** keyword, Replication Agent:
  - Generates a SQL script that creates the Replication Agent tables and procedures in the primary database.
  - Saves the generated script in a file called `partinit.sql` in the `RAX-15_5\inst_name\scripts\xlog` directory, where `inst_name` is the name of the Replication Agent instance.

    Note: If the value of the **pdb_auto_run_scripts** configuration parameter is false, the `partinit.sql` script is saved but not executed. However, you cannot manually run

the script. To complete initializing Replication Agent, first set **pdb_auto_run_scripts** to true, and then re-run the **pdb_xlog init** command.

  - Executes the script to create the Replication Agent system objects in the primary database (if the value of the **pdb_auto_run_scripts** configuration parameter is true).
  - After the script completes successfully, moves the `partinit.sql` file to the `RAX-15_5\inst_name\scripts\xlog\installed` directory.
  - If the create script fails, it is stored in a file (`partinit.sql`) in the `RAX-15_5\inst_name\scripts\xlog` directory and the transaction log is not created. You can examine the script by viewing the `partinit.sql` file.
- If you invoke **pdb_xlog** with the **init** keyword and the Replication Agent objects already exist in the primary database or the RASD has been initialized, **pdb_xlog** returns an error message.
- When you invoke **pdb_xlog** with the **remove** keyword, Replication Agent:
  - Generates a SQL script that deletes the tables and procedures required for the Replication Agent system objects in the primary database.
  - Saves the generated script in a file called `partdeinit.sql` in the `RAX-15_5\inst_name\scripts\xlog` directory, where `inst_name` is the name of the Replication Agent instance.

    Note: If the value of the **pdb_auto_run_scripts** configuration parameter is false, the `partdeinit.sql` script is saved but not executed automatically. You cannot manually run the script. To complete deinitializing Replication Agent, first set **pdb_auto_run_scripts** to true, then re-run the **pdb_xlog remove** command.

  - Executes the script to delete the Replication Agent objects from the primary database (if the value of the **pdb_auto_run_scripts** configuration parameter is true).
  - After the script completes successfully, moves the `partdeinit.sql` file to the `RAX-15_5\inst_name\scripts\xlog\installed` directory.
  - If the script fails, it is stored in a file (`partdeinit.sql`) in the `RAX-15_5\inst_name\scripts\xlog` directory and the Replication Agent objects are not deleted from the primary database. You can examine the script by viewing the `partdeinit.sql` file.
- When you invoke **pdb_xlog** with the **remove** keyword followed by the **force** keyword, the `partdeinit.sql` script continues executing, even if errors occur. The **force** keyword may be useful when a previous remove operation failed and the `partdeinit.sql` script terminated with an error.
- If you invoke **pdb_xlog** with the **remove** keyword, and Replication Agent objects do not exist in the primary database or the RASD has not been initialized, **pdb_xlog** returns an error message.
- If you invoke **pdb_xlog** with the **remove** keyword and any objects in the primary database are still marked for replication, **pdb_xlog** returns an error message.
  You can use the **pdb_setrepproc** and **pdb_setreptable** commands to determine which stored procedures and tables in the primary database are still marked. You also can use the **pdb_setrepddl** command to determine if DDL is enabled.

Even if objects are marked in the primary database, you can use **pdb_xlog** with the **remove** keyword followed by the **force** keyword to unmark any marked objects, and then remove the transaction log objects.

- If you invoke **pdb_xlog** with no option, the command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down states.
- If you invoke **pdb_xlog** with either the **init** or **remove** keyword, the command is valid only when the Replication Agent instance is in the Admin or Replication Down state.
- The **pdb_xlog init** command verifies that these privileges have been granted to **pds_username**:
  - **EXECUTE_CATALOG_ROLE**
  - **SELECT ON V_$LOGMNR_CONTENTS**
  - **SELECT ON V_$LOGMNR_LOGS**

  These privileges are necessary for the **ra_dumptran** and **ra_helpop** commands to function properly. These privileges are not required for replication, only for using the **ra_dumptran** and **ra_helpop** commands, which are used in debugging and troubleshooting. If these privileges have not been granted at the time you invoke **pdb_xlog init**, a warning message is returned and logged in the Replication Agent log file.
- For more information about the Replication Agent transaction log, see the section for your specific primary data server in the *Replication Agent Primary Database Guide*.

See also

- *pdb_setrepcol* on page 25
- *pdb_setrepproc* on page 37
- *pdb_setreptable* on page 47
- *ra_admin* on page 66
- *ra_locator* on page 95

quiesce

Stops all Replication Agent processing in Replicating state, and puts the Replication Agent instance in Admin state.

Syntax

```
quiesce
```

Usage

- When the **quiesce** command is invoked, it stops all current replication processing in the Replication Agent instance:
  - The Log Reader component stops reading operations from the transaction log when the scan reaches the end of the log. It continues to send change-set data to the Log Transfer Interface component until it finishes processing the last operation scanned.

  - The Log Transfer Interface component stops sending LTL commands to Replication Server as soon as it finishes processing the last change set it receives from the Log Reader.
  - When the Log Transfer Interface component is finished processing its input queue and sending the resulting LTL, the Replication Agent instance releases all of its connections to the primary database, and drops its connection to the primary Replication Server (and RSSD, if connected).
  - The Replication Agent instance goes from Replicating state to Admin state.
- If the Replication Agent internal queues are full when the **quiesce** command is invoked, the quiesce processing may take a while to complete, and there may be a delay before the Replication Agent instance completes its transition to Admin state.
- Before moving Replication Agent to the Admin state, the **quiesce** command waits until all data in the primary log has been read and sent to Replication Server.
- If the Replication Agent instance is in Admin state, the **quiesce** command returns an error.
- The **quiesce** command is valid only when the Replication Agent instance is in Replicating state.

Note: The action of the **suspend** command is similar to that of the **quiesce** command, except that the **suspend** command stops Replication Agent processing immediately and flushes all data in the internal queues.

See also

- *ra_status* on page 112
- *resume* on page 124
- *shutdown* on page 130
- *suspend* on page 131

ra_admin

Returns the names of Replication Agent system objects; creates them in the primary database, or removes them from the primary database.

ra_admin verifies that permissions are valid for Replication Agent to obtain system data from the primary database. **ra_admin** also checks the condition of the primary database to determine whether archiving is turned on, and then loads the RASD with system data from the primary database.

The **ra_admin** command verifies that permissions are valid for Replication Agent to obtain system data from the primary database. The command also checks the condition of the primary database to determine whether archiving is turned on, and then loads the RASD with system data from the primary database.

Note: Use **ra_admin** and **ra_locator** instead of **pdb_xlog**, which has been deprecated.

Syntax

```
ra_admin [ { init | refresh | deinit[, force] } ]
```

Parameters

- **init –** the keyword for creating Replication Agent system objects in the primary database.
- **refresh –** a keyword for reinitializing Replication Agent, creating, or reloading any required transaction log base components.
- **deinit –** the keyword for removing a transaction log.
- **force –** a keyword that refers to the **deinit** operation.

Examples

- **Example 1 –**

  ```
  ra_admin init
  ```

  This command initializes Replication Agent, creating any required transaction log base components.

- **Example 2 –**

  ```
  ra_admin refresh
  ```

  This command reinitializes Replication Agent, creating or reloading any required transaction log base components.

- **Example 3 –**

  ```
  ra_admin deinit
  ```

  This command removes any Replication Agent transaction log base components.

- **Example 4 –**

  ```
  ra_admin deinit, force
  ```

  This command removes any Replication Agent transaction log base components and ignores any individual errors that occur during removal.

Usage

- When you invoke **ra_admin** with no option, it returns the actual names (not synonyms or aliases) of all Replication Agent system objects in the primary database. If you have initialized Replication Agent, it returns the name of the component and the primary database instance name.

  When you invoke **ra_admin** with no option, it returns the actual names (not synonyms or aliases) of all Replication Agent system objects in the primary database. If you have initialized Replication Agent, it returns the name of the component and the primary database instance name.

See the section for your specific primary data server in the *Replication Agent Primary Database Guide* for more information on Replication Agent object names.

- If you invoke **ra_admin** with no option and the Replication Agent system objects do not exist in the primary database, or the RASD has not been initialized, the command returns no information.
- If you invoke **ra_admin** with the **init** keyword, the truncation point is moved to the end of the primary database transaction log.
- When you invoke **ra_admin** with the **init** keyword, Replication Agent:
  - Generates a SQL script that creates the Replication Agent tables and procedures in the primary database.
  - Saves the generated script in a file called `partinit.sql` in the `RAX-15_5\`*`inst_name`*`\scripts\xlog` directory, where *inst_name* is the name of the Replication Agent instance.

    Note: If the value of the **pdb_auto_run_scripts** configuration parameter is false, the `partinit.sql` script is saved but not executed. However, you cannot manually run the script. To complete initializing Replication Agent, first set **pdb_auto_run_scripts** to true, then re-run the **ra_admin init** command.
  - Executes the script to create the Replication Agent system objects in the primary database (if the value of the **pdb_auto_run_scripts** configuration parameter is true).
  - After the script completes successfully, moves the `partinit.sql` file to the `RAX-15_5\`*`inst_name`*`\scripts\xlog\installed` directory.
  - If the create script fails, it is stored in a file (`partinit.sql`) in the `RAX-15_5\`*`inst_name`*`\scripts\xlog` directory and the transaction log is not created. You can examine the script by viewing the `partinit.sql` file.
- If you invoke **ra_admin** with the **init** keyword and the Replication Agent objects already exist in the primary database or the RASD has been initialized, **ra_admin** returns an error message.
- If you invoke **ra_admin** with the **refresh** keyword, the truncation point is moved to the end of the log if Replication Agent is not already initialized. However, if Replication Agent is already initialized, the truncation point is not moved.

  Note: Use **ra_admin refresh** only when advised by Sybase Technical support.
- When you invoke **ra_admin** with the **deinit** keyword, Replication Agent:
  - Generates a SQL script that deletes the tables and procedures required for the system objects in the primary database.
  - Saves the generated script in a file called `partdeinit.sql` in the `RAX-15_5\`*`inst_name`*`\scripts\xlog` directory, where *inst_name* is the name of the Replication Agent instance.

    Note: If the value of the **pdb_auto_run_scripts** configuration parameter is false, the `partdeinit.sql` script is saved but not executed automatically. You cannot

manually run the script. To complete deinitializing Replication Agent, first set **pdb_auto_run_scripts** to true, then re-run the **ra_admin deinit** command.

- Executes the script to delete the Replication Agent objects from the primary database (if the value of the **pdb_auto_run_scripts** configuration parameter is true).
- After the script completes successfully, moves the `partdeinit.sql` file to the `RAX-15_5\`*`inst_name`*`\scripts\xlog\installed` directory.
- If the script fails, it is stored in a file (`partdeinit.sql`) in the `RAX-15_5\`*`inst_name`*`\scripts\xlog` directory and the Replication Agent objects are not deleted from the primary database. You can examine the script by viewing the `partdeinit.sql` file.

- When you invoke **ra_admin** with the **deinit** keyword followed by the **force** keyword, the `partdeinit.sql` script continues executing, even if errors occur. The **force** keyword may be useful when a previous remove operation failed and the `partdeinit.sql` script terminated with an error.
- If you invoke **ra_admin** with the **deinit** keyword, and Replication Agent objects do not exist in the primary database or the RASD has not been initialized, **ra_admin** returns an error message.
- If you invoke **ra_admin** with the **deinit** keyword and any objects in the primary database are still marked for replication, **ra_admin** returns an error message.

  You can use the **pdb_setrepproc** and **pdb_setreptable** commands to determine which stored procedures and tables in the primary database are still marked. You also can use the **pdb_setrepddl** command to determine if DDL is enabled.

  Even if objects are marked in the primary database, you can use **ra_admin** with the **deinit** keyword followed by the **force** keyword to unmark any marked objects, and then remove the transaction log objects.
- If you invoke **ra_admin** with no option, the command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down states.
- If you invoke **ra_admin** with either the **init** or **deinit** keyword, the command is valid only when the Replication Agent instance is in the Admin or Replication Down state.
- The **ra_admin init** command verifies that these privileges have been granted to **pds_username**:
  - **EXECUTE_CATALOG_ROLE**
  - **SELECT ON V_$LOGMNR_CONTENTS**
  - **SELECT ON V_$LOGMNR_LOGS**

  These privileges are necessary for the **ra_dumptran** and **ra_helpop** commands to function properly. These privileges are not required for replication, only for using the **ra_dumptran** and **ra_helpop** commands, which are used in debugging and troubleshooting. If these privileges have not been granted at the time you invoke **ra_admin init**, a warning message is returned and logged in the Replication Agent log file.
- For more information about the Replication Agent transaction log, see the section for your specific primary data server in the *Replication Agent Primary Database Guide*.

See also

- *pdb_setrepcol* on page 25
- *pdb_setrepproc* on page 37
- *pdb_setreptable* on page 47
- *ra_admin* on page 66
- *ra_locator* on page 95
- *pdb_xlog* on page 61

ra_config

Returns help information for Replication Agent configuration parameters, or sets the value of a specified configuration parameter.

Syntax

```
ra_config [ { param [, value] | password_parameter, [value] } ]
```

Parameters

- **param –** The name of a Replication Agent configuration parameter.
- **value –** The value to be assigned to the configuration parameter specified in the **param** option. You can use the keyword **default** to set the specified parameter to its default value.
- **password_parameter –** Affects password security parameters.

Table 3. Password Parameters

password_parameter	Description and *value*	Default Value
min_password_len	Minimum number of characters required. Range – 6 to 12.	-1 (no minimum length)
max_password_len	Maximum number of characters. Always set **max_password_len** to a value greater than **min_password_len**. Range – 13 to 255.	255 characters
password_lowercase_required	Whether lowercase characters are required. • True – required. • False – not required.	False

password_parameter	Description and *value*	Default Value
password_uppercase_required	Whether uppercase characters are required. • True – required. • False – not required.	False
password_numeric_required	Whether a numeric character is required. • True – required. • False – not required.	False
password_special_required	Whether a special character is required. • True – required. • False - not required.	False
password_expiration	Number of days after which the password expires. • 0 – password never expires (default). • Range – 0 to 32,767. If the password has expired, the account is locked and the Replication Agent notifies the user that the password has expired. The account is unlocked only after a password that complies with the password security requirements is set. An administrator can override the initial password expiration value that is set during the upgrade process.	0 (password never expires)

Examples

- **Example 1 –**

```
ra_config
```

  When **ra_config** is issued with no parameters, it returns a list of all Replication Agent configuration parameters.

- **Example 2 –**

```
ra_config use_rssd
```

This command returns the current value of the **use_rssd** configuration parameter.

- **Example 3 –**

```
ra_config scan_sleep_max, 60
```

This command changes the value of the **scan_sleep_max** parameter to 60.

- **Example 4 –**

```
ra_config max_password_len, 15
```

This command changes the value of the **max_password_len** parameter to 15 for all users.

Usage

- If **ra_config** is invoked with no option, it returns a list of all Replication Agent configuration parameters.
- If **ra_config** is invoked with the **param** option, it returns information only for the specified configuration parameter.
- If **ra_config** is invoked with the **param** and **value** options, it changes the setting of the specified configuration parameter to the value specified in the **value** option.
- You can use the keyword **default** in place of the **value** option to reset a configuration parameter to its default value. For example:

```
ra_config use_rssd, default
```

- The following information is returned for each configuration parameter:
  - Parameter name – the name of the parameter.
  - Parameter type – the datatype of the parameter value (for example, string, numeric, or Boolean).
  - Current value – the value of the parameter in effect at the time **ra_config** is invoked.
  - Pending value – if different from the current value, the value to which the parameter was set by a previous invocation of the **ra_config** command, but which has not yet taken effect.
  - Default value – the value of the parameter when the Replication Agent instance configuration file is created.
  - Legal values – the values that are allowed for the parameter, for example, a range of numbers or a list of specific strings.
  - Category – refers to the Replication Agent component affected by the value of the parameter.
  - Restart – refers to parameters that require the Replication Agent instance to be shut down and restarted before a change in value takes effect.
- When **ra_config** is invoked with either no option, or only the **param** option, the command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

- If **ra_config** is invoked when the Replication Agent instance is in Replicating state, with the **param** and **value** options for a parameter that can be changed only in Admin state, it returns an error.
- When **ra_config** is invoked with the **param** and **value** options, the command is always valid when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *ra_set_login* on page 103
- *ra_help* on page 82

ra_date

Returns the current date and time from the Replication Agent instance.

Syntax

```
ra_date
```

Usage

- When **ra_date** is invoked, it returns the current date and time from the Replication Agent instance in the form of a Sybase `datetime` datatype, as follows:

```
Current RA Date
 ------------------------------
     Jan 11 2010 12:09:47.310
(1 row affected)
```

- The **ra_date** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *pdb_date* on page 10
- *ra_config* on page 70

ra_downgrade

The **ra_downgrade** command prepares Replication Agent to downgrade to an earlier version.

The **ra_downgrade** command copies the RASD contents to an export file. This file is then applied by the instance to which Replication Agent is being downgraded (the earlier version) to complete the downgrade.

Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands have been deprecated. Use the **ra_downgrade** and **ra_migrate** commands where possible. See the *Replication Agent Primary Database Guide*.

Syntax

```
ra_downgrade [list]
```

Parameters

- **list –** Displays the name and location of the file to which RASD content was exported.

Usage

- To prepare for downgrading to an earlier version of Replication Agent, **ra_downgrade** changes Replication Agent system objects in the primary database to match those of the instance to which Replication Agent is being downgraded.
- The **ra_downgrade** command extracts the contents of the Replication Agent System Database (RASD) to a file named *timestamp*.`export`, where *timestamp* is a timestamp taken at the moment **ra_downgrade** was invoked. By default, this file is located in the `import` subdirectory under the directory specified by the **rasd_backup_dir** configuration parameter of the Replication Agent instance to which you are downgrading (the earlier version). The path to this file is returned if **ra_downgrade** executes successfully.
- The **ra_downgrade** command is valid when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *ra_downgrade_accept* on page 74
- *ra_downgrade_prepare* on page 75
- *ra_migrate* on page 99

ra_downgrade_accept

The **ra_downgrade_accept** command is executed by the instance to which Replication Agent is being downgraded (the earlier version).

This command completes the downgrade process started by the **ra_downgrade_prepare** command, which is executed by the instance from which Replication Agent is being downgraded (the later version).

Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands have been deprecated. Use the **ra_downgrade** and **ra_migrate** commands where possible. See the *Replication Agent Primary Database Guide*.

Syntax

```
ra_downgrade_accept export_file
```

Parameters

- **export_file –** The file name to which the RASD was extracted during execution of the **ra_downgrade_prepare** command. This can be the file name alone or the file name with its absolute path. The file is named *timestamp*.`export`, where *timestamp* is a timestamp taken at the moment **ra_downgrade_prepare** was invoked.

Usage

- The **ra_downgrade_accept** command restores the RASD from the specified file. By default, this file is located in the `import` subdirectory under the directory specified by the **rasd_backup_dir** configuration parameter of the Replication Agent instance to which you are downgrading (the earlier version). The path to this file was returned if **ra_downgrade_prepare** executed successfully.
- To use the **ra_downgrade_accept** command, Replication Agent must be able to connect to the primary database.
- If the **ra_downgrade_accept** command executes successfully, Replication Agent shuts down.
- The **ra_downgrade_accept** command is valid when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *ra_downgrade* on page 73
- *ra_downgrade_prepare* on page 75
- *ra_migrate* on page 99

ra_downgrade_prepare

The **ra_downgrade_prepare** command prepares Replication Agent to downgrade to an earlier version.

When executed in Replication Agent for Oracle, the **ra_downgrade_prepare** command copies the RASD contents to an export file. This file is then applied by the instance to which Replication Agent is being downgraded (the earlier version) to complete the downgrade.

Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands have been deprecated. Use the **ra_downgrade** and **ra_migrate** commands where possible. See the *Replication Agent Primary Database Guide*.

Syntax

```
ra_downgrade_prepare { list | target_inst_path }
```

Parameters

- **list –** Keyword to list the RASD tables that can be exported.
- **target_inst_path –** The absolute path of the Replication Agent instance to which you are downgrading (the earlier version).

Usage

- To prepare for downgrading to an earlier version of Replication Agent, the **ra_downgrade_prepare** command first changes Replication Agent system objects in the primary database to match those of the instance to which Replication Agent is being downgraded.
- The **ra_downgrade_prepare** command also extracts the contents of the Replication Agent System Database (RASD) to a file named *timestamp*`.export`, where *timestamp* is a timestamp taken at the moment **ra_downgrade_prepare** was invoked. By default, this file is located in the `import` subdirectory under the directory specified by the **rasd_backup_dir** configuration parameter of the Replication Agent instance to which you are downgrading (the earlier version). The path to this file is returned if **ra_downgrade_prepare** executes successfully.
- The **ra_downgrade_prepare** command is valid when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *ra_downgrade* on page 73
- *ra_downgrade_accept* on page 74
- *ra_migrate* on page 99

ra_dump

Emulates the Replication Server **rs_dumpdb** and **rs_dumptran** system functions.

Syntax

```
ra_dump { database | transaction } , dbname, dump_label
```

Parameters

- **database –** A keyword that causes the primary Replication Server to apply the function string associated with the **rs_dumpdb** system function.
- **transaction –** A keyword that causes the primary Replication Server to apply the function string associated with the **rs_dumptran** system function.
- **dbname –** The name of the database to be dumped.
- **dump_label –** A `varchar(30)` value that contains information to identify the database dump.

Examples

- **Example 1 –**

```
ra_dump database, MSSQL_source, rssddmp
```

  Here, MSSQL_source is the database name, and rssddmp is the dump label.

Usage

- When **ra_dump** is invoked, Replication Agent places a **dump** marker in the Replication Agent transaction log to facilitate coordinated dumps.
- The **ra_dump** command returns an error message if the transaction log does not exist.
- The **ra_dump** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.
- For more information about the Replication Server **rs_dumpdb** and **rs_dumptran** system functions, see the *Replication Agent Administration Guide* and *Replication Agent Primary Database Guide*.

See also

- *ra_config* on page 70
- *ra_migrate* on page 99

ra_dumptran

This command returns information for use in troubleshooting a specified database transaction.

Syntax

```
ra_dumptran "{ opid | locator | tranid }"
```

Parameters

- **opid –** The Replication Agent operation ID for a database operation.
- **locator –** The Replication Agent locator for a database operation.
- **tranid –** The Oracle transaction ID for the database transaction.

Examples

- **Example 1 –**

```
ra_dumptran

0x0000.01783d95.0000:0001.000003fe.00000031.0010

go
```

  This command returns information about the operation specified by the Replication Agent operation ID:

```
Name                 Value
-------------------  ------------------------------------------
-
BEGIN OPERATION ID   0x0000.01783d95.0000:0001.000003fe.
00000031.0010
BEGIN SCN            24657302
TRANSACTION ID       0004.0016.00000016
USERNAME             AUSER
EXECUTION TIME       2010-07-12 10:28:14.0
THREAD NUMBER        1
TRANSACTION SKIP
COMMAND              pdb_skip_op add, 24657301, 1, 1022.49.16
DUMP FILE NAME       C:\somepath\XID0004.0016.00000016.log
(8 rows affected)
```

- **Example 2 –**

```
ra_dumptran
0000000001783d9600020001000003fe0000003400e8000001783d9500000000
go
```

This command returns information about the operation specified by the Replication Agent locator:

```
Name                 Value
-------------------  ------------------------------------------
-
BEGIN OPERATION ID   0x0000.01783d95.0000:0001.000003fe.
00000031.0010
BEGIN SCN            24657302
TRANSACTION ID       0004.0016.00000016
USERNAME             AUSER
EXECUTION TIME       2010-07-12 10:28:14.0
THREAD NUMBER        1
TRANSACTION SKIP
COMMAND              pdb_skip_op add, 0.24657301.0, 1, 1022.49.16
DUMP FILE NAME       C:\somepath\XID0004.0016.00000016.log
(8 rows affected)
```

- **Example 3 –**

```
ra_dumptran 0x0004.0016.00000016

go
```

This command returns information about the transaction specified by the Oracle transaction ID:

```
Name                 Value

-------------------  ---------------------------------------------
-

BEGIN OPERATION ID  0x0000.01783d95.0000:0001.000003fe.
00000031.0010

BEGIN SCN           24657302

TRANSACTION ID      0004.0016.00000016

USERNAME            AUSER

EXECUTION TIME      2010-07-12 10:28:14.0

THREAD NUMBER       1

TRANSACTION SKIP
COMMAND             pdb_skip_op add, 0.24657301.0, 1, 1022.49.16

DUMP FILE NAME      C:\somepath\XID0004.0016.00000016.log

(8 rows affected)
```

- **Example 4 –** The following is an example of log-file output from the **ra_dumptran** command. The log file contains both transaction information and information about all operations in the transaction:

```
File name: C:\somepath\XID0004.0016.00000016.log

File contents:

BEGIN OPERATION ID  0x0000.01783d95.0000:0001.000003fe.
00000031.0010

BEGIN SCN           24657302

TRANSACTION ID      0004.0016.00000016

USER NAME          AUSER

EXECUTION TIME   2010-07-12 10:28:14.0

THREAD NUMBER    1

TRANSACTION SKIP
COMMAND          pdb_skip_op add, 24657301, 1, 1022.49.16

SCN     THREAD OPERATION ID                                  OBJECT
ID OBJECT NAME REPLICATE OPERATION SQL

-------- ------ ----------------------------------------------
--------- ----------- --------- ---------
```

```
------------------------------------------------------------------
------------------------------------------------------------

24657301 1      0x0000.01783d95.0000:0001.000003fe.00000031.0010
0         NULL        NO        START     set transaction read
write;

24657301 1      0x0000.01783d95.0000:0001.000003fe.00000031.0010
51809     BLL$TEST    YES       INSERT    insert into "QA7USER"."BLL
$TEST"("QUANTITY","ORDER_NUMBER") values ('85','1234567890');

24657302 1      0x0000.01783d96.0000:0001.000003fe.00000033.010c
51809     BLL$TEST    YES       DELETE    delete from "QA7USER"."BLL
$TEST" where "QUANTITY" = '85' and "ORDER_NUMBER" = '1234567890'
and ROWID = 'AAAMphAAEAAAYrWAAC';

24657303 1      0x0000.01783d97.0000:0001.000003fe.00000035.00c4
0         NULL        NO        COMMIT    Commit;
```

Usage

- The **ra_dumptran** command dumps all operations for a specified transaction to an exclusive log file used in troubleshooting a failed operation or transaction. The log-file header consists of the result set returned by **ra_dumptran** and includes the following rows:
  - BEGIN OPERATION ID – the Replication Agent operation ID for the transaction **begin** operation. This field is in the format *wrap.scn.subscn.thread.lsn.block.offset*, where:
    - *wrap.scn.subscn* is the system change number (SCN) for the **begin** operation.
    - *thread* is the database thread number.
    - *lsn* is the log sequence number for the **begin** operation.
    - *block* is the block where the **begin** operation resides.
    - *offset* is the offset into the operation where the **begin** operation resides.
  - BEGIN SCN – the SCN for the transaction operation as logged in a redo log file.
  - TRANSACTION ID – the ID of the transaction that the operation is a part of.
  - USERNAME – the name of the user that executed the transaction.
  - EXECUTION TIME – the date and time at which the transaction was executed.
  - THREAD NUMBER – the Oracle thread that executed the transaction.
  - TRANSACTION SKIP COMMAND – the Replication Agent command that causes the transaction to be skipped by Replication Agent during replication.
  - DUMP FILE NAME – The name of the log file to which the transaction is written by **ra_dumptran**.
- The log file specified by the DUMP FILE row also contains the operation results for the specified transaction:
  - SCN – the SCN for the operation as logged in a redo log file.
  - THREAD – the thread that executed the operation.

  - OPERATION ID – the Replication Agent operation ID for the transaction **begin** operation. This field is in the format *wrap.scn.subscn.thread.lsn.block.offset*.
  - OBJECT ID – the object ID of the affected object.
  - OBJECT NAME – the name of the affected object.
  - REPLICATE – whether or not (YES or NO) the object affected by the operation is marked for replication by Replication Agent.
  - OPERATION – the operation type.
  - SQL – the SQL statement for the operation.
- The **ra_dumptran** command cannot operate properly unless the Oracle LogMiner script, `$ORACLE_HOME/rdbms/admin/dbmslm.sql`, has been installed at the primary database. If this script has not been installed, **ra_dumptran** will return an error.
- After LogMiner is installed, create a public synonym so that you do not have to log in as the owner to execute LogMiner functions:

```
CREATE PUBLIC SYNONYM DBMS_LOGMNR FOR
   SYS.DBMS_LOGMNR;
```

Note: This is required if you are using Oracle 10g.

- The following privileges must be granted to **pds_username** for the **ra_dumptran** command to function properly:
  - **EXECUTE_CATALOG_ROLE**
  - **SELECT ON V_$LOGMNR_CONTENTS**
  - **SELECT ON V_$LOGMNR_LOGS**
  - **SELECT ANY TRANSACTION**
- If the **ra_dumptran** command returns no result for a specified *opid* or *locator* value, the corresponding database operation may be one of many operations in a database transaction. In this case, you should instead specify the ID of the transaction to which the database operation belongs.

ra_finalize_upgrade

Finalizes the upgrade of an instance from a previous version and prevents downgrade to the previous version.

Syntax

```
ra_finalize_upgrade
```

Parameters

- **None –** There are no parameters.

Usage

- The **ra_finalize_upgrade** command allows you to manually force upgrade finalization of an instance from a previous version and prevent downgrade to the previous version.
- The **ra_finalize_upgrade** command is not valid for the instances for which upgrade is already finalized.
- Any new functionality is not enabled until the upgrade is finalized, to allow downgrade, if necessary. By manually finalizing the upgrade, you can force the Replication Agent to use the new functionality.
- The use of any new functionality automatically triggers upgrade finalization.

ra_help

Returns help information for Replication Agent commands.

Syntax

```
ra_help [command]
```

Parameters

- **command –** The name of a Replication Agent command for which you want to view help information.

Examples

- **Example 1 –**

  ```
  ra_help
  ```

  This command returns help for all Replication Agent commands.

- **Example 2 –**

  ```
  ra_help pdb_gen_id
  ```

  This command returns help for the **pdb_gen_id** command.

Usage

- If **ra_help** is invoked with no option, it returns help information for all Replication Agent commands.
- If **ra_help** is invoked with the command option, it returns help information only for the specified command.
- The **ra_help** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_config* on page 70

ra_helparchive

Displays a list of metadata for all managed archive logs, for a specific redo log thread, or for archive logs for a specific redo log thread.

Syntax

```
ra_helparchive [ redo_log_thread_id ]
```

Parameters

- ***redo_log_thread_id* –** is the ID of the archive log or redo log thread to display metadata for.

Usage

- If no thread ID is specified, **ra_helparchive** returns a list of metadata for all managed archive logs.

See also

- *pdb_archive_path* on page 167

ra_helparticle

Returns information about primary database articles from the RASD.

Syntax

```
ra_helparticle [article [, version]]
```

Parameters

- **article –** The name or object ID of an article (table or procedure) in the primary database. Article names can be qualified with an owner name in the following form:

  ```
  owner.article
  ```

  Owner qualification of article names is optional.
- **version –** A hexadecimal locator value that identifies the version of the article specified in the *article* option.

Examples

- **Example 1 –**

  ```
  ra_helparticle
  ```

  This command returns information about all versions of all articles in the RASD.

- **Example 2 –**

  ```
  ra_helparticle table1
  ```

  This command returns information about the current version of the article named “table1” in the RASD.

- **Example 3 –**

  ```
  ra_helparticle table1,
  00000000000210a400003334000700003334000699940000d413c50000000000
  ```

  This command returns information about version **00000000000210a400003334000700003334000699940000d413c50000000000** of the article named “table1” in the RASD.

Usage

- The **ra_helparticle** command returns the following information for articles (tables and procedures):
  - Article object ID
  - Primary database name
  - Article owner name or alias
  - Article name
  - Article type (table or procedure)
  - Article status (Current, Archived, or Dropped)
  - Article version number

  All information except the article type, article status, and article version number are the values returned by the primary database when Replication Agent is initialized with the **pdb_xlog init** command.
- If **ra_helparticle** is invoked with no option, it returns information for all versions of all articles (tables and procedures) in the RASD.
- If **ra_helparticle** is invoked with the *article* option, it returns information only for the current version of the specified article in the RASD.
- If **ra_helparticle** is invoked with the *article* and *version* options, it returns information only for the specified version of the specified article in the RASD.
- The **ra_helparticle** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_helpdb* on page 85
- *ra_helpfield* on page 87
- *ra_helplocator* on page 89
- *ra_helpuser* on page 93

ra_helpdb

Returns information about the primary database from the RASD.

Syntax

```
ra_helpdb
```

Usage

- When **ra_helpdb** is invoked, it returns the following information about the primary database:
  - Database object ID
  - Database name

  The database ID and database name are the values returned by the primary database when Replication Agent is initialized with the **pdb_xlog init** command.
- The **ra_helpdb** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.
- The **ra_helpdb** command is valid only after the RASD has been initialized, that is, only after you have executed **pdb_xlog init.**

See also

- *ra_helparticle* on page 83
- *ra_helpdevice* on page 85
- *ra_helpfield* on page 87
- *ra_helplocator* on page 89
- *ra_helpuser* on page 93
- *ra_updatedevices* on page 115

ra_helpdevice

This command returns information about the primary database log devices from the RASD log device repository.

Syntax

```
ra_helpdevice [device]
```

Parameters

- **device –** The device ID of the primary database log device.

Examples

- **Example 1 –**

```
ra_helpdevice
```

  This command returns information about all primary database log devices recorded in the log device repository.

- **Example 2 –**

```
ra_helpdevice 1
```

  This command returns information about the primary database log device ID "1" in the log device repository.

Usage

- The **ra_helpdevice** command returns the following information for each primary database log device recorded in the RASD:
  - Device ID – the log device ID defined by the primary data server.

    Note: For Oracle, the ID is the value of the Oracle Redo Log Group that this file belongs.
  - Database name – the name of the primary database associated with the log device.
  - Device name – the logical name of the log device defined by the primary data server.
  - Server device path – the path to a multiplexed version of the log device.
  - Disk mirror path – the path to the log device (at the standby site).
  - Disk device status – the current status of the server device path (**ACCESSIBLE**, **NOT_VALID**, or **OPEN**).
- The log device ID, primary database name, log device name, and server log device path are values returned by the primary data server when Replication Agent is initialized with the **pdb_xlog init** command, or when the log device repository is updated with the **ra_updatedevices** command.
- The disk mirror path is the current value recorded in the RASD. To find each log device, Replication Agent uses the disk mirror path recorded in its RASD.

  For each log device recorded in the RASD, you can set or change the disk device path with the **ra_devicepath** command.

  If you do not specify a disk device path using **ra_devicepath**, the value recorded for the disk mirror path is **DEFAULT**, and Replication Agent uses the value recorded for the server device path to find the log device.
- The disk device status is updated by the Log Reader component each time you invoke the **ra_helpdevice** command.
- If **ra_helpdevice** is invoked with no option, it returns information for all log devices recorded in the RASD log device repository.

- If **ra_helpdevice** is invoked with the **device** option, it returns information only for the specified log device.
- The **ra_helpdevice** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_helpdb* on page 85
- *ra_updatedevices* on page 115

ra_helpfield

Returns information about primary database fields (columns in tables, or input parameters in stored procedures) from the RASD.

Syntax

```
ra_helpfield article [, version [, field]]
```

Parameters

- **article –** The name or object ID of an article (table or procedure) in the primary database. Article names can be qualified with an owner name in the following form:

  ```
  owner.article
  ```

  Owner qualification of article names is optional.
- **version –** A hexadecimal locator value that identifies the version of the specified article.
- **field –** The name or object ID of a field (column or input parameter) in the specified article.

Examples

- **Example 1 –**

  ```
  ra_helpfield authors
  ```

  This command returns information about all fields in the current version of the article named *authors* in the RASD.

- **Example 2 –**

  ```
  ra_helpfield authors,
  00000000000210a400003334000700003334000699940000d413c50000000000
  ```

  This command returns information about all fields in version **00000000000210a400003334000700003334000699940000d413c50000000000** of the article named “authors” in the RASD.

- **Example 3 –**

```
ra_helpfield authors,
00000000000210a400003334000700003334000699940000d413c50000000000,
au_fname
```

This command returns information about the field named *au_fname* in version **00000000000210a400003334000700003334000699940000d413c50000000000** of the article named “authors” in the RASD.

Usage

- The **ra_helpfield** command returns the following information for fields:
  - Field (column or input parameter) object ID
  - Field name
  - Field type ID
  - Field datatype (with precision, length, and scale)
  - Field NULL mode
  - Field IDENTITY status
  - Field primary key status

  All field information items are the values returned by the primary database when Replication Agent is initialized with the **pdb_xlog init** command.
- If **ra_helpfield** is invoked with the *article* option, it returns information for all fields in the current version of the specified article in the RASD.
- If **ra_helpfield** is invoked with the *article* and *version* options, it returns information for all fields in the specified version of the specified article in the RASD.
- If **ra_helpfield** is invoked with the *article*, *version*, and *field* options, it returns information for the specified field in the specified version of the specified article in the RASD.
- The **ra_helpfield** command is valid when Replication Agent is in either Admin, Replicating, or Replication Down state.
- No results are returned by **ra_helpfield** if the RASD has not yet been initialized with the **pdb_xlog init** command.

See also

- *ra_config* on page 70
- *ra_help* on page 82
- *ra_helparticle* on page 83
- *ra_helpdb* on page 85
- *ra_helpdevice* on page 85
- *ra_helplocator* on page 89
- *ra_helpuser* on page 93

ra_helplocator

Returns information about fields in the LTM Locator value.

Syntax

```
ra_helplocator [locator_value]
```

Parameters

- **locator_value –** The hexadecimal string value of an LTM Locator.

Examples

- **Example 1 –**

  ```
  ra_helplocator
  ```

  This command returns information about fields in the current LTM Locator value.

- **Example 2 –**

  ```
  ra_helplocator locator_value
  ```

  This command returns information about fields in the specified LTM Locator value.

Usage

- The **ra_helplocator** command returns the following information about the LTM Locator value:
  - Locator field names
  - Locator field hexadecimal values
  - Locator field decimal values
- If **ra_helplocator** is invoked with no option, it returns information about fields in the current LTM Locator value.
- If **ra_helplocator** is invoked with the *locator_value* option, it returns information about fields in the specified LTM Locator value.
- The **ra_helplocator** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_config* on page 70
- *ra_help* on page 82
- *ra_locator* on page 95

ra_helpop

This command returns information for use in troubleshooting a specified database transaction log operation.

Syntax

```
ra_helpop "{ opid | lsn | locator}"
```

Parameters

- **opid –** The Replication Agent operation ID for a database operation.
- **locator –** The Replication Agent locator for a database operation.

Examples

- **Example 1 –**

```
ra_helpop
0x0000.01783d96.0000:0001.000003fe.00000034.00e8
go
```

This command returns information about the operation specified by the Replication Agent operation ID:

```
Name             Value
---------------  ---------------------------------
OPERATION ID     0x0000.01783d96.0000:0001.000003fe
                 .00000034.00e8
SCN              24657302
THREAD           1
USERNAME         AUSER
EXECUTION TIME   2010-07-12 10:28:14.0
OBJECT ID        51809
OBJECT NAME      BLL$TEST
OPERATION        INSERT
REPLICATE        YES
TRANSACTION ID   0004.0016.00000016
REDO SQL         insert into "AUSER"."BLL$TEST"(
                 "QUANTITY","ORDER_NUMBER") values
                 ('85','1234567890');
UNDO SQL         delete from "AUSER"."BLL$TEST" where
                 "QUANTITY"='85' and "ORDER_NUMBER"
                 = '1234567890' and ROWID =
                 'AAAMphAAEAAAYrWAAC';
SKIP COMMAND     pdb_skip_op add, 24657302, 1,
                 1022.52.2322
(13 rows affected)
```

- **Example 2 –**

```
ra_helpop
0000000001783d9600020001000003fe0000003400e8000001783d9500000000
go
```

This command returns information about the operation specified by the Replication Agent locator:

```
Name             Value
---------------  ---------------------------------
OPERATION ID     0x0000.01783d96.0000:0001.000003fe
                 .00000033.010c
SCN              24657302
THREAD           1
USERNAME         AUSER
EXECUTION TIME   2010-07-12 10:28:14.0
OBJECT ID        51809
OBJECT NAME      BLL$TEST
OPERATION        INSERT
REPLICATE        YES
TRANSACTION ID   0004.0016.00000016
REDO SQL         insert into "AUSER"."BLL$TEST"(
                 "QUANTITY","ORDER_NUMBER") values
                 ('85','1234567890');
UNDO SQL         delete from "AUSER"."BLL$TEST" where
                 "QUANTITY"='85' and "ORDER_NUMBER"
                  = '1234567890' and ROWID =
                 'AAAMphAAEAAAYrWAAC';
SKIP COMMAND     pdb_skip_op add, 24657302, 1,
                 1022.52.232
(13 rows affected)
```

Usage

- The **ra_helpop** command displays database and Replication Agent information for a specified operation for use in troubleshooting a failed operation or transaction. The result set returned by **ra_helpop** includes the following rows:
  - OPERATION ID – the Replication Agent operation ID in the format *wrap.scn.subscn.thread.lsn.block.offset*, where:
    - *wrap.scn.subscn* is the system change number (SCN) for the specified operation.
    - *thread* is the database thread number.
    - *lsn* is the log sequence number for the specified operation.
    - *block* is the block where the specified operation resides.
    - *offset* is the offset into the operation where the specified operation resides.
  - SCN – the SCN for the operation as logged in a redo log file.
  - THREAD – the thread that executed the operation.
  - USERNAME – the name of the user that executed the operation.
  - EXECUTION TIME – the date and time at which the operation was executed.
  - OBJECT ID – the database ID of the affected object.
  - OBJECT NAME – the name of the affected object.
  - OPERATION – the operation type.
  - REPLICATE – whether or not (YES or NO) the object affected by the operation is marked for replication by Replication Agent.

  - TRANSACTION ID – the ID of the transaction that the operation is a part of.
  - REDO SQL – the SQL that can be used to replay the operation.
  - UNDO SQL – the SQL that can be used to undo the operation.
  - SKIP COMMAND – the Replication Agent command that causes the operation to be skipped by Replication Agent during replication.
- The **ra_helpop** command can display information for more than one operation. Operation information is returned for each operation that has the SCN specified by *locator* or *opid*.
- The **ra_helpop** command cannot operate properly unless the Oracle LogMiner script, `$ORACLE_HOME/rdbms/admin/dbmslm.sql`, has been installed at the primary database. If this script has not been installed, **ra_helpop** will return an error.
- After LogMiner is installed, create a public synonym so that you do not have to log in as the owner to execute LogMiner functions:

```
CREATE PUBLIC SYNONYM DBMS_LOGMNR FOR
   SYS.DBMS_LOGMNR;
```

Note: This is required if you are using Oracle 10g.

- The following privileges must be granted to **pds_username** for the **ra_helpop** command to function properly:
  - **EXECUTE_CATALOG_ROLE**
  - **SELECT ON V_$LOGMNR_CONTENTS**
  - **SELECT ON V_$LOGMNR_LOGS**
  - **SELECT ANY TRANSACTION**
- A single DML command may be represented in the Oracle redo log as a succession of two or more operations. Oracle LogMiner, however, will display only the SCN of the first operation. Consequently, **ra_helpop** may return no result for an operation that occurs in the middle of a succession of operations in the Oracle redo log. If **ra_helpop** returns no result for a specified opid or locator value, use the **ra_dumptran** command, specifying the transaction ID of the transaction to which the database operation belongs, and dump the entire transaction from the Oracle redo log. Then, search the dump for the operation that Oracle LogMiner did not find.

ra_helptran

Returns a list of all open transactions.

Syntax

```
ra_helptran
```

Usage

If there are no open transactions, **ra_helptran** returns an empty result set.

ra_helpuser

Returns information about primary database users from the RASD.

Syntax

```
ra_helpuser [user [, version]]
```

Parameters

- **user –** The name or user ID of a user in the primary database.
- **version –** The version number of the database user in the RASD.

Examples

- **Example 1 –**

  ```
  ra_helpuser
  ```

  This command returns information about all versions of all users in the RASD.

- **Example 2 –**

  ```
  ra_helpuser bob
  ```

  This command returns information about the current version of the database name “bob” in the RASD.

- **Example 3 –**

  ```
  ra_helpuser bob,
  00000000000210a400003334000700003334000699940000d413c50000000000
  ```

  This command returns information about version **00000000000210a400003334000700003334000699940000d413c50000000000** of the database user named “bob” in the RASD.

Usage

- The **ra_helpuser** command returns the following information about primary database users:
  - User ID
  - User name
  - User status (Current, Archived, or Dropped)
  - Primary database version (locator value)

  The user ID and user name are the values returned by the primary database when Replication Agent is initialized with the **pdb_xlog init** command.

- If **ra_helpuser** is invoked with no option, it returns information about all users in all versions of the primary database in the RASD.
- If **ra_helpuser** is invoked with the *user* option, it returns information about the current version of the specified user in the primary database in the RASD.
- If **ra_helpuser** is invoked with the *user* and *version* options, it returns information about the specified user in the specified version of the primary database in the RASD.
- The **ra_helpuser** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.
- No results are returned by **ra_helpuser** if the RASD has not been initialized by the **pdb_xlog init** command.

See also

- *ra_config* on page 70
- *ra_help* on page 82
- *ra_helparticle* on page 83
- *ra_helpdb* on page 85
- *ra_helpdevice* on page 85
- *ra_helpfield* on page 87
- *ra_helplocator* on page 89

ra_license

Returns license information for Replication Agent and its licensed features.

Syntax

```
ra_license [param]
```

Parameters

- **param –** Directs **ra_license** to return information about the configurable parameters for the license.

Examples

- **Example 1 –**

  ```
  ra_license
  ```

  This command returns basic license information like:

  ```
  License Name  Version    Quantity  Status     Expiry Date
  ------------  ---------  --------  ---------  ------------------
  RAX_SERVER    2012.xxxx  2         expirable  Oct 10 2013 7:30AM
  ```

- **Example 2 –**

  ```
  ra_license param
  ```

This command returns information about the configurable parameters for the license:

```
Property          Value
--------          -----
License Edition   Development and
                  Testing License
License Type      CP
Licensed to       Sybase, Inc.
Total Licenses    2
Total in Use      1
Email Severity    NONE
SMTP Host         smtp
SMTP Port         25
Email Sender      tomservo@sybase.com
Email Recipients  deep13@sybase.com
10 rows Affected.
```

Usage

- The **ra_helpuser** command without any keyword returns basic license information.
- The **ra_helpuser** command with the **param** keyword returns information about configurable license parameters, including the license edition, type, SMTP host and SMTP port.

ra_locator

Returns the current value of the LTM Locator maintained by Replication Agent, requests an LTM Locator value from the primary Replication Server, or sets the value of the LTM Locator maintained by Replication Agent to zero.

Syntax

```
ra_locator [ { update | zero | move_truncpt } ]
```

Parameters

- **update –** the optional keyword to request a new LTM Locator value from the primary Replication Server.
- **zero –** The optional keyword to set the value of the LTM Locator stored in the Replication Agent transaction log to zero.
- **move_truncpt –** The keyword that moves the truncation point.

Examples

- **Example 1 –**

  ```
  ra_locator
  ```

  This command returns the current value of the LTM Locator maintained by Replication Agent, as shown:

  ```
  Locator
   ------------------------------------------------
  ```

```
 0000000052000000000000000527FFFFFFFFFFFFFFF0022FB3B
 (1 row affected)
```

- **Example 2 –**

  ```
  ra_locator update
  ```

  This command requests a new LTM Locator value from the primary Replication Server.

- **Example 3 –**

  ```
  ra_locator zero
  ```

  This command sets the value of the LTM Locator maintained by Replication Agent to all zeros.

- **Example 4 –**

  ```
  ra_locator move_truncpt
  ```

  This command moves the transaction log truncation point to the end of the current transaction log.

Usage

- When you invoke **ra_locator** with no option, it returns the current value of the LTM Locator maintained by the Replication Agent instance. Replication Agent stores the value of the LTM Locator in the RASD.

  Note: The value of the LTM Locator that is maintained by Replication Agent is also known as the origin queue ID.

- When you invoke **ra_locator** with the **update** keyword, it requests a new LTM Locator value from the primary Replication Server, and Replication Agent saves the value.

  Note: When the you invoke **ra_locator** with the **update** keyword, the change takes effect only if the Replication Agent instance is in Replicating state.

- When you invoke **ra_locator** with the **zero** keyword, it sets the value of the LTM Locator maintained by Replication Agent to zero.
- The LTM Locator contains information that Replication Agent uses to determine where to start reading the transaction log.

  Upon start-up or recovery from a connection failure, Replication Agent automatically requests an LTM Locator value from the primary Replication Server.
  - If the value of the LTM Locator returned from the primary Replication Server is zero, Replication Agent uses the LTM Locator value stored in the transaction log system table.
  - If the value of the LTM Locator in the transaction log system table is zero, Replication Agent starts reading the transaction log from either the current beginning of the log.
- For more information about the format of the origin queue ID, see the section for your specific primary data server in the *Replication Agent Primary Database Guide*.

- If the Replication Agent transaction log does not exist, the **ra_locator** command returns an error message.
- The **ra_locator** command with the **zero** keyword is valid only when the Replication Agent instance is in the Admin or Replication Down state.
- Without the **zero** keyword, the **ra_locator** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.
- If you invoke **ra_locator** with the **move_truncpt** keyword, the truncation point is moved to the end of the log without change or modification to any Replication Agent components. (for Oracle, this is the end of the current online redo log.) The **move_truncpt** option has no effect if Replication Agent has not been initialized.

  Note: To prevent Replication Server from requesting a log starting point that occurs earlier in the log than the location established by the **move_truncpt** option, Replication Server's LTM locator value for the primary connection must be zeroed. Execute the Replication Server System Database (RSSD) **rs_zeroltm** command against the primary database connection to zero the LTM locator.

  If you move the secondary truncation point to the end of the primary database transaction log using **ra_locator move_truncpt**, you risk skipping over any DDL commands record in the log. The DDL commands might have been used by Replication Agent to update information stored within the Replication Agent System Database (RASD). If the RASD contents are incorrect due to skipping processing of some log records, you may force all of the schema information in the RASD to be refreshed using **ra_admin refresh**. If only the schema for a single object stored in the RASD is of concern, you can unmark and remark just that single object, which forces the schema of the object to be reread into the RASD.

See also

- *pdb_gen_id* on page 11
- *pdb_truncate_xlog* on page 60
- *pdb_xlog* on page 61
- *ra_admin* on page 66

ra_maintid

Returns the login name of the primary database maintenance user.

Syntax

```
ra_maintid
```

Usage

- Replication Server requires a maintenance user login name for each database connection. The maintenance user login name for a database connection is specified with the Replication Server **create connection** or **alter connection** command.

  When the primary database maintenance user login name is changed in Replication Server (using the **alter connection** command), Replication Server automatically sends the new

maintenance user login name to Replication Agent, if Replication Agent is in Replicating state.

Each time Replication Agent goes into Replicating state, it automatically retrieves the primary database maintenance user login name from the primary Replication Server, and caches it.

- When **ra_maintid** is invoked, it returns the login name of the primary database maintenance user that is cached, as follows:

```
maintenance user
 ----------------
 SYS
(1 row affected)
```

- If **ra_maintid** is invoked when Replication Agent is in Replicating state, it always returns the correct maintenance user login name.

  If **ra_maintid** is invoked when Replication Agent is in Admin or Replication Down state, it may not return the correct maintenance user login name, because the maintenance user login name could have changed in Replication Server after the last time Replication Agent retrieved the value and stored it.
- The **filter_maint_userid** configuration parameter is provided to support bidirectional replication, wherein the primary database also acts as a replicate database that has transactions applied to it by a Replication Server.

  If the value of the **filter_maint_userid** parameter is **true**, database operations applied by the maintenance user are not replicated from the primary database. When it reads the transaction log, the Replication Agent Log Reader component filters out data-changing operations applied by the maintenance user.
- The **ra_maintid** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_config* on page 70
- *ra_statistics* on page 104

ra_marker

Places a marker in the primary database transaction log.

Syntax

```
ra_marker command_tag
```

Parameters

- **command_tag –** A `varchar` value that contains information used for subscription materialization. When used for Oracle data servers, this value is `varchar(4000)`.

Examples

- **Example 1 –**

```
ra_marker 'activate subscription 309 0 with suspension'
```

  This command places a marker object in the Primary Database transaction log that invokes the Replication Server **activate subscription** command.

Usage

- When **ra_marker** is invoked, Replication Agent executes a transaction in the Primary Database that is captured in the Primary Database transaction log. The replicated transaction is sent as a marker object to the primary Replication Server.
- The **ra_marker** command returns an error message if the Replication Agent transaction log does not exist.
- The **ra_marker** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.
- For more information about the Replication Server **rs_marker** system function, refer to the *Replication Server Administration Guide* and *Replication Server Reference Manual*.

See also

- *ra_dump* on page 76

ra_migrate

Performs any necessary migration for upgrade and downgrade tasks between releases of Replication Agent.

This command is used to complete the upgrade process or to complete the downgrade process started by the **ra_downgrade** command, which is executed by the instance from which Replication Agent is being downgraded (the later version).

Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands have been deprecated. Use the **ra_downgrade** and **ra_migrate** commands where possible. See the *Replication Agent Primary Database Guide*.

Syntax

```
ra_migrate
```

Parameters

- **None –** There are no parameters.

Usage

- After upgrading to a new release of Replication Agent, you must first run **ra_migrate** to update to the latest version of Replication Agent.
- The **ra_migrate** command is valid when the Replication Agent instance is in the Admin or Replication Down state.
- The **ra_migrate** command will verify that the following privileges have been granted to **pds_username**:
  - **EXECUTE_CATALOG_ROLE**
  - **select on V_$LOGMNR_CONTENTS**
  - **select on V_$LOGMNR_LOGS**

  These privileges are necessary for the **ra_dumptran** and **ra_helpop** commands to function properly. These privileges are not required for replication, only for using the **ra_dumptran** and **ra_helpop** commands, which are used in debugging and troubleshooting. If these privileges have not been granted at the time **ra_migrate** is invoked, a warning message is returned and logged in the Replication Agent log file.
- After a downgrade, the **ra_migrate** command restores the RASD from file.
- To use **ra_migrate**, Replication Agent must be able to connect to the primary database.

See also

- *ra_downgrade* on page 73
- *ra_downgrade_accept* on page 74
- *ra_downgrade_prepare* on page 75

ra_purge_first_open

Removes the first open transaction from the list of open transactions.

Syntax

```
ra_purge_first_open
```

Usage

- If there are no open transactions, invoking **ra_purge_first_open** results in an error.
- Use **ra_purge_first_open** only under the direction of Sybase Technical Support if there is a possibility that the transaction in question may contain content that is to be replicated.

ra_regenerate_keys

Regenerates the value of the **instance_rand** configuration property.

Syntax

```
ra_regenerate_keys
```

Parameters

- **None –** There are no parameters.

Usage

- When the **ra_regenerate_keys** command is invoked, all encrypted passwords in the user-info and password tables are reencrypted. However, the unencrypted values do not change.
- When the **ra_regenerate_keys** command is invoked, the date of creation (ctime) attribute in the encryption keys table is set to the time of execution of **ra_regenerate_keys**.

ra_set_autocorrection

Enables or disables autocorrection for marked tables.

Syntax

```
ra_set_autocorrection { all | tablename } [, { enable | disable } ]
```

Parameters

- **all –** To enable autocorrection for all marked tables, follow the **ra_set_autocorrection** command with the **all** and **enable** keywords:

  ra_set_autocorrection all, enable

  To disable autocorrection for all marked tables, follow the **ra_set_autocorrection** command with the **all** and **disable** keywords:

  ra_set_autocorrection all, disable
- **tablename –** To enable autocorrection for one marked table, follow the **ra_set_autocorrection** command with the *tablename* parameter and the **enable** keyword:

  ra_set_autocorrection tablename, enable

  To disable autocorrection for one marked table, follow the **ra_set_autocorrection** command with the *tablename* parameter and the **disable** keyword:

  ra_set_autocorrection tablename, disable

  To display autocorrection status for one marked table, follow the **ra_set_autocorrection** command with the *tablename* parameter alone:

  ra_set_autocorrection tablename
- **enable –** Use the **enable** keyword to enable autocorrection for one marked table or all marked tables.
- **disable –** Use the **disable** keyword to disable autocorrection for one marked table or all marked tables.

Examples

- **Example 1 –**

```
ra_set_autocorrection mytable, enable
```

  This command enables autocorrection for the marked table named mytable.

- **Example 2 –**

```
ra_set_autocorrection all, enable
```

  This command enables autocorrection for all marked tables.

- **Example 3 –**

```
ra_set_autocorrection mytable, disable
```

  This command disables autocorrection for the marked table named mytable.

- **Example 4 –**

```
ra_set_autocorrection all, disable
```

  This command disables autocorrection for all marked tables.

- **Example 5 –**

```
ra_set_autocorrection mytable
```

  This command displays autocorrection status for the marked table named mytable.

Usage

- This command is used to support Replication Server autocorrection functionality.
- You cannot set autocorrection for tables that have not been marked for replication. If a marked table for which autocorrection is enabled is subsequently unmarked, autocorrection is automatically disabled for the table.
- The **column_compression** and **ltl_send_only_primary_keys** configuration parameters are disregarded when Replication Agent is replicating a marked table for which autocorrection has been enabled.
- When autocorrection is enabled, Replication Server converts each **update** or **insert** operation into a pair of operations: one **delete** operation followed by an **insert**.
- If your primary database is Oracle and table-level supplemental logging has not already been enabled, enabling autocorrection will enable supplemental logging for all columns of the specified table.
- If your Replication Agent instance is configured to send minimal column data—**column_compression** and **ltl_send_only_primary_keys** are set to true—some column data may be omitted for columns that are specified as searchable in a replication definition. Consequently, errors may occur at a subscribing database where data needed for an insert, subscription migration, or custom function string is missing. Sybase therefore

recommends that you enable autocorrection for any table referenced in a replication definition with searchable columns.

ra_set_login

Sets the Replication Agent administrator login and password.

Syntax

```
ra_set_login username, password[, encryption ]
```

Parameters

- **username –** The login name of the Replication Agent administrator.

  Do not specify the following configuration parameter names as the Replication Agent administrator login name:

  - ddl_username
  - ddl_password
  - function_username
  - function_password
  - pds_username
  - pds_password
  - rssd_password
  - rssd_username
  - rs_password
  - rs_username
  - asm_username
  - asm_password
  - rman_username
  - rman_password
- **password –** The password of the Replication Agent administrator.
- **encryption –** The encryption mode for the Replication Agent administrator login password:

  - 3 – clients must use the Tabular Data Stream™ (TDS) Extended Plus Encrypted Password protocol.
  - 2 – clients must use the extended encrypted password negotiation or the TDS Extended Plus Encrypted Password protocol.
  - 1 – clients must use the extended encrypted password negotiation.
  - 0 – clients may choose the encryption mode and may use no encryption.

Examples

- **Example 1 –**

```
ra_set_login tom, S3Rv0
```

This command sets the Replication Agent administrator login to "tom" and the password to "S3Rv0."

```
ra_set_login crow, Tr0bOt, 3
```

This command sets the Replication Agent administrator login to "crow," sets the password to "Tr0b0t," and requires clients to use the TDS Extended Plus Encrypted Password protocol.

Usage

- The Replication Agent administrator login has permission to log in to the Replication Agent instance through the administration port.
- Only one Replication Agent administrator login name is valid at any time.
- Any change in the Replication Agent administrator login or password takes place immediately, and you must use the new login and password the next time you log in to the Replication Agent instance.
- The password specified for the administrator login is encrypted in the Replication Agent configuration file.
- The **ra_set_login** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.
- You can specify an encrypted password for this parameter.

See also

- *ra_config* on page 70

ra_statistics

Returns performance-related statistics for Replication Agent components and the Java Virtual Machine (Java VM), or resets the statistics counters.

Note: The statistics counters may vary by primary database.

Syntax

```
ra_statistics [component|reset]
```

Parameters

- **component –** The optional keyword that identifies a Replication Agent component or the Java VM. Valid *component* keywords are:

  - **LR** – Log Reader
  - **LTI** – Log Transfer Interface
  - **LTM** – Log Transfer Manager
  - **VM** – Java Virtual Machine
- **reset –** The optional keyword that resets the statistics counters.

Examples

- **Example 1 –**

```
ra_statistics
```

  This command returns performance statistics for the Replication Agent instance and the Java VM.

- **Example 2 –**

```
ra_statistics reset
```

  This command resets the statistics counters for the Replication Agent instance.

- **Example 3 –**

```
ra_statistics VM
```

  This command returns statistics for the Java VM. See table 4 for details.

Usage

- If you invoke **ra_statistics** with no option, it returns statistics for all Replication Agent components and the Java VM.
- If you invoke **ra_statistics** with a *component* option, the ra_statistics command returns statistics for the specified Replication Agent component or the Java VM.
- *Table 4. Java VM Statistics* on page 105 lists the statistics returned for the Java VM.

Table 4. Java VM Statistics

Statistic	Description
VM maximum memory	Maximum memory (in bytes) available to the Java VM
VM total memory allocated	Total memory (in bytes) allocated to the Java VM at start-up
VM free memory	Memory (in bytes) allocated but not used by the Java VM
VM memory usage	Memory (in bytes) allocated and in use by the Java VM
VM % max memory used	Percentage of the maximum memory available to the Java VM, currently in use by the Java VM

- *Table 5. Log Transfer Manager Statistics* on page 106 lists the statistics returned for the Log Transfer Manager component.

Table 5. Log Transfer Manager Statistics

Statistic	Description
Time statistics obtained	Day, date, and time when **ra_statistics** was invoked and information returned
Time replication last started	Day, date, and time that Replicating state was entered
Time statistics last reset	Day, date, and time that statistics counters were reset
Items held in Global LRUCache	Number of object references in the internal Least Recently Used cache

- *Table 6. Log Reader Statistics for Oracle* on page 106 lists the statistics returned for the Log Reader component for Oracle.

Table 6. Log Reader Statistics for Oracle

Statistic	Description
Average RBA search time (ms)	The average record byte address (RBA) search time during log scanner positioning
Total bytes read	The total number of bytes read from the primary database transaction log
Total log records read	The total number of log records read from the primary database transaction log
Average number of bytes read per second	The average number of bytes read from the primary database transaction log per second
Average number of bytes per record	The average number of bytes per log record read
Average time (ms) per log read	The average time per primary database transaction log read
Total online log read time (ms)	The total time spent reading the primary database online transaction redo log
Total archive log read time (ms)	The total time spent reading primary database transaction redo log archives
Average time (ms) per online log device read	The average time per online log device read
Average time (ms) per archive log device read	The average time per archive log device read

Statistic	Description
Total log records queued	The total number of log records queued for processing
Total log records filtered	The total number of log records filtered
Log scan checkpoint set size	The current number of log records in the checkpoint set
Average number of log records per checkpoint	The average number of log records for each checkpoint log record read
Average number of seconds between log record checkpoints	The average number of seconds between reading log record checkpoints
Total operations scanned	Number of operations read from log devices since last reset
Total operations processed	Number of operations read from log devices and passed to LTI since last reset
Total operations skipped	Number of operations read from log devices and not processed for any reason since last reset
Total maintenance user operations filtered	Number of maintenance user operations read from log devices and skipped since last reset
Avg operation processing time	Average Log Reader operation processing time (in milliseconds) since last reset
Total transactions processed	Number of transactions read from log devices since last reset
Total transactions skipped	Number of transactions read from log devices and not processed for any reason since last reset
Total transactions opened	Number of **begin transaction** commands read from log devices since last reset
Total transactions closed	Number of **commit** and **rollback** commands read from log devices since last reset
Total transactions committed	Number of **commit** commands read from log devices since last reset
Total transactions aborted (rolled back)	Number of **rollback** commands read from log devices since last reset
Total system transactions skipped	Number of system transactions read from log devices and skipped since last reset
Avg ops per transaction	Average number of operations in each transaction read from log devices since last reset
Current scan buffer size	Current size of the Log Reader scan buffer
Current operation queue size	Current size of the Log Reader operation queue
Current session cache size	Current size of the session cache

Statistic	Description
Total LOB operations processed by query data from PDB	The total number of LOB operations that have been processed from the primary database
Avg time used to query PDB for LOB operation processing	The average time taken to query the primary database to process a LOB
Current Op Proc RASD marked object cache size	Current size of the operation processor marked object repository cache
Total number of Op Proc RASD marked object cache hits	Total number of operation processor marked object repository cache hits
Total number of Op Proc RASD marked object cache misses	Total number of operation processor marked object repository cache misses
Log reposition point locator	Locator value of reposition point in log device
Last processed operation locator	Locator value of most recently processed operation read from log devices
Avg xlog operation wait time (ms)	Average time (in milliseconds) that Log Reader had to wait for each new operation to appear in the log since last reset
Avg sender operation processing time (ms)	Average time (in milliseconds) that Log Reader sender took to process each operation since last reset
Avg sender operation wait time (ms)	Average time (in milliseconds) that Log Reader sender had to wait to send each processed operation to the LTI input queue since last reset
Avg change set send time (ms)	Average time (in milliseconds) that Log Reader sender took to send each processed operation to the LTI input queue since last reset
Number of sender operations processed	Number of operations that Log Reader sender processed since last reset
Current marked objects cache size	Marked objects cache size

- *Table 7. Additional Statistics for Oracle RAC* on page 108 lists the statistics returned when the primary database is Oracle RAC. These statistics exist in addition to the normal Log Reader statistics listed in the *Table 6. Log Reader Statistics for Oracle* on page 106.

Table 7. Additional Statistics for Oracle RAC

Statistic	Description
Log scan reader current LSN	The current log sequence number of the log being read for each cluster instance
Log scan reader end-of-log status	The current end of log status for each cluster log scanner

Statistic	Description
Log scan reader last read time	The number of seconds since the last read for each cluster scanner
Log scan record set distribution	Distribution of the log scan checkpoint set across all log scan threads
Log scan reader last record SCN	The SCN of the last log record read by each cluster scanner
Log scan reader checkpoints	The checkpoint SCN of the last checkpoint log record read by each cluster scanner
Log scan checkpoint SCN	The current checkpoint SCN, based on all cluster scanners
Log scan active checkpoint SCN	The active checkpoint SCN, based on all cluster scanner
Total log records read per redo log thread	The distribution of total log records read across all log scan threads
Log scan record set sizes	The current scan record set size for each cluster scanner
Log scan checkpoint queue sizes	The current checkpoint queue size for each cluster scanner

- *Table 8. Log Transfer Interface Statistics* on page 109 lists the statistics returned for the Log Transfer Interface component.

Table 8. Log Transfer Interface Statistics

Statistic	Description
Number of LTL commands sent	Total number of LTL commands sent to Replication Server since last reset
Avg LTL command size	Average size (in bytes) of each LTL command sent to Replication Server since last reset
Avg LTL commands/sec	Average number of LTL commands sent per second to Replication Server since last reset
Total bytes sent	Number of bytes sent to Replication Server since last reset
Avg Bytes/second during transmission	Average bytes per second sent over connection to Replication Server since last reset
Avg LTL buffer cache time	Average time (in milliseconds) it takes between placing the LTL commands into the LTL buffer to the time it is actually sent to Replication Server
Avg Rep Server turnaround time	Average time (in milliseconds) it takes Replication Server to acknowledge each LTL command buffer sent since last reset
Avg time to create distributes	Average time (in milliseconds) LTI takes to convert a change-set into LTL since last reset

Statistic	Description
Avg LTL buffer size	Average size (in bytes) of each LTL buffer sent to Replication Server since last reset
Avg LTM buffer utilization (%)	Average utilization (in percentage of LTL buffer size) of each LTL buffer sent to Replication Server since last reset
Avg LTL commands/buffer	Average number of LTL commands per buffer sent to Replication Server since last reset
Encoded column name cache size	Current encoded column name cache size
Current number of commands in the LTI queue	Current number of commands in the LTI queue
Current number of unformatted commands in the LTI queue	Current number of unformatted commands in the LTI queue
Last QID sent	Hex value of most recent origin queue ID sent to Replication Server
Last transaction id sent	Hex value of most recent transaction ID sent to Replication Server

- Statistics counters are reset automatically each time the Replication Agent instance goes into Replicating state.
- If you invoke **ra_statistics** with the **reset** keyword, Replication Agent immediately resets all of the statistics, except the following:
  - Time statistics obtained (LTM)
  - Time replication last started (LTM)
  - Time statistics last reset (LTM)
  - Last QID sent (LTI)
  - Last transaction ID sent (LTI)
  - All Java VM statistics

  Note: All Java VM statistics are refreshed each time you invoke **ra_statistics**. All queue, buffer, and cache size statistics are also refreshed and are not reset in the way averages or sums are reset.
- The **ra_statistics** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_statrack_interval* on page 195
- *ra_status* on page 112

ra_statrack

Starts and stops the statistics tracking thread.

Syntax

```
ra_statrack { start | stop | status }
```

Parameters

- **start –** starts the statistics tracking thread.
- **stop –** stops the statistics tracking thread.
- **status –** displays a running status for Replication Agent.

Usage

The statistics tracking thread gathers statistics at a default interval of 60 seconds and logs statistics into `STATRACK.log`. **ra_statrack** also displays the current running status for Replication Agent.

See also

- *ra_statrack_interval* on page 195
- *ra_statrack_list* on page 111

ra_statrack_list

Adds or removes a group of statistics from the tracking list, replaces the tracking list, and displays a list of statistics currently being tracked.

Syntax

```
ra_statrack_list
[ reset  |
  { {add | delete}, statistic_name }  |
  { replace, statistic_list } ]
```

Parameters

- **reset –** resets the list to track all available statistics.
- **add –** adds the specified statistic to the list of statistics being tracked.
- **delete –** removes the specified statistic from the list of statistics being tracked.
- ***statistic_name* –** is the statistic to be added to or removed from the list of statistics being tracked.
- **replace –** replaces the current list of statistics being tracked with the specified list.
- ***statistic_list* –** is the list of statistics to replace the current list of statistics being tracked.

Usage

- The value of *statistic_name* must be VM, LTM, LTI, or LR:
  - **LR** – Log Reader
  - **LTI** – Log Transfer Interface
  - **LTM** – Log Transfer Manager
  - **VM** – Java Virtual Machine

  These values are not case sensitive.
- The value of *statistic_list* must be VM, LTM, LTI, or LR. These values are not case sensitive. If more than one value is specified, the list must be enclosed in double quotes, and the list items must be separated by commas.

See also

- *ra_statrack* on page 111
- *ra_statrack_interval* on page 195

ra_status

Returns the current state of the Replication Agent instance.

Syntax

```
ra_status
```

Usage

- When **ra_status** is invoked, it returns the current state of the Replication Agent instance, and a brief description of the current state, as follows:

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

  Note: If the first word in the description is "Transitioning," the Replication Agent instance is in transition between states. Some commands are not valid when the Replication Agent instance is in state transition.
- Replication Agent states are:
  - Admin – in this state, the Replication Agent instance is running, but no connections are up. You can change any configuration parameter when the Replication Agent instance is in Admin state.
  - Replicating – in this state, the Log Reader component is scanning the transaction log for operations to replicate from the primary database. If there are operations to be replicated, the Log Transfer Interface component is sending LTL commands to Replication Server.

- Replicating (Resynchronization) – in this state, Replication Agent has been restarted and is resynchronizing the primary and replicate databases.
- Replication Down – in this state, replication has stopped due to an error. After the error has been resolved, Replication Agent may return to the Replicating state.

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

  See the *Replication Agent Administration Guide* for more information about Replication Agent states.
- The **ra_status** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *quiesce* on page 65
- *ra_statistics* on page 104
- *resume* on page 124
- *shutdown* on page 130
- *suspend* on page 131

ra_truncatearticles

Truncates unused articles in the RASD.

Syntax

```
ra_truncatearticles locator
```

Parameters

- **locator –** The log locator value (LTM Locator) that identifies the cutoff point for truncating older versions of articles from the system data repository.

Usage

- When **ra_truncatearticles** is invoked, it truncates all non-current versions of all primary database articles in the system data repository older than the version identified by the *locator* value.

  If the current (most recent) version of an article is older than the version identified by the *locator* value, it is not truncated.
- Most common DDL commands and stored procedures executed in the primary database (such as **alter table**) are recorded in the transaction log, and replicated to the standby database. When it processes those DDL transactions for replication, Replication Agent updates its RASD automatically, creating a new version of the affected primary database articles.

Use **ra_truncatearticles** as part of a periodic maintenance procedure to prevent the RASD from growing indefinitely. See the *Replication Agent Administration Guide* for more information.

Note: Be sure to back up the RASD using **rasd_backup** before you truncate it.

- The **ra_truncatearticles** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_truncateusers* on page 114

ra_truncateddlfilters

Truncates old lists of DDL commands that are filtered in the RASD.

Syntax

```
ra_truncateddlfilters locator
```

Parameters

- **locator –** The log locator value (LTM Locator) that identifies the cutoff point for truncating older lists of DDL commands that are filtered from the system data repository.

Usage

- When **ra_truncateddlfilters** is invoked, it truncates all lists of filtered DDL commands in the system data repository that are older than the list version identified by the *locator* value.

ra_truncateusers

Truncates older versions of primary database users in the system data repository in the RASD.

Syntax

```
ra_truncateusers locator
```

Parameters

- **locator –** The log locator value (LTM Locator) that identifies the cutoff point for truncating older versions of database users from the system data repository.

Usage

- When **ra_truncateusers** is invoked, it truncates all non-current versions of all primary database users in the system data repository older than the version identified by the *locator* value.

If the current (most recent) version of a user is older than the version identified by the *locator* value, it is not truncated.

- The **ra_truncateusers** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *ra_truncatearticles* on page 113

ra_updatedevices

Updates information about primary database log devices in the RASD.

Syntax

```
ra_updatedevices
```

Usage

- When Automatic Storage Management (ASM) manages the redo log files and the disk group is changed by either adding or dropping disks, you must invoke the **ra_updatedevices** command to be sure the log device repository is updated with correct ASM storage information.
- When **ra_updatedevices** is invoked, Replication Agent:
  - Refreshes the archive log information
  - Deletes all of the data in its log device repository

    Note: If the device location is set, it is not overwritten.
  - Queries the primary database for information about all of its log devices
  - Re-populates the log device repository in the RASD with current information about primary database log devices returned by the primary database
- If any log device associated with the primary database is added, dropped, extended, or moved at the primary data server, you must:
  - Stop replication (using **quiesce** or **suspend**) to put the Replication Agent instance in Admin state
  - Invoke **ra_updatedevices** to update the log device repository in the RASD

  See the *Replication Agent Administration Guide* for more information.

  Note: The primary database need not be quiesced when you update the log device repository.
- If the primary data server writes to a new (or altered) log device before you update the log device repository, the Replication Agent instance stops replication processing and goes to Replication Down state.

  Coordinate all log device changes at the primary database with updating the Replication Agent log device repository.

- Because Replication Agent re-creates the entire log device repository when you invoke **ra_updatedevices**, any log device path that you modified previously (using **ra_devicepath**) is overwritten with the current log device information from the primary database.

  For example:

  ```
  ID=1 serverpath=/dev1 mirror=/dev1a
  ```

  becomes the following when you change the `server path` to "`dev44`":

  ```
  ID=1 serverpath=/dev44 mirror=/dev1a
  ```

  Note: If you need to alter the "default" path for a log device (that is, the log device path returned by the primary database), you must use the **ra_devicepath** command after you invoke **ra_updatedevices**.

- For each log device recorded in the RASD, you can set or change the disk device path with the **ra_devicepath** command.

  If you do not specify a disk device path (using **ra_devicepath**), the value recorded for the disk device path is **DEFAULT**, and Replication Agent uses the value recorded for the server device path to find the log device.
- The **ra_updatedevices** command is valid only when the Replication Agent instance is in the Admin or Replication Down state.
- Replication Agent uses the disk map file, to create mirror log devices, when log devices are created during transaction log initialization and when devices are updated using the **ra_updatedevices** command. When Replication Agent is in the Replicating state, it reads data from the mirrored disks specified in the map file.
  - The **ra_updatedevices** command updates the ASM disk map file. When executed the ASM disk map file is updated as follows:
    - ASM is queried to see if the disk groups required to read any redo logs have changed. If new disks have been added to any of the ASM disk groups, a default mirror entry is added in the ASM disk map file for the new disk.
    - The ASM disk group specified by the archive log path parameter is checked for new disks as well as the ASM disk group for each online redo log.
    - The ASM disk map file is updated before the log devices are updated to ensure any ASM disk path changes are included in updated log devices.
    - If new disk entries are added to the ASM disk map file, the log devices are not updated in the repository. A message is returned to the user, indicating that new entries are in the file that may need to be changed before devices are updated. The next time the **ra_updatedevices** command is executed, the log devices are updated.
  - The Replication Agent command **ra_helpdevice** provides device information for the log device status where the device is physically located and if it is being mirrored to another device. The physical information is a simple path to a file or raw device. ASM uses disk groups with potentially many disks, the physical information for ASM devices is provided to show all disks required for the device. There is one row of output for each disk in the group where the device is stored.

See also

- *ra_helpdevice* on page 85

ra_updateusers

Reloads user information from the primary database to the RASD.

Syntax

```
ra_updateusers
```

Usage

- **ra_updateusers** reloads user information to the RASD from the primary database. Use **ra_updateusers** when user information in the RASD becomes unsynchronized with the primary database.
- When you invoke **ra_updateusers**, Replication Agent:
  1. Deletes all user information from the RASD
  2. Queries the primary database for user information
  3. Repopulates the RASD with the user information returned from the primary database
- Use **ra_updateusers** only when Replication Agent is in the Admin or Replication Down state.

ra_version

Returns the version of the Replication Agent instance, the host operating system version, and the JRE version.

Syntax

```
ra_version
```

Usage

When **ra_version** is invoked, it returns the Replication Agent version string in one row:

```
Sybase Replication Agent for Unix &
Windows/15.7.0.6100/P/generic/JDK
7.0/main/6100/VM: Oracle Corporation
1.7.0_02/OPT/Wed Apr 15 06:38:13 MST
2012
```

See also

- *pdb_version* on page 61
- *ra_status* on page 112
- *ra_version_all* on page 118

ra_version_all

Returns the name, type, and version of the Replication Agent instance, and version information for the primary data server, primary Replication Server, and communications drivers.

Syntax

```
ra_version_all
```

Note: When Replication Agent is configured to connect to ASM (an **asm_tns_connection** is configured with a non-null value that is not the default value), **ra_version_all** includes an additional line of output that describes the version of ASM being connected to. When **asm_tns_connection** is not configured, no ASM entry is listed in **ra_version_all** output.

Usage

- When **ra_version_all** is invoked, it returns the following information:

```
Component             Version
-----------------------------------------------------------------
Instance:             rao_rac11r2 - Oracle
RepAgent:             Sybase Replication Agent for Unix &
                      Windows/15.7.1.6100/P/generic/JDK 7.0/main
                      /6100/VM: Oracle Corporation 1.7.0_02/OPT/
                      Mon Mar 05 16:54:15 MST 2012
JRE:                  Oracle Corporation Java(TM) SE Runtime
                      Environment/1.7.0_02-b13/Windows XP 5.1/
                      x86/32
RASD:                 SQL Anywhere/11.0.0.1264/WindowsXP
Primary Data Server:  Oracle Oracle Database 11g Enterprise
                      Edition Release 11.2.0.3.0 - 64bit
                      Production With the Partitioning, Real
                      Application Clusters, Automatic Storage
                      Management, OLAP, Data Mining and Real
                      Application Testing options
PDS JDBC Driver:      Oracle JDBC driver 11.2.0.3.0
ASM Server:           Oracle Oracle Database 11g Enterprise
                      Edition Release 11.2.0.3.0 - 64bit
                      Production With the Real Application
                      Clusters and Automatic Storage Management
                      options
RepServer:            Replication Server/15.6/P/NT (IX86)/Windows
                      2003/1/DEBUG/Thu Sep 16 14:03:14 2010
RSSD:                 SQL Anywhere/11.0.0.1264/WindowsXP
Sybase JDBC Driver:   jConnect (TM) for JDBC(TM)/7.07 GA(Build
                      26714)/P/EBF19793/JDK 1.6.0/jdbcmain/Tue
                      Feb 28 07:32:34 PST 2012
```

See also

- *pdb_version* on page 61

- *ra_status* on page 112
- *ra_version* on page 117

rasd_backup

Backs up the Replication Agent System Database (RASD).

Syntax

```
rasd_backup
```

Usage

- When **rasd_backup** is invoked, it starts the database backup process for the RASD.

  Note: Always back up the RASD before you truncate using **ra_truncatearticles** or **ra_truncateusers**.
- Replication Agent places RASD backup files in the directory identified by the **rasd_backup_dir** configuration parameter.

  When you create a Replication Agent instance, a RASD backup directory is created automatically as part of the instance directory structure. The default value of the **rasd_backup_dir** parameter points to that directory.
- The **rasd_backup** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *rasd_restore* on page 121
- *ra_truncatearticles* on page 113
- *ra_truncateusers* on page 114

rasd_helpbackup

Displays a list of RASD backups.

Syntax

```
rasd_helpbackup
```

Usage

- When **rasd_helpbackup** is invoked, it displays a list of RASD backups stored in the directory, identified by the **rasd_backup_dir** configuration parameter.
- The backups are named using the date and time the backup was created.

See also

- *rasd_restore* on page 121

- *rasd_backup* on page 119
- *rasd_removebackup* on page 120

rasd_removebackup

Removes RASD backups.

Syntax

```
rasd_removebackup [ backup_name | all ]
```

Parameters

- **backup_name –** The name of the backup that you are removing.
- **all –** A keyword that allows you to remove all RASD backups.

Examples

- **Example 1 –**

```
rasd_removebackup 2008-07-24_15.41.10

This command causes the backup named
2008-07-24_15.41.10 to be removed

rasd_removebackup 2008-07-24_15.41.10
go

RASD Backups removed
--------------------
2008-07-24_15.41.10
(1 row affected)
```

- **Example 2 –**

```
rasd_removebackup all
This command causes all of the backups to be removed.

rasd_removebackup all
2> go

RASD Backups removed
--------------------
2008-07-28_10.08.27
2008-07-28_10.09.29
2008-07-28_10.11.31
2008-07-28_10.20.55
(4 rows affected)
```

Usage

- When **rasd_removebackup** is invoked, it removes a RASD backup that is stored in the directory identified by the **rasd_backup_dir** configuration parameter.
- When **rasd_removebackup** is invoked with the **all** keyword, all RASD backups that are stored in the directory identified by the **rasd_backup_dir** configuration parameter are removed.
- When **rasd_backup** is invoked, the names of all backups removed appear.
- The backups are named using the date and time the backup was created.

See also

- *rasd_restore* on page 121
- *rasd_backup* on page 119
- *rasd_helpbackup* on page 119

rasd_restore

Allows you to restore the RASD.

Syntax

```
rasd_restore [backup_name]
```

Note: After executing **rasd_restore**, Replication Agent automatically shuts down if **rasd_restore** is successful.

Parameters

- **backup_name –** The name of the backup that you are restoring from. If you omit the backup name, the most recent backup is restored.

Examples

- **Example 1 –**

  ```
  rasd_restore
  ```

  This command with no parameters restores the RASD from the most recent backup.

- **Example 2 –**

  ```
  rasd_restore 2008-07-24_15.41.10
  ```

  This command restores the RASD from the 2008-07-24_15.41.10 backup.

Usage

- When **rasd_restore** is invoked, it starts the restore process for the RASD.

- When no parameters are used. Replication Agent looks for the most recent RASD backup in the directory identified by the **rasd_backup_dir** configuration parameter.
- If a backup name is provided as a parameter, Replication Agent restores from the specified backup in the directory identified by the **rasd_backup_dir** configuration parameter.
- When you create a Replication Agent instance, an RASD backup directory is automatically created as part of the instance directory structure. The default value of the **rasd_backup_dir** parameter points to that directory.
- If you invoke **rasd_restore** when the Replication Agent instance is in Replicating state, it returns an error.
- The **rasd_restore** command is valid only when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *rasd_backup* on page 119
- *rasd_helpbackup* on page 119
- *rasd_removebackup* on page 120

rasd_trunc_schedule

Manages a truncation schedule. **rasd_trunc_schedule** returns a list of the repository truncation schedule, and can also add and remove a specific schedule.

Syntax

```
rasd_trunc_schedule [ add, schedule | remove, schedule | clear |
force ]
```

Parameters

- **schedule –** The day and time string in the form of restricted UNIX cron style that indicates the time automatic repository truncation is to be performed.

  The following is a valid schedule string in UNIX cron style format:

  ```
  [mm][HH][DOM][MON][DOW]
  ```

  where:

  - mm – is the minutes past the hour.
  - HH – is the hour in 24-hour notation.
  - DOM – represents the days of the month, 2-digit number between **1** and **31,** which represents the day of the month.
  - MON – represents the month of the year, abbreviated in 3-character format, such as “Jan”, “Feb”, and so on, or a 2-digit number between 1 and 12, which represents the month in a year from January to December.

- DOW – represents the day of the week, abbreviated in 3-character format, such as “Sun”, “Sat”, and so on, or a 2-digit number between **1** and **7,** which represents the day in a week from Sunday to Saturday.
- Use an asterisk to match any valid value in a specific schedule field, ([mm],[HH], [DOM],[MON],[DOW]):
  - For example, “1720*** represents a daily schedule at 8:17 p.m.
  - When both the DOW and DOM are specified, the schedule represents two days that match either DOW or DOM.

    For example, “*1216*Mon” represents 12:00 a.m. every Monday or 12:00 a.m. the 16th of every month.
  - Multiple entries can be provided using a semicolon.

    For example, “*1216*Mon” or “1720***;*1216*Mon”.
  - Do not leave spaces between fields; otherwise, the schedule is rejected as an invalid schedule format. For example,“* 12 16 *Mon” is as an invalid schedule.
- Use the dash “-” operator to specify a range of values. For example, “1-6” represents the sequence “1,2,3,4,5,6.”
- Use the slash “/” operator to skip a given number of values. For example, “*/3” in the hour time field represents the sequence “0,3,6,9,12,15,18,21.”

- **clear –** To remove all repository truncation schedules, enter:

  ```
  rasd_trunc_schedule clear
  ```

  When the repository truncation schedule list is not set or empty, repository truncation by schedule is disabled.
- **force –** To perform an immediate repository truncation manually, regardless of the automatic truncation time schedule, enter:

  ```
  rasd_trunc_schedule force
  ```

Examples

- **Example 1 –**

  ```
  rasd_trunc_schedule
  ```

  This command returns a list of all repository truncation schedule times when repository truncation occurs.
- **Example 2 –**

  ```
  rasd_trunc_schedule add, 1720***
  ```

  This command adds daily repository truncation schedule at 8:17 PM to the schedule list.
- **Example 3 –**

  ```
  rasd_trunc_schedule remove, 1720***
  ```

This command removes the daily repository truncation schedule at 8:17 PM from the schedule list.

- **Example 4 –**

```
rasd_trunc_schedule clear
```

This command clears all repository truncation schedules that have been set.

- **Example 5 –**

```
rasd_trunc_schedule force
```

This command truncates the repository immediately, regardless of the existence of any truncation schedule.

Usage

- When **rasd_trunc_schedule** is invoked, its function is determined by the keywords and options you specify.
- When you specify multiple keywords and options, separate each must using a comma. A blank space before or after a comma is optional. For example:

```
rasd_trunc_schedule add, 1720***
```

See also

- *ra_truncatearticles* on page 113
- *ra_truncateusers* on page 114

resume

Starts replication processing in the Replication Agent instance.

Syntax

```
resume [ resync[, init] | purge ]
```

Parameters

- **resynch –** Replication Agent sends a **resync database** marker to Replication Server.
- **init –** This keyword is used only with the **resync** keyword. Replication Agent sends both a **resync database** marker and an initialization command to Replication Server.
- **purge –** Replication Agent sends a **purge** command to Replication Server when replication resumes.

Usage

- When **resume** is invoked, the Replication Agent instance attempts to go to Replicating state and start replication operations, as follows:

    - Replication Agent attempts to open network connections to the primary database, primary Replication Server, and RSSD.

      If it fails to establish a connection, Replication Agent logs a warning message in its system log, and it attempts to retry the connection, based on its configuration parameters for the connection.
    - If Replication Agent cannot establish a connection to the primary database after exhausting its configured retry attempts, it aborts all subsequent **resume** processing, returns to Replication Down state, and logs the error.
    - Replication Agent requests the current LTM Locator value from the primary Replication Server, and it stores the value in the Replication Agent transaction log.
    - The Log Reader component begins scanning the transaction log, looking for operations to be replicated. Log Reader begins scanning the log at the point identified by the LTM Locator value.
    - When it finds transactions to replicate, Log Reader passes them (as change-set data) to the input queue of the Log Transfer Interface component.
    - The Log Transfer Interface component reads the change-set data from its input queue, generates LTL commands, and places the LTL commands in its output queue for transmission to Replication Server.
- When **resume resync** is invoked, the Replication Agent instance attempts to start in the Replicating (Resynchronization) state.
    - Replication Agent sends a **resync database** marker to Replication Server, which processes this **resync database** marker and awaits a **dump database** marker from Replication Agent.
    - When **pdb_xlog init, force** is invoked before **resume resync**, Replication Agent sends instructions for Replication Server to purge all open transactions from the inbound queue and reset duplicate detection before receiving any new inbound transactions. Replication Server then awaits a **dump database** marker from Replication Agent.
    - When **resume resync, init** is invoked, Replication Agent sends a **resync database** marker and an initialization command instructing Replication Server to purge all open transactions from the inbound queue, reset duplicate detection, and suspend the outbound DSI. Use this option when you want to reload the primary database from the same dump as the replicate database.

  For more information about configuring database resynchronization, see the *Replication Server Administration Guide*.
- When **resume purge** is invoked, the Replication Agent instance sends a **purge** command to Replication Server when replication resumes. Replication Server will then purge data from the inbound queue for the connection to which this Replication Agent instance is connected. The **purge** keyword should be used only after downgrading to an earlier version of Replication Agent or when otherwise recommended by Sybase Technical Support.
- If any start-up operation fails, the Replication Agent instance returns to Replication Down state, and it logs the error.

- If the **resume** command is successful, the Replication Agent instance goes to Replicating state. To determine the current state of Replication Agent, use the **ra_status** command.
- The **resume** command returns an error under any of the following conditions:
  - The Replication Agent instance is already in Replicating state.
  - The system data repository in the RASD does not exist or is not initialized.
  - The Replication Agent connection configuration parameters are not set correctly, or it fails otherwise to connect with the primary database or the primary Replication Server.
  - The database connection for the primary database is not defined correctly in the primary Replication Server.
- If the **resume** command is successful, the Replication Agent instance goes into Replicating state.
- The **resume** command is valid only when the Replication Agent instance is in the Admin or Replication Down state.

See also

- *quiesce* on page 65
- *ra_status* on page 112
- *shutdown* on page 130
- *suspend* on page 131

rs_create_repdef

Creates a replication definition at Replication Server for a specific marked table and procedure, or for all marked tables and procedures.

Replication Agent is pre-configured to match replication definition datatypes available in Replication Server 15.0 and later. If replication definitions are to be generated against an earlier version of Replication Server, this configuration needs to be changed. Contact Sybase Technical Support for assistance in making this adjustment.

Syntax

```
rs_create_repdef {all | name}
```

Parameters

- **all –** A replication definition is created for all tables and procedures that are marked for replication.
- **name –** A replication definition is created for the table or procedure specified by *name*.

Note: rs_create_repdef always assumes that a database replication definition exists for the primary database.

Usage

- The **rs_username** user must have create object permission before Replication Agent can use it to create replication definitions from Replication Server. You must grant this permission manually from the RSSD.
- When a table is marked for replication and the owner mode is set to **on**, the replication definition created by **rs_create_repdef** includes the owner name as part of the table name for a table replication definition in the **with primary table named** clause.
- This command always assumes that a database replication definition exists for the primary database. All replication definitions created by **rs_create_repdef** include the **send standby** clause, which means the replication definition will only be used by Replication Server if there is already a database level replication definition. The replication definition created by **rs_create_repdef** cannot be individually subscribed to. If you do not wish to have a database level replication definition, you must use a different tool, or create replication definitions manually, and not use **rs_create_repdef**.
- Replication definitions created by **rs_create_repdef** always define the datatypes using available user defined datatypes that are installed in Replication Server. This means that customers using **rs_create_repdef** should not set Replication Agent configuration parameter **pdb_convert_datetime** to **true**, as doing so converts date and timestamp datatypes to Sybase format, instead of UDD format.
- Using the Replication Agent configuration parameter **pdb_auto_create_repdefs** has the same result as executing **rs_create_repdef**.
- When **rs_create_repdef** is invoked and the parameter "all" or "ALL" is entered, a replication definition is created for all tables or procedures that are marked for replication.
- When **rs_create_repdef** is invoked and the name of a table or procedure that is marked for replication is entered, a replication definition is created only for that table or procedure.
- For each table or procedure for which a replication definition create is attempted, a result set is returned. The result set contains the replication definition name and status of the create. If the replication definition was created, the status will be "created." If an error occurred, an error message from Replication Server will be returned.
- The character case of the object names in the replication definition will be set according to the **ltl_character_case** setting.
- The following applies to replication definition table and procedure names:
  - All non-alphanumeric characters and spaces are removed and are not part of the table or procedure name.
  - Underscores are kept as part of the name even though they are non-alphanumeric characters.
  - Periods are replaced with underscores.
- Replication definition names for tables always begin with the prefix "*ra$,*" followed by a unique alphanumeric identifier (maximum of 8 characters), and ending with a table or object name. For example, for a replicate name of "My Table," the resulting replication definition name is "ra$0x7952_mytable." For an especially long replicate name of "mytable8901234567890123456789O" (30 characters), the resulting replication definition name is "ra$0x7952_mytable8901234567890" (30 or 255 characters

maximum, depending on whether or not the **pdb_support_large_identifier** configuration parameter is set).

- For `date` columns, the **rs_create_repdef** command creates a replication definition with a column datatype defined that assumes the Replication Agent **pdb_convert_datetime** configuration parameter is set to **false**. If **pdb_convert_datetime** is set to **true**, the format of the `date` value does not match the format expected by Replication Server. To avoid this problem, change the **pdb_convert_datetime** configuration parameter to **false**, or manually create the replication definitions (without using the **rs_create_repdef** command).

See also

- *rs_drop_repdef* on page 128

rs_drop_repdef

A replication definition at the configured Replication Server for a table and procedure is dropped.

Syntax

```
rs_drop_repdef name
```

Parameters

- **name –** The name of a table or procedure.

Usage

- When **rs_drop_repdef** is invoked, a replication definition for that table is dropped at Replication Server.
- When **rs_drop_repdef** is invoked, a replication definition is dropped for that table or procedure.
- For each table or procedure for which a replication definition is dropped, a result set is returned. The result set contains the table name and status of the create. If the replication definition was created, the status will be "dropped." If an error occurred, an error message from Replication Server will be returned.
- The character case of the object names in the replication definition will be set according to the **ltl_character_case** setting.
- The following applies to replication definition table and procedure names:
  - All non-alphanumeric characters and spaces are removed and are not part of the table or procedure name.
  - Underscores are kept as part of the name even though they are non-alphanumeric characters.
  - Periods are replaced with underscores.

- Replication definition names for tables always begin with the prefix "*ra$,*" followed by a unique alphanumeric identifier (maximum of 8 characters), and ending with a table or object name. For example, for a replicate name of "My Table," the resulting replication definition name is "ra$0x7952_mytable." For an especially long replicate name of "mytable8901234567890123456789O" (30 characters), the resulting replication definition name is "ra$0x7952_mytable8901234567890" (30 or 255 characters maximum, depending on whether or not the **pdb_support_large_identifier** configuration parameter is set).

See also

- *rs_create_repdef* on page 126

rs_ticket

Supports Replication Server **rs_ticket** processing by placing an **rs_ticket** marker in the primary database transaction log. This command was created in support of the Replication Server **rs_ticket** feature.

Syntax

```
rs_ticket H1 [, H2[, H3 [, H4]]]
```

Parameters

- **H1, H2, H3 –** Each parameter contains from 1-10 characters. It is free form and is to be used as an identifier.
- **H4 –** It contains from 1-50 characters. It is free form and is also to be used as an identifier.

Examples

- **Example 1 –**

  The following executes **rs_ticket** and monitors the processing time for the record identified by the four parameters (only one parameter is required):

  ```
  rs_ticket test1, 1221, appxyz.monitoring_system
  ```

  Note: The parameters are optional, and can be used to identify or differentiate executions of **rs_ticket**.

  It can be used independently or grouped with additional executions to allow processing times to be compared.

  In this example, the following information will be sent to Replication Server.

  ```
  rs_ticket 'V=1;H1=test1;H2=1221;H3=appxyz;
  H4=monitoring_system;PDB(name)=hh:mm:ss.ddd'
  ```

  where "name" is the name of the primary database.

When **rs_ticket** reaches the replicate database, Replication Server will add additional time values for the EXEC, DIST and DSI components of Replication Server. The final result seen by the replicate database will look similar to:

```
rs_ticket 'V=1;H1=test1;H2=1221;H3=appxyz;
H4=monitoring_system;PDB(name)=hh:mm:ss.ddd;
EXEC=hh:mm:ss.ddd;DIST=hh:mm:ss.ddd;
DSI(name)=hh:mm:ss.ddd;RDB(name)=hh:mm:ss.ddd'
```

You can use the information provided to monitor replication latency and performance. By using different or descriptive H1-H4 parameters, users can more easily identify which **rs_ticket** data matches the activity or timing of the command when entered at the primary database.

- **Example 2 –**

  To measure performance of a batch of work, you can surround the work with **rs_ticket** executions, similar to the following sequence:

  (Execute in Replication Agent)

  ```
  rs_ticket start
  ```

  (Execute in primary data server)

  ```
  execute replication benchmarks
  ```

  (Execute in Replication Agent)

  ```
  rs_ticket stop
  ```

Usage

- The Replication Server EXEC, DIST, and DSI modules parse and process **rs_ticket** subcommands.
- There are no subscriptions for **rs_ticket**. DIST does not send **rs_ticket** to DSI unless there is at least one subscription from the replicate site.
- **rs_ticket** requires that the user name specified by **pds_username** be different from the user ID specified in the connection to Replication Server (the maintenance user). You can get the name of the maintenance user by executing **ra_maintid**.

See **rs_ticket** in the Replication Server documentation.

shutdown

Shuts down the Replication Agent instance, terminating its process.

Syntax

```
shutdown [immediate]
```

Parameters

- **immediate –** The optional keyword that shuts down the Replication Agent instance immediately.

Usage

- When **shutdown** is invoked with no option, Replication Agent starts a normal (graceful) shutdown.

  In a normal shutdown, Replication Agent first quiesces, and then the process terminates. See quiesce for more information about quiescing Replication Agent.
- When **shutdown** is invoked with the **immediate** keyword, Replication Agent starts an immediate shutdown.

  In an immediate shutdown, Replication Agent:
  - Stops all of its replication processing, without regard to transactions in process or in transit
  - Drops all of its connections
  - Terminates the application process
- The **shutdown** command with the **immediate** keyword is valid at any time, when the Replication Agent instance is in any state, including transition between states.
- The **shutdown** command with no keyword (normal shutdown) is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state, but not in state transition.

See also

- *quiesce* on page 65
- *ra_status* on page 112
- *resume* on page 124
- *suspend* on page 131

suspend

Stops all current replication processing and puts the Replication Agent instance into Admin state.

Syntax

```
suspend
```

Usage

- When **suspend** is invoked, it stops all current replication processing in the Replication Agent instance.

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

Note: The action of the **quiesce** command is similar to that of the **suspend** command, except that **quiesce** allows pending transactions in the Replication Agent internal queues to be processed first, before putting the Replication Agent instance in Admin state.

- If the Replication Agent instance is in Admin state, the **suspend** command returns an error.
- The **suspend** command is valid only when the Replication Agent instance is in Replicating state.

See also

- *quiesce* on page 65
- *ra_status* on page 112
- *resume* on page 124
- *shutdown* on page 130

test_connection

Tests Replication Agent connection configurations and network connectivity.

Syntax

```
test_connection [conn_name]
```

Note: When Replication Agent is configured to connect to ASM (an **asm_tns_connection** is configured with a non-null value that is not the default value), **test_connection** includes an additional line of output that describes the version of ASM being connected to. When **asm_tns_connection** is not configured, no ASM entry is listed in **test_connection** output.

Parameters

- **conn_name –** The keyword for a Replication Agent connection to be tested. Valid keywords are:
  - **PDS** – primary data server
  - **RS** – primary Replication Server (and RSSD, if so configured)

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

Examples

- **Example 1 –**

```
test_connection
```

  This command tests all Replication Agent connections, including the primary data server connection, the primary Replication Server connection, and the RSSD connection (if so configured).

- **Example 2 –**

```
test_connection PDS
```

  This command tests only the Replication Agent connection for the primary data server.

Usage

- When **test_connection** is invoked with no option, Replication Agent tests all of its connections by attempting to log in to the corresponding server for each connection, using the connection parameters stored in its configuration file.
- When **test_connection** is invoked with either of the keyword (**RS** or **PDS**), Replication Agent tests the specified connection.
- The **test_connection** command verifies both network connectivity and the following Replication Agent connection configuration parameters for the primary database:
  - connection type (connectivity driver and protocol) – **pds_connection_type**
  - database name – **pds_database_name**
  - data server name – **pds_server_name**
  - Data source name (ODBC drivers only) – **pds_datasource_name**
  - host machine name – **pds_host_name**
  - port number – **pds_port_number**
  - user login access – **pds_password** and **pds_username**

  Note: The **test_connection** command does not validate Replication Agent user login permissions in the primary database. It verifies only that the user login and password specified in the **pds_username** and **pds_password** parameters can log in to the primary data server.
- The **test_connection** command verifies both network connectivity and the following Replication Agent connection configuration parameters for the primary Replication Server (and RSSD, if so configured):
  - Database name – **rssd_database_name** (RSSD only)

- Replication Server data source (as specified in the Replication Server primary database connection) – **rs_source_db** and **rs_source_ds** (Replication Server only)
- Host machine name – **rs_host_name** (and **rssd_host_name**)
- Network packet size – **rs_packet_size** (Replication Server only)
- Port number – **rs_port_number** (and **rssd_port_number**)
- User login access – **rs_password**, **rs_username** (and **rssd_password** and **rssd_username)**

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

- The **test_connection** command returns the connection type and its status, as follows:

```
Type            Connection
----            ----------
PDS             succeeded
RS              succeeded

(2 rows affected)
```

  If the connection status is **failed**, it indicates one of the following:
  - The Replication Agent connection configuration parameters are not set correctly.
  - A network failure or communication error prevents the connection.
  - The server associated with the connection is down.
- If the connection status is **failed**, check the Replication Agent system log to determine the cause of the failure.

Note: You may also need to check the system log of the server associated with the connection to determine the cause of the failure.

- See the *Replication Agent Administration Guide* for information about setting up Replication Agent connection configuration parameters.
- See Configuration Parameters for information about specific connection configuration parameters.
- The **test_connection** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down states.

See also

- *Configuration Parameters* on page 139
- *ra_config* on page 70
- *ra_statistics* on page 104
- *ra_status* on page 112

trace

Returns current trace flag settings, or changes trace flag settings for the Replication Agent instance.

Syntax

```
trace [ {flag | all }, switch ]
```

Parameters

- **flag –** The name of the trace flag to change the setting for.
- **all –** A keyword that allows you to apply a switch value to all of the trace flags at once.
- **switch –** A Boolean (true or false) value that enables or disables tracing for the trace point identified in the *flag* option.

Usage

- The **trace** command is intended for use by Sybase Technical Support engineers when troubleshooting Replication Agent.
- When **trace** is invoked with no option, it returns the current settings for all Replication Agent trace flags.
- When **trace** is invoked with the *flag* and *switch* options, it changes the setting of the trace flag identified, and it returns the current (new) setting for the trace flag.
- When **trace** is invoked with the **all** keyword and a *switch* option, it sets all Replication Agent trace flags to the value specified in the *switch* option, and it returns the current (new) setting for all of the trace flags.
- Changes made with the **trace** command take effect immediately.
- When a trace flag is set to true, tracing is enabled for the trace points identified by the flag. When set to false, tracing is disabled for the trace points.
- Output from all trace points (except LTITRACELTL) is sent to the Replication Agent system log file. Use the **log_system_name** command to find the name and path of the Replication Agent system log file.
- Output from the LTITRACELTL trace point is sent to a separate trace output file named `LTITRACELTL.log`. To view the contents of the `LTITRACELTL.log` file, your file viewer must be capable of handling very long lines.

  Note: The `LTITRACELTL.log` file contains a human-readable representation of the LTL, not the actual LTL commands sent to the primary Replication Server.
- *Table 9. Replication Agent Trace Flags* on page 136 lists Replication Agent trace flags:

Table 9. Replication Agent Trace Flags

Trace Flag	Description
BMGRTRACE	When set to “true,” this flag enables Bean Management event tracing.
CACHETRC	When set to “true,” this flag enables tracing of internal cache events.
DBCONTEXT	When set to “true,” this flag turns on tracing of database context events.
LATRC	When set to “true,” this flag traces general Log Administrator operations.
LATRCSQL	When set to “true,” this flag traces SQL conversations between Log Administrator and the primary database.
LICTRACE	When set to “true,” this flag traces feature license check-in/checkout events.
LOGREADTRC	When set to “true”, turns on trace of database log reading.
LRTRACE	When set to “true,” this flag traces general execution of the Log Reader component.
LTITRACE	When set to “true,” this trace flag enables tracing operations of the Log Transfer Interface component.
LTITRACELTL	When set to “true,” this trace flag enables LTL statement tracing in the `LTITRACELTL.log` file.
LTMCI	When set to “true,” causes tracing of LTM component interface invocations and LTM invocations of other components' interfaces.
LTLFMTTRC	When set to “true,” this trace flag enables tracing of the LTL formatter.
LTMHL	When set to “true,” causes highlights in the LTM execution path to be noted.
LTMSC	When set to “true,” causes tracing of all Replication Agent state changes.
RACONTRC	When set to “true,” causes tracing of connection and query execution.
RACONTRCSQL	When set to “true,” causes tracing of SQL statements to be executed.
RASDTRC	When set to “true,” turns on tracing of Replication Agent System Data Repository events.
RATRACE	When set to “true,” causes tracing of Replication Agent events.
RSTICKETTRC	When set to “true,” causes Replication Agent to log trace message including the **rs_ticket** value to the Replication Agent system log during LTL formatting.
STMTRACE	When set to “true,” causes tracing of LTM state monitor events.

Trace Flag	Description
THREADTRC	When set to “true,” logs ThreadPool trace events.

- You cannot change the settings of SYSTEM trace flags.
  Table 10. Replication Agent SYSTEM Trace Flags on page 137 lists Replication Agent SYSTEM trace flags:

Table 10. Replication Agent SYSTEM Trace Flags

Trace Flag	Description
CONFIG	Configuration change event logged.
ERROR	Serious error; manual intervention may be needed to recover.
FATAL	Critical error; application shut down; manual intervention required to recover.
INFORMATION	Information only; no action required.
WARNING	Minor error; operation not affected, or problem is recoverable.

- The **trace** command is valid when the Replication Agent instance is in the Admin, Replicating, or Replication Down state.

See also

- *log_system_name* on page 8

Configuration Parameters

Configuration parameters record the user-configurable settings that control how a Replication Agent instance operates. The current values of all configuration parameters are stored in the configuration file of each Replication Agent instance.

Replication Agent Configuration File

The configuration file is created automatically when you create a Replication Agent instance. Each time a Replication Agent instance starts up, it reads the configuration file to get the configuration information needed to run.

After start-up, the only time the Replication Agent accesses the configuration file is when the **ra_config** or **ra_set_login** command is invoked to change the value of a configuration parameter. The configuration file resides in the instance subdirectory, under the Replication Agent base directory. The configuration file is named after the Replication Agent instance, with the extension `.cfg` (for example, if the instance is named "my_ra," the configuration file is `my_ra.cfg`).

When the value of a configuration parameter is changed, Replication Agent saves the new value, overwriting the entire configuration file.

Configuration File Format

The configuration file is a flat ASCII file that contains configuration information for a single Replication Agent instance.

The first two lines in the configuration file identify the file as a Replication Agent configuration file and record the time that the file was last modified. For example:

```
#RAO Property File
#Fri Jan 12 07:33:18 MST 2008
```

Each configuration parameter name appears on a separate line, followed by the equal symbol (=) and the current value of the parameter. For example:

```
compress_ltl_syntax=true
```

If the Replication Agent instance is not running, you can view the configuration file to examine the current Replication Agent configuration.

Note: Do not edit the configuration file, because Replication Agent overwrites the entire configuration file every time the **ra_config** or **ra_set_login** command is invoked to change a parameter value.

If the Replication Agent instance is running, use the **ra_config** command to view the current Replication Agent configuration.

Changing Configuration Parameters

To view, set, or change the current value of a Replication Agent configuration parameter, use the **ra_config** command.

To change the current Replication Agent administrator login (**ltm_admin_user**) or administrator password (**ltm_admin_pw**), you must use the **ra_set_login** command.

Note: You cannot directly use the **ltm_admin_user** and **ltm_admin_pw** parameters, and they do not appear in the parameter list returned by **ra_config**.

See Command Reference, for more information about using the **ra_config** and **ra_set_login** commands.

See also

- *Command Reference* on page 3
- *ra_config* on page 70
- *ra_set_login* on page 103

Copying a Replication Agent Configuration

When you create a new Replication Agent instance with the **ra_admin** utility, you can specify the new instance to use the same configuration parameter values as an existing Replication Agent instance.

Note: When you copy an existing configuration instance when creating a new Replication Agent instance, certain configuration parameter values are not copied to the new configuration. See the *Replication Agent Administration Guide* for more information.

If you do not copy an existing configuration when you create a new Replication Agent instance, the **ra_admin** utility creates a default configuration file, with default values for all configuration parameters.

Configuration Parameter Reference

The Replication Agent configuration parameters table lists all of the Replication Agent configuration parameters and a brief description of each parameter. See *Table 3. Password Parameters* on page 70 for a list and description of password parameters with their default values.

Table 11. Replication Agent Configuration Parameters

Parameter Name	Description
admin_port on page 148	Port number that Replication Agent will use to listen for administrative connections.
asa_password on page 148	Identifies the password for Replication Agent System Database (RASD).
column_compression on page 149	Use minimal column information in LTL.
compress_ltl_syntax on page 150	Use abbreviated LTL syntax.
connect_to_rs on page 150	Enable/disable connection from LTI to Replication Server.
ddl_password on page 151	Password for **ddl_username**.
ddl_username on page 151	The database user name included in LTL for replicating DDL commands to the replicate database.
dump_batch_timeout on page 152	Number of seconds to wait before sending an incomplete LTL buffer to Replication Server.
filter_maint_userid on page 153	Log Reader filters operations with maintenance user ID.
function_password on page 153	Password for user ID passed in LTL with replicated stored procedure invocations.
function_username on page 154	User ID passed in LTL with replicated stored procedure invocations.
log_backup_files on page 154	Determines the number of log backup files kept in the log directory.
log_directory on page 155	Directory where Replication Agent system log file is located.
log_trace_verbose on page 155	Switch on/off verbose mode in trace log file.
log_wrap on page 156	Number of 1KB blocks written to log file before wrapping.
lr_max_lobdata_cache on page 156	Maximum size of LOB data cache for off-row LOB data.
lr_max_op_queue_size on page 157	Maximum number of operations permitted in the log reader operation queue during replication.

Parameter Name	Description
lr_max_scan_queue_size on page 157	Maximum number of log records permitted in the log reader log scan queue during replication.
lr_ntext_byte_order on page 157	Specifies which byte order to use when replicating NCLOB.
lr_send_trunc_partition_ddl on page 158	Determines whether truncate partition commands are sent as DDL or DML to the replicate database.
lti_batch_mode on page 159	Switches on/off LTI batch mode.
ltl_formatter_count on page 160	Number of threads in the LTL formatter that work concurrently on items in the LTI queue.
lti_max_buffer_size on page 160	Maximum number of change sets stored in the LTI input buffer.
lti_update_trunc_point on page 161	Number of LTL commands sent before LTI requests new LTM Locator.
ltl_batch_size on page 162	Size of the LTL batch buffer.
ltl_big_endian_unitext on page 162	Specifies whether unicode LOB data should be converted from little endian to big endian before sending LTL to Replication Server.
ltl_character_case on page 163	Case of database object names sent to Replication Server.
ltl_origin_time_required on page 163	Specifies whether to send origin_time command tag in LTL.
ltl_send_only_primary_keys on page 164	Controls whether Replication Agent sends only primary key columns data for the *before* image for update and delete operations.
ltm_admin_pw on page 165	Password for Replication Agent administrative port.
ltm_admin_pw_min_length on page 166	The minimum length of the Replication Agent administrator login password.
ltm_admin_user on page 166	User ID for Replication Agent administrative port.
max_ops_per_scan on page 167	Maximum number of operations Log Reader will read in a single log scan.

Parameter Name	Description
pdb_archive_path on page 167	Identifies the directory path where Replication Agent expects to find archived Oracle redo log files.
pdb_archive_remove on page 168	Enables or disables the removal of archived transaction log files from the path specified by **pdb_archive_path**.
pdb_auto_create_repdefs on page 169	If set to true, when tables and procedures are marked for replication, a replication definition is automatically created at Replication Server for that table or procedure.
pdb_automark_tables on page 170	Determines if Replication Agent automatically marks tables for replication during initialization or DDL replication.
pdb_auto_run_scripts on page 171	Automatic execution of SQL scripts used to create/remove transaction log objects and mark/unmark primary database objects.
pdb_convert_datetime on page 172	Converts native date/time formats to Sybase `datetime` format.
pdb_dflt_column_repl on page 173	Enables replication for LOB columns by default when table is marked.
pdb_dflt_object_repl on page 174	Enables replication by default when object is marked.
pdb_ignore_unsupported_anydata on page 175	Determines whether or not Replication Agent ignores data of unsupported datatypes stored in columns of type `ANYDATA`.
pdb_include_archives on page 176	Enables or disables the use of Oracle archive log files.
pdb_skip_missing_user on page 176	Determines whether or not Replication Agent skips the processing of any command for which there is no matching session or user information.
pdb_support_large_identifier on page 177	To support replication of large identifiers up to 255 characters in length with Replication Server 12.6 or later.

Parameter Name	Description
pdb_timezone_file on page 177	Specifies the file to read at Replication Agent initialization to obtain Oracle time zone information.
pdb_xlog_device on page 178	Name of the primary database device.
pdb_xlog_prefix on page 178	Character string prefix used to identify transaction log objects.
pdb_xlog_prefix_chars on page 179	Non-alphabetic characters allowed in **pdb_xlog_prefix**.
pds_connection_type on page 180	Type of connection to primary data server.
pds_database_name on page 181	Name of database replicated from the primary data server.
pds_host_name on page 181	Name of primary data server host machine.
pds_password on page 182	Password for user ID that Replication Agent uses to access the primary data server.
pds_port_number on page 182	Port number for the primary data server.
pds_retry_count on page 182	Number of times to retry connection to primary data server.
pds_retry_timeout on page 183	Number of seconds to wait between connection retry attempts.
pds_ssl_sc_dn on page 183	The distinguished name (DN) of the primary data server certificate.
pds_tns_connection on page 184	Oracle connection name found in the `tnsnames.ora` file which identifies the connection information for the primary database.
pds_tns_filename on page 184	Oracle file name identifying the Oracle `tnsnames.ora` file to be used to identify the connection information for the primary database.
pds_use_ssl on page 185	Indicates whether to use SSL to connect to the primary data server.
pds_username on page 185	User ID that Replication Agent uses to access primary data server.
ra_admin_device on page 186	The primary database device on which Replication Agent system objects are created.

Parameter Name	Description
ra_admin_instance_prefix on page 186	The prefix string used to identify Replication Agent system objects specific to one Replication Agent instance.
ra_admin_prefix on page 187	The prefix string used to identify shared Replication Agent system objects.
ra_admin_prefix_chars on page 188	Non-alphabetic characters that are allowed in the database object name prefix string that identifies Replication Agent system objects.
ra_admin_owner on page 188	The owner of all Replication Agent system objects, including shared and instance-specific system objects.
ra_retry_count on page 189	Number of times LTM attempts to get back to Replicating state after a failure.
ra_retry_timeout on page 189	Number of seconds to wait between LTM attempts to get back to Replicating state.
rasd_backup_dir on page 190	The directory path for Replication Agent System Database (RASD) backup files.
rasd_database on page 190	The directory path for the Replication Agent System Database (RASD) database file.
rasd_mirror_tran_log on page 191	Enables or disables Replication Agent System Database (RASD) transaction log mirroring.
rasd_trace_log_dir on page 192	The directory path for the Replication Agent System Database (RASD) trace log file.
rasd_tran_log on page 192	The directory path for the Replication Agent System Database (RASD) transaction log file.
rasd_tran_log_mirror on page 193	The directory path for the Replication Agent System Database (RASD) transaction log file mirror.
ra_standby on page 194	Determines whether or not Replication Agent functions in standby mode.
ra_statrack_interval on page 195	Determines the interval, in seconds, at which statistics are sampled.

Parameter Name	Description
rman_enabled on page 195	Indicates whether or not Replication Agent uses the Oracle **RMAN** utility to truncate old archive log files.
rman_password on page 196	Password used with **rman_username** to connect to the Oracle **RMAN** utility.
rman_username on page 196	Login name used with **rman_password** to connect to the Oracle **RMAN** utility.
rs_charset on page 197	Character set used to communicate with Replication Server.
rs_host_name on page 198	Name of primary Replication Server host machine.
rs_packet_size on page 198	Network I/O packet size for data sent to Replication Server.
rs_password on page 199	Password for user ID Replication Agent uses to access Replication Server.
rs_port_number on page 199	Port number for primary Replication Server.
rs_replicate_owner_required on page 200	Indicates if the owner is always included with the replicate table clause when generating replication definitions.
rs_retry_count on page 200	Number of times to retry connection to primary Replication Server.
rs_retry_timeout on page 200	Number of seconds to wait between connection retry attempts.
rs_source_db on page 201	Name of primary database identified to Replication Server.
rs_source_ds on page 201	Name of primary data server identified to Replication Server.
rs_ssl_sc_dn on page 202	The distinguished name (DN) of the Replication Server certificate and is only valid if **rs_use_ssl** is set to true.
rs_ticket_version on page 202	Determines whether Replication Agent records the primary database time or the primary database date and time into the **rs_ticket** marker.

Parameter Name	Description
rs_use_ssl on page 202	Indicates whether Replication Agent (as a client) should use SSL to connect to Replication Server.
rs_username on page 203	User ID that Replication Agent uses to access primary Replication Server.
rssd_charset on page 203	Character set used to communicate with RSSD.
rssd_database_name on page 204	Name of RSSD database.
rssd_host_name on page 204	Name of RSSD host machine.
rssd_password on page 205	Password for user ID that Replication Agent uses to access RSSD.
rssd_port_number on page 205	Port number for RSSD.
rssd_username on page 206	User ID that Replication Agent uses to access RSSD.
scan_fetch_size on page 206	Number of rows to fetch from the primary database when Oracle LogMiner is scanning the log in each network round-trip.
scan_sleep_increment on page 207	Number of seconds to increase Log Reader wait before next scan after finding no operations to replicate.
scan_sleep_max on page 207	Maximum number of seconds for Log Reader to wait before next scan after finding no operations to replicate.
skip_lr_errors on page 208	Determines whether Replication Agent ignores log record processing errors.
skip_ltl_errors on page 208	LTI ignores error messages returned by Replication Server.
ssl_identity_filename on page 209	Indicates the path to the Replication Agent instance identity file, a PKCS #12 file containing an asymmetric key pair used for SSL communication.
ssl_identity_password on page 209	The passsword to access a Replication Agent instance identity file.
ssl_certificates_filename on page 210	Indicates the path of the file containing Certificate Authority (CA) certificates included with the Replication Agent installation.

Parameter Name	Description
structured_tokens on page 210	LTI uses structured tokens when generating LTL output.
truncation_interval on page 211	Number of minutes to wait between automatic log truncations.
truncation_type on page 211	Methods of log truncation allowed.
use_rssd on page 212	Switches on/off access to RSSD for replication definitions.
use_ssl on page 213	Indicates whether clients must use SSL to connect to Replication Agent.

admin_port

The client socket port number of Replication Agent.

Default
10000

Value
A valid port number on the Replication Agent host machine.

Comments

- When you create a Replication Agent instance, you must specify a client socket port number for the instance administration port. Client applications use this port number to connect to the Replication Agent instance.
- You must specify a port number that does not conflict with any port numbers already in use on the Replication Agent host machine.
- If you change the value of the **admin_port** parameter with the **ra_config** command, the new value is recorded in the configuration file immediately, but you must shut down and restart the Replication Agent instance to make the new port number take effect.
- After you change the value of the **admin_port** parameter with the **ra_config** command, the next time you log in to the Replication Agent administration port, you must use the new port number.

asa_password

Identifies the password for Replication Agent System Database (RASD).

Default
Not configured.

Value
A valid password.

Comments

- The value of the **asa_password** parameter must adhere to the ASA password policy. Passwords are case sensitive and they cannot:
  - begin with white space, single quotes, or double quotes
  - end with white space
  - contain semicolons
  - be longer than 255 bytes in length
- The value of the **asa_password** parameter is the password for the ASA database user name specified in the **asa_username** parameter.
- The value of the **asa_password** parameter is encrypted in the Replication Agent configuration file.
- Changing the **asa_password** parameter by using **sp_password** affects both configuration file and the ASA database.

column_compression

Determines whether the Log Transfer Interface component sends all columns in row after images, or only the columns that changed in an **update** operation.

Default
true

Values
true – enables minimal column information (only changed columns in row after images) in Log Transfer Language (LTL) for **update** operations.

false – disables minimal column information in LTL for **update** operations.

Comments

- When the **column_compression** parameter is set to **false**, the LTI component sends complete row after images in LTL, including columns in which no data changed as a result of an **update** operation.
- When the **column_compression** parameter is set to **true**, the LTI component sends minimal column information in the row *after* images in LTL, with only the columns that changed as a result of an **update** operation. Columns in which no data changed as a result of the **update** are not sent in LTL.
- In general, setting the value of the **column_compression** parameter to **true** provides better Replication Agent throughput.
- If your Replication Agent instance is configured to send minimal column data—**column_compression** and **ltl_send_only_primary_keys** are set to true—some column

data may be omitted for columns that are specified as searchable in a replication definition. Consequently, errors may occur at a subscribing database where data needed for an insert, subscription migration, or custom function string is missing. Sybase therefore recommends that you enable autocorrection for any table referenced in a replication definition with searchable columns.

See also

- *ltl_send_only_primary_keys* on page 164

compress_ltl_syntax

Determines whether the Log Transfer Interface component compresses Log Transfer Language (LTL) commands using abbreviated syntax.

Default
true

Values
true – enables LTL compression, using abbreviated LTL syntax.

false – disables LTL compression.

Comments

- Setting the value of the **compress_ltl_syntax** parameter to **true** will provide better Replication Agent throughput.
- See the *Replication Agent Administration Guide* for more information about LTL commands and abbreviated LTL syntax.

connect_to_rs

Enables or disables the network connection to the primary Replication Server.

Default
true

Values
true – enables the network connection to Replication Server.

false – disables the network connection to Replication Server.

Comments

- When the value of the **connect_to_rs** parameter is false, the network connection from Replication Agent to Replication Server is disabled, and no replication can occur.

- When the network connection to Replication Server is disabled by the **connect_to_rs** parameter, the Replication Agent instance can still go to Replicating state, with the following limitations:
  - A “dummy” connection in Replication Agent emulates a real connection to Replication Server.
  - The value of the LTM Locator stored in the Replication Agent transaction log is set to zero.
  - The maintenance user name is set to an invalid user ID.

  Note: maintenance user operations cannot be filtered when the value of the **connect_to_rs** parameter is false.
- You can use the **connect_to_rs** parameter to temporarily disable the network connection to Replication Server for testing.
- When the value of the **connect_to_rs** parameter is false, you can put the Replication Agent instance in Replicating state, set the value of the LTITRACELTL trace flag to true, and view a readable representation of the LTL that would have been sent to Replication Server if the connection had not been disabled.
- During normal Replication Agent operation, the value of the **connect_to_rs** parameter must be true.

ddl_password

Updates the log device repository in the RASD. Identifies the password for **ddl_username**.

Default
"" (empty string)

Value
A valid password.

Comments

- The value of the **ddl_password** parameter can be up to 30 characters.
- The value of the **ddl_password** parameter is the password for the database user name specified in the **ddl_username** parameter.
- The value of the **ddl_password** parameter is encrypted in the Replication Agent configuration file.

ddl_username

The database user name included in LTL for replicating DDL commands to the standby database. This user must have permission to execute all replicated DDL commands at the standby database.

Default
Not configured.

Value
A valid user name in the standby database.

Comments

- The value for the **ddl_username** must not be the same as the value of the maintenance user defined in Replication Server for the standby connection. Failure to provide different names results in a Replication Server error.
- The value of the **ddl_username** parameter is sent in the LTL for all replicated DDL statements.
- The value of the **ddl_password** parameter is the password for the database user name specified in the **ddl_username** parameter.
- When DDL is replicated, Replication Server will connect to the replicate database using the **ddl_username** and **ddl_password**.
- For Oracle, Replication Server issues the following message:
  `ALTER SESSION SET CURRENT_SCHEMA=`*`user`*
  where *user* is the user ID that generated the DDL operation at the primary database. The actual DDL command is then executed against the replicate database. If the **ddl_username** does not have permission to issue **ALTER SESSION SET CURRENT_SCHEMA** or to execute the DDL command against the *user* schema, the command fails.

dump_batch_timeout

Specifies the number of seconds to wait before sending the contents of the Log Transfer Interface (LTI) buffer to Replication Server, even if the buffer is not full.

Default
5

Value
An integer from 1 to 60.

Comments

- The value of the **dump_batch_timeout** parameter is the number of seconds from the time the previous LTI buffer was sent to Replication Server until the next buffer will be sent.
- The **dump_batch_timeout** parameter has no effect if the value of the **lti_batch_mode** parameter is **false**.

filter_maint_userid

Determines whether operations applied by the maintenance user are ignored.

Default
true

Values
true – enables the Log Reader to ignore maintenance user operations.

false – disables the Log Reader filter to allow replicating maintenance user operations.

Comments

- The **filter_maint_userid** configuration parameter is provided to support bidirectional replication, in which the primary database also serves as a replicate database that has transactions applied to it by a Replication Server maintenance user.
- If the value of the **filter_maint_userid** parameter is true, database operations applied by the maintenance user are not replicated. The Log Reader component filters out (ignores) operations applied by the maintenance user when it reads the transaction log.
- If the value of the **filter_maint_userid** parameter is false, database operations applied by the maintenance user are replicated. The Log Reader component replicates all operations on marked objects, regardless of the user that applied the operation.
- The maintenance user login is specified when the database connection for the primary database is created in Replication Server.

function_password

The password included in Log Transfer Language for replication of “request” stored procedures.

Default
"" (empty string)

Values
A valid password.

Comments

- The value of the **function_password** parameter can be up to 30 characters.
- The value of the **function_password** parameter is the password for the database user name specified in the **function_username** parameter.
- The value of the **function_password** parameter is encrypted in the Replication Agent configuration file.

- For more information about “request” stored procedures, see Replication Server documentation.

function_username

The database user name included in Log Transfer Language (LTL) for replication of “request” stored procedures.

Default
sa

Values
A valid user name in the primary database.

Comments

- The value of the **function_username** parameter is sent in the LTL for all replicated stored procedures in the primary database.
- The value of the **function_password** parameter is the password for the database user name specified in the **function_username** parameter.
- For more information about “request” stored procedures, see Replication Server documentation.

log_backup_files

The number of backup log files kept in the Replication Agent instance `log` directory.

Default
3

Values
An integer greater than or equal to 1.

Comments

- When the system log wraps, Replication Agent copies the current log file to a backup file, with a generated number appended to the file name.
  For example, if the system log file is named `my_ra.log`, the first backup file created when the system log wraps would be named `my_ra1.log`. The second backup file created would be named `my_ra2.log`, and so on.
- When the number of backup files exceeds the value of the **log_backup_files** parameter, the oldest backup file (that is, the one with the lowest generated number) is deleted from the `log` directory before the next backup file is created.

log_directory

The directory for Replication Agent system log files.

Default

The path to the `log` directory created when the Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

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

  where:
  - *%SYBASE%* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.
- On UNIX platforms:

  ```
  $SYBASE/RAX-15_5/inst_name/log
  ```

  where:
  - *$SYBASE* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.

Value

A valid path on the Replication Agent host machine.

Comments

- When a Replication Agent instance is created, the `log` directory is created as part of the instance directories. The default value of the **log_directory** parameter points to that directory.
- If you specify any valid path as the value of the **log_directory** parameter, the Replication Agent instance places its system log files in the directory you specify the next time it is started.
- If you specify the default value of the **log_directory** parameter by using the **default** keyword in the **ra_config** command, then the next time it is started, Replication Agent will place its system log files in the `log` directory that was created when the Replication Agent instance was created.
- If you change the value of the **log_directory** parameter with the **ra_config** command, the new value is recorded in the configuration file immediately, however, you must shut down and restart the Replication Agent instance to make the new value take effect.

log_trace_verbose

Enables or disables additional diagnostic information in Replication Agent system log files.

Default

false

Values
true – enables detailed diagnostic information in log files.

false – disables detailed diagnostic information in log files.

Comment
Detailed diagnostic information is intended for troubleshooting only, with assistance from Sybase Technical Support.

log_wrap

The maximum size, in 1K blocks, of the Replication Agent system log file before wrapping.

Default
1000

Value
An integer greater than or equal to 1000.

Comments

- The value of the **log_wrap** parameter is the number of 1KB blocks written by Replication Agent, before it wraps the system log file.
- Larger values for the **log_wrap** parameter allow more log history in each file. Smaller values produce smaller log files.
- When the log file wraps, Replication Agent copies the current log file to a backup file, with a generated number appended to the file name.

  For example, if the system log file is named `my_ra.log`, the first backup file created when the system log wraps would be named `my_ra1.log`. The second backup file created would be named `my_ra2.log`, and so on.
- When the number of backup files exceeds the value of the **log_backup_files** parameter, the oldest backup file (that is, the one with the lowest generated number) is deleted from the `log` directory before the next backup file is created.

lr_max_lobdata_cache

The maximum size in bytes of LOB data cache for off-row LOB data.

Default
8192

Values
An integer from 8192 to 2147483647.

Comments

- Use **lr_max_lobdata_cache** to tune Replication Agent performance for replicating LOB data.
- For best performance, set **lr_max_lobdata_cache** to a value larger than the maximum size of LOB data to be replicated. For example, if your LOB data varies from 10KB to 120KB, set **lr_max_lobdata_cache** to a value greater than 120KB.

lr_max_op_queue_size

The maximum number of operations permitted in the log reader operation queue during replication.

Default
1000

Values
An integer from 25 to 2^{31} - 1.

Comments

- The **lr_max_op_queue_size** parameter can be used to tune Replication Agent performance.

lr_max_scan_queue_size

The maximum number of log records permitted in the log reader log scan queue during replication.

Default
1000

Values
An integer from 25 to 2^{31} - 1.

Comments

- The **lr_max_scan_queue_size** parameter can be used to tune Replication Agent performance.

lr_ntext_byte_order

Ensures the byte order of NCLOB data is sent correctly to the replicate database.

The correct byte order is necessary when you are replicating with different primary and replicate database types , or primary and replicate databases that are on different platforms (little endian Windows to big endian UNIX).

Default
big

Values
big – big endian

little – little endian

Example
When replicating Oracle to Oracle, you need to know the endian for the replicate database and set the parameter so the correct endian is sent to the replicate. The same applies when replicating to databases other than Oracle.

Comments

- If **lr_ntext_byte_order** is set to **little**, **ltl_big_endian_unitext** must be set to **false**.
- Big endian indicates a left-to-right byte-order architecture; little endian indicates a right-to-left byte-order architecture.
- The default behavior of Replication Agent is to force any unicode data to big endian order, as defined by configuration parameter **ltl_big_endian_unitext**. In order to allow configuration parameter **lr_ntext_byte_order** to successfully override the SQL Server byte order, you must also set **ltl_big_endian_unitext** configuration parameter to **false** whenever the **lr_next_byte_order** parameter is used.
- The following describes the relationship between **ltl_big_endian_unitext** and **lr_ntext_byte_order** configuration parameters:
  - When **ltl_big_endian_unitext** is set **true** by default, Replication Agent ensures that all unicode data is sent in big endian order.
  - When set to false, **ltl_big_endian_unitext** allows unicode data to be sent in the byte order that is used when the data is stored in the transaction log file
  - In contrast, **lr_ntext_byte_order**, forces the result of Unicode data read from the transaction log to be in the requested byte order, regardless of how it normally exists in the transaction log file.

See also

- *ltl_big_endian_unitext* on page 162

lr_send_trunc_partition_ddl

Determines whether **truncate partition** commands are sent as DDL or DML to the replicate database.

Default
true

Values
true – the **truncate partition** command is sent as a DDL command (**alter table**).

false – the **truncate partition** is sent as a DML operation.

Comments

- If set to true, the **truncate partition** command is sent as a DDL command. Normally, it is set to replicate to Oracle.
- Set the value to false when replicating to databases that treat **truncate partition** commands as DML.

lti_batch_mode

Enables or disables the LTL batch mode for the Log Transfer Interface component.

Default
true

Values
true – enables LTL batch mode.

false – disables LTL batch mode.

Comments

- If the value of the **lti_batch_mode** parameter is true, the LTI component sends LTL commands to Replication Server in batches, instead of one command at a time:
  - The LTI component fits as many LTL commands as it can into its LTL batch mode buffer, before it sends any commands to Replication Server.
  - When the time interval specified in the **dump_batch_timeout** parameter expires, the LTI component sends the current LTL batch mode buffer contents to Replication Server, even if the buffer is not full.
- If the value of the **lti_batch_mode** parameter is false, the LTI component sends individual LTL commands to Replication Server for each change set in its input queue.
- When Replication Agent connects to Replication Server, it determines the version of Replication Server:
  - If the Replication Server version is earlier than 12.5, the size of the LTL batch mode buffer is set to 16KB automatically.
  - If the Replication Server version is 12.5 or later, Replication Agent sets the size of the LTL batch mode buffer to the size specified by the **ltl_batch_size** parameter.
- If the Replication Server version is 12.5 or later, you can use the Replication Agent **ltl_batch_size** parameter to set the size of the LTI component LTL batch mode buffer.

Note: Adjusting the size of the LTL batch mode buffer can help you optimize the performance of the replication system.

- If the Replication Server version is earlier than 12.5 and the value of the **lti_batch_mode** parameter is true, if any single LTL **distribute** command exceeds the 16K size of the LTL batch mode buffer, Replication Server returns an error and Replication Agent goes into Replication Down state.
- In general, setting the value of the **lti_batch_mode** parameter to true provides better Replication Agent throughput.

lti_formatter_count

The number of threads in the LTL formatter that work concurrently on items in the LTI queue. Adjust the value of this parameter according to the number of processors available to Replication Agent in the machine on which it is installed.

Default
3

Value
1–200

Comments

- Each thread specified by **lti_formatter_count** works on a separate item in the LTI queue.
- To determine if performance may be improved by increasing the value of **lti_formatter_count**, examine the LTI statistics for "Current number of commands in the LTI queue" and "Current number of unformatted commands in the LTI queue." When the number of commands in the LTI queue is at or near capacity while the number of unformatted commands is closer to capacity than to zero, increasing the value of **lti_formatter_count** may improve Replication Agent performance.
- You can change **lti_formatter_count** only when Replication Agent is in the Admin or Replication Down state.
- You can improve Replication Agent performance on a machine with multiple processors by adjusting the number of LTL formatter threads working in parallel.

lti_max_buffer_size

The maximum number of items that can be stored in the LTI queue. Set this parameter according to the number of LTL formatter process threads and the number of processors available.

Default
5000

Value
An integer in the range of 1000 to 100000.

Comments

- The **lti_max_buffer_size** specifies the maximum number of items that the log reader can place in the LTI queue.
- Setting **lti_max_buffer_size** to a value that is too large may degrade performance if there is insufficient memory available.
- You can change **lti_max_buffer_size** only when Replication Agent is in the Admin or Replication Down state.
- You can improve Replication Agent performance on a machine with multiple processors by adjusting the size of the LTI queue with **lti_max_buffer_size**.

lti_update_trunc_point

The number of Log Transfer Language (LTL) commands sent before requesting a new LTM Locator from the Replication Server.

Default
10000

Value
An integer from 1 to 100000.

Comments

- The value of the **lti_update_trunc_point** parameter is the number of LTL commands that Replication Agent sends to, before it requests a new LTM Locator (secondary truncation point) from the Replication Server.
- Lower numbers cause Replication Agent to request a new LTM Locator from more often.
- If the value of the **truncation_type** parameter is **locator_update**, setting the value of the **lti_update_trunc_point** parameter to a lower number causes automatic log truncation to occur more frequently.
- The value of the **lti_update_trunc_point** parameter is a trade-off between better system performance and longer recovery time:
  - Lower values reduce the time it takes to recover from a replication failure, but they may have an adverse affect on overall system throughput.
  - Higher values improve overall system throughput, but they may increase the time it takes to recover from a replication failure.
- If Replication Agent is operating in an unreliable network environment, it may be prudent to set the **lti_update_trunc_point** parameter to a lower value to ensure faster recovery.

ltl_batch_size

The size in bytes of the Log Transfer Interface component Log Transfer Language (LTL) batch mode buffer.

Default
40000

Value
An integer from 16384 to 10485760.

Comments

- The value of the **ltl_batch_size** parameter is the size (in bytes) of the LTI component LTL batch mode buffer.
- When Replication Agent connects to Replication Server, it determines the version of the Replication Server:
  - If the Replication Server version is earlier than 12.5, the size of the LTL batch mode buffer is set to 16K automatically, and the value of the **ltl_batch_size** parameter is ignored.
  - If the Replication Server version is 12.5 or later, Replication Agent sets the size of the LTL batch mode buffer to the size specified by the **ltl_batch_size** parameter.
- The Log Transfer Interface component uses the LTL batch mode buffer only if the value of the **lti_batch_mode** parameter is **true**. If the value of the **lti_batch_mode** parameter is **false**, the LTL batch mode buffer is not used.

ltl_big_endian_unitext

Specifies whether "unitext" data is converted from *little endian* to *big endian* before sending LTL to Replication Server.

Default
true

Values
true – Unitext data that is in little endian byte order will be changed to big endian byte order.

false – Unitext data byte order is not changed.

Comments
When setting this parameter, you must know how the **lr_ntext_byte_order** is set. If parameter **lr_ntext_byte_order** is set to send the correct byte order for the replicate database, then **ltl_big_endian_unitext** must be set to false so the byte order will not be changed.

See also

- *lr_ntext_byte_order* on page 157

ltl_character_case

The character case used for database object names in Log Transfer Language (LTL) sent to Replication Server.

Default
asis

Values
asis – database object names are sent in the same character case as they are returned from the primary database, or (if the value of the **use_rssd** parameter is true) in the same character case as they are specified in replication definitions.

lower – database object names are sent in all lowercase, regardless of how they are returned from the primary database, or specified in replication definitions.

upper – database object names in LTL are sent in all uppercase, regardless of how they are returned from the primary database, or are specified in replication definitions.

Comments

- The **ltl_character_case** configuration parameter allows you to customize the handling of database object names in LTL to work with replication definitions that specify the object names differently than the way the primary database returns them.
- If the value of the **ltl_character_case** parameter is **asis**, and the value of the **use_rssd** parameter is true, database object names are sent in the same character case as they are specified in replication definitions.
- If the value of the **ltl_character_case** parameter is **asis**, and the value of the **use_rssd** parameter is false, database object names are sent in the same character case as they are returned from the primary database.
- If replication definitions specify database object names in all lowercase, set the value of the **ltl_character_case** parameter to **lower**.
- If replication definitions specify database object names in all uppercase, set the value of the **ltl_character_case** parameter to **upper**.
- If you want to send database object names with “mixed” character case (for example, *MyTable*), set the value of the **ltl_character_case** parameter to **asis**.

ltl_origin_time_required

Enables or disables the Log Transfer Language (LTL) **origin_time** command tag.

Default
false

Values
true – enables the **origin_time** command tag in LTL.

false – disables the **origin_time** command tag in LTL.

Comments

- If the value of the **ltl_origin_time_required** parameter is true, the Log Transfer Interface component includes the **origin_time** command tag in the LTL it generates.
- If a Replication Server function string checks for the **origin_time** command tag, set the value of the **ltl_origin_time_required** parameter to true.
- The `datetime` value placed in the LTL **origin_time** command tag is the time that the original primary database operation was recorded in the transaction log, not the time it was scanned and processed by the Log Reader component.
- Setting the value of the **ltl_origin_time_required** parameter to false provides better Replication Agent throughput.
- If you use Replication Manager to report latency, you must set the value of the **ltl_origin_time_required** parameter to true.

ltl_send_only_primary_keys

Determines whether Replication Agent sends only before image primary key columns, or sends all before image columns to Replication Server for update and delete operations to the replicate database.

Default
true

Value
true – sends only the before image primary key columns to Replication Server.

false – sends all before image columns to Replication Server.

Comments

- When set to true and a replication definition exists that identifies the primary key column(s) for a table, only the primary key column value(s) are sent for the before image in update and delete operations. Sending only primary key column data reduces the amount of data sent to Replication Server, since only primary keys are used to construct the “where” clauses for update and delete operations.
- When set to false, before image values are sent for all columns available, regardless of primary key definition.
- Setting of **ltl_send_only_primary_keys** to false, is only recommended when additional before image values provide benefit, such as for supporting custom function strings at the

Replicate database, or for resolving other issues where additional column data provides benefit.

- Primary keys are defined within a table level Replication Definition. If configuration **use_rssd** is set to false, the setting of **ltl_send_only_primary_keys** has no impact, since Replication Definition information will not be gathered from the Replication Server System Database (RSSD).
- If you set the value to false, the performance will be slower.
- If your Replication Agent instance is configured to send minimal column data—**column_compression** and **ltl_send_only_primary_keys** are set to true—some column data may be omitted for columns that are specified as searchable in a replication definition. Consequently, errors may occur at a subscribing database where data needed for an insert, subscription migration, or custom function string is missing. Sybase therefore recommends that you enable autocorrection for any table referenced in a replication definition with searchable columns.

See also

- *column_compression* on page 149

ltm_admin_pw

The Replication Agent administrator login password.

Default
"" (empty string)

Note: The default value **""** (empty string) is deprecated by Replication Server® Option for Oracle 15.7.1 ESD #2. You must explicitly enter a password for the user name authorized to log in to Replication Agent. The default Replication Agent password policy requires a password to be at least six characters but not more than 255 characters.

Value
A password that complies with the password security requirements.

Comments

- The value of the **ltm_admin_pw** parameter is the password for the user name authorized to log in to Replication Agent.
- The value of the **ltm_admin_pw** parameter is encrypted in the Replication Agent configuration file.
- To change the value of the **ltm_admin_pw** parameter, use **ra_set_login**.
- When you change the value of the **ltm_admin_pw** parameter with **ra_set_login**, the new value is recorded in the RASD immediately. However, you must shut down and restart the Replication Agent instance to make the new password take effect.

After you change the value of the **ltm_admin_pw** parameter with **ra_set_login**, you must use the new password the next time you log in to Replication Agent.

See *Table 3. Password Parameters* on page 70 for a list and description of password parameters with their default values.

See also

- *ra_config* on page 70

ltm_admin_pw_min_length

The minimum length of the Replication Agent administrator login password.

Note: This parameter has been deprecated. Use the **min_password_len** parameter instead.

Default
-1 (disabled)

Value
An integer from 3 to 12.

Comments

- When users attempt to change the Replication Agent administrator login password, the new password is validated against **ltm_admin_pw_min_len** to ensure the password has no fewer than three and no more than twelve characters.

See also

- *ra_config* on page 70

ltm_admin_user

The Replication Agent administrator login name.

Default
sa

Value
A valid user name on the Replication Agent host machine.

Comments

- The value of the **ltm_admin_user** parameter is the user name authorized to log in to Replication Agent.
- To change the value of the **ltm_admin_user** parameter, use the **ra_set_login** command.
- If you change the value of the **ltm_admin_user** parameter with the **ra_set_login** command, the new value is recorded in the RASD immediately. However, you must shut

down and restart the Replication Agent instance to make the new administrator name take effect.

- After you change the value of the **ltm_admin_user** parameter with **ra_set_login**, you must use the new administrator name the next time you log in to Replication Agent.

max_ops_per_scan

The **max_ops_per_scan** parameter is deprecated and only provided for backward compatibility. Changes made to the **max_ops_per_scan** parameter will not affect Replication Agent behavior.

Default
1000

Values
An integer from 25 to 2^{31} - 1.

pdb_archive_path

Identifies the directory path where Replication Agent expects to find archived redo log files. When archived redo log files or archived transaction log files are stored in the file system, the configuration parameter is set to a file system path.

Default
<not configured>

Values
A valid directory path on the machine that points to a location where Oracle places the archived redo log files. For example,

```
ORACLE_HOME\oradata\orcl\archive
```

Archive logs stored in and managed by ASM are owned by the corresponding unique Oracle database name. If the Oracle database name differs from the global unique database name, you must set **pdb_archive_path** to both the name of the ASM disk group and the globally unique name of the database in which the archive logs are stored:

```
pdb_archive_path=+DISK_GROUP1/database_name
```

You can also set **pdb_truncate_xlog** to manually remove archive logs. Set the **pdb_archive_path** to the ASM disk group name, and precede the archive logs to be manually removed with a plus "+" sign.

Comments

- You must set **pdb_archive_path** when configuration parameter **pdb_include_archives** is set to true, and you must set it to a valid location before Replication Agent can be placed in a Replicating state.

- If Replication Agent cannot find an expected log record in the Oracle online redo logs, Replication Agent will search this directory for the archived log file containing the required record.

See also

- *pdb_archive_remove* on page 168
- *pdb_include_archives* on page 176

pdb_archive_remove

Enables or disables the removal of Oracle archived redo log files from the path specified by **pdb_archive_path**.

Default
false

Values
true – Allows the removal of archived Oracle redo log from the path specified by **pdb_archive_path**. Removal occurs based on the execution of command **pdb_truncate_xlog**, or the timing of automatic truncation based on parameters **truncation_type** and **truncation_interval**.

false – Disables the removal of archived Oracle redo log files.

Comments

- Set this configuration to **true** when the path specified by **pdb_archive_path** is established solely for Replication support, and automatic removal of unneeded archived log files is desired.
- If the path specified by **pdb_archive_path** is shared by other processes, or the removal of archived log files is expected to be performed by processes other than Replication Agent, this parameter should be **false**.
- Configuration parameters **truncation_type** and **truncation_interval**, and command **pdb_truncate_xlog** have no impact when this configuration parameter is set to **false**.

See also

- *pdb_archive_path* on page 167
- *truncation_interval* on page 211
- *truncation_type* on page 211
- *pdb_truncate_xlog* on page 60

pdb_auto_create_repdefs

Configures Replication Agent to automatically create replication definitions at Replication Server when a table or procedure is marked for replication, after initialization. To improve performance, **pdb_auto_create_repdefs** is ignored during initialization.

Note: The **pdb_xlog create** command no longer checks the setting of **pdb_auto_create_repdefs** during initialization. To create replication definitions for all marked tables after executing **pdb_xlog create**, execute command **rs_create_repdef all**.

Default
false

Values
true – Replication Agent automatically creates replication definitions at Replication Server when tables or procedures are marked after initialization.

false – no replication definitions are created when tables or procedures are marked.

Comments

Note: Replication Agent always assumes that a database replication definition exists for the primary database.

- The **rs_username** user must have **create object** permission before Replication Agent can use it to create replication definitions from Replication Server. You must grant this permission manually from the RSSD.
- The table and procedure replication definitions that Replication Agent creates assume that a database level replication definition for the primary database already exists at Replication Server. All replication definitions created using **pdb_auto_create_repdefs** include the **send standby** clause, which means the replication definition will only be used by Replication Server if there is a database level replication definition or the primary Replication Server connection is for a warm standby configuration. The replication definition created by **rs_create_repdef** can not be individually subscribed to. If you do not wish to have a database level replication definition or warm standby configuration, you must use a different tool or create replication definitions manually, and not use **rs_create_repdef**.
- Replication definitions created by **rs_create_repdef** will always define the datatypes using available user defined datatypes (UDDs) that are installed in Replication Server. This means that customers using **rs_create_repdef** should not set Replication Agent configuration parameter **pdb_convert_datetime** to true, as doing so converts date and timestamp datatypes to Sybase format, instead of UDD format.
- If this parameter is set to true and when **pdb_setrepproc** is invoked to mark a procedure or procedures, a replication definition is created at Replication Server for each procedure that gets marked for replication.

- If this parameter is set to true and when **pdb_setreptable** is invoked to mark a table or tables, a replication definition is created at Replication Server for each table that gets marked for replication.
- If this parameter is set to true and when **pdb_setreptable** is invoked to unmark a table or tables, the replication definition is dropped at Replication Server for each table that gets unmarked for replication.
- If this parameter is set to true and when **pdb_setrepproc** is invoked to unmark a procedure or procedures, a replication definition is dropped at Replication Server for each procedure that gets unmarked for replication.
- The following applies to replication definition table and procedure names:
  - All non-alphanumeric characters and spaces are removed and are not part of the table or procedure name.
  - Underscores are kept as part of the name even though they are non-alphanumeric characters.
  - Periods are replaced with underscores.
- Replication definition names for tables always begin with the prefix "*ra$,*" followed by a unique alphanumeric identifier (maximum of 8 characters), and ending with a table or object name. For example, for a replicate name of "My Table," the resulting replication definition name is "ra$0x7952_mytable." For an especially long replicate name of "mytable890123456789012345678 90" (30 characters), the resulting replication definition name is "ra$0x7952_mytable8901234567890" (30 or 255 characters maximum, depending on whether or not the **pdb_support_large_identifier** configuration parameter is set).
- Replication definition names for procedures are the same name as the procedure.

pdb_automark_tables

Determines if Replication Agent automatically marks tables for replication during DDL replication.

Default
true

Values
true – user tables are automatically marked during DDL replication.

false – user tables are not automatically marked during DDL replication. They must always be marked using the **pdb_setreptable** command (default).

Comments

- The default value for **pdb_automark_tables** is set to **true** when a Replication Agent instance is created. In this default setting, all user tables (those whose owners are not contained in the **owner_filter_list**) are marked for replication when the **pdb_xlog** command is executed with the **init** keyword. In addition, when replication of DDL

commands is enabled (**pdb_setrepddl** setting is enabled by default), any create table command for a user table (those whose owners are not contained in the **owner_filter_list**) is automatically marked for replication. If automatic marking of tables is not desired, this configuration parameter value should be changed to **false**.

- Automatic marking of new tables (those created in the primary database with the **create table** command) will only occur when replication of DDL commands is enabled (**pdb_setrepddl** is set to enable) and the table is a user table (those whose owners are not contained in the **owner_filter_list**) and **pdb_automark_tables** is set to **true**. Modifying the **owner_filter_list** after the initialization may cause inconsistencies.
- Tables are automatically unmarked for replication when a **drop table** command issued at the primary and is recorded in the transaction log, regardless of the settings of **pdb_setrepddl** or **pdb_automark_tables**. This is due to the fact that a dropped table cannot be replicated from.
- Automatic marking of user tables is independent of manual marking of tables using the **pdb_setreptable** command. In other words, you can always mark or unmark individual or all tables for replication using the **pdb_setreptable** command, regardless of the setting of **pdb_automark_tables**.

pdb_auto_run_scripts

Determines whether Replication Agent automatically runs scripts (for transaction log creation and removal, and object marking and unmarking) at the primary database.

Default
true

Values
true – Replication Agent automatically runs scripts.

false – Replication Agent generates and saves the scripts, but it does not automatically run them at the primary database.

Comments

- When the **pdb_xlog** command is invoked to create or remove the transaction log, Replication Agent generates a script to create or remove the Replication Agent system objects.
- When the **pdb_setrepproc** command is invoked to mark or unmark an object in the primary database, Replication Agent generates a script to create or remove the transaction log objects necessary for object marking.
- Replication Agent always saves the scripts in a file. Log creation and removal scripts are saved in files named `partinit.sql` and `partdeinit.sql`. Object marking and unmarking scripts are saved in files named `partmark.sql` and `unmark.sql`.
- When the **pdb_auto_run_scripts** parameter is set to **false**, the scripts are created but no action is taken. This allows you to review the scripts to see what action will be taken before

execution. You cannot execute the scripts. You must **set pdb_auto_run_scripts** parameter back to t**rue** and re-execute the command to have the desired action take place.

- As described above for the **pdb_xlog** and **pdb_setrepproc commands**, Oracle creates the `partinit`, `partdeinit`, `partmark`, and `partunmark` scripts. These scripts cannot be executed (since they do not update the RASD) and are for informational purposes only.
- This parameter must be set to **true** for initialization to occur.

pdb_convert_datetime

Determines whether Replication Agent converts non-Sybase temporal datatype values to the Sybase `datetime` format.

Note: This parameter has been deprecated. If you use Replication Server version 12.0 or later, use the Replication Server heterogeneous datatype support (HDS) feature for all datatype conversion and translation.

Default
false

Values
true – Replication Agent converts all data in the primary database native date/time datatypes to the Sybase `datetime` format.

false – Replication Agent replicates data in the primary database native date and time datatypes as character strings.

Comments

- The **pdb_convert_datetime** parameter is provided for backward compatibility with previous versions of Replication Agents and Replication Server. If you use Replication Server version 12.0 or later, use the Replication Server heterogeneous datatype support (HDS) feature for all datatype conversion and translation.
- Replication definitions created by the **rs_create_repdef** command always define the datatypes using available user defined datatypes (UDDs) that are installed in Replication Server. If you use the **rs_create_repdef** command, do not set the Replication Agent configuration parameter **pdb_convert_datetime** to true, as doing so converts `date` and `timestamp` datatypes to Sybase format, instead of UDD format.
- Replication Agent checks the value of the **pdb_convert_datetime** parameter at the time an object is marked for replication. Transaction log objects that support replication of the marked object are constructed to provide the desired date format.

  If you change the value of the **pdb_convert_datetime** parameter after an object is marked, it has no effect on the marked object. To change the `datetime` datatype conversion for a marked object, you must unmark the object, change the value of the **pdb_convert_datetime** parameter, then re-mark the object.

- For log-based Replication Agents, the conversion takes place after the log records have been read and before LTL is generated to send to Replication Server.
- Any missing component in the non-Sybase date/time datatype format is treated as an implied 0 (zero) when it is converted to the Sybase `datetime` format.
- When the value of the **pdb_convert_datetime** parameter is true, the replication definition for each table should specify that the declared datatype for all date/time columns is `datetime`.
- If the value of the **pdb_convert_datetime** parameter is false, Replication Agent sends date/time data to the primary Replication Server as character strings. The character string size varies by database and datatype:
  - Oracle: `DATE` = `char(19)`
- Set the value of the **pdb_convert_datetime** parameter to true if all date/time values replicated from the primary database will be replicated as the Sybase `datetime` datatype.
- **pdb_convert_datetime** must be false if a table containing replicated LOB columns has `datetime` datatype in the primary key.
- Replication Agent date/time datatype conversion does not work with LOB column replication, unless either of the following conditions exist (these conditions are not required for Oracle):
  - There are no date/time columns in the tables that have LOB column replication enabled, or
  - The primary keys in tables that have LOB column replication enabled do not contain date/time datatypes.

  Otherwise, if you use the **pdb_setrepcol** command to enable LOB column replication, you must set the value of the **pdb_convert_datetime** parameter to false.
- Set the value of the **pdb_convert_datetime** parameter to false for better Replication Agent throughput performance and optimal datatype handling.
- If **pdb_convert_datetime** is true and there are DB2 Universal Database parameterized `TIMESTAMP` values, the resulting `datetime` value has a precision of 3, regardless of the `TIMESTAMP` precision parameter. For example, if `TIMESTAMP(0)` is used for the value `2012-05-20 10:15:45`, the `datetime` value is `2012-05-20 10:15:45:000`. If `TIMESTAMP(12)` is used for the value `2012-05-20 10:15:45:123456789012`, the `datetime` value is `2012-05-20 10:15:45:123`.

pdb_dflt_column_repl

Determines whether LOB column replication is enabled by default when tables are marked.

Default
false

Values
true – LOB column replication is enabled by default (automatically) when tables are marked.

false – LOB column replication is disabled by default when tables are marked.

Comments

- If the value of the **pdb_dflt_column_repl** parameter is false when a table is marked for replication, no transactions that affect LOB columns in the table can be replicated until replication is explicitly enabled with the **pdb_setrepcol** command.
- You can use the **pdb_setrepcol** command to enable or disable replication for all LOB columns in all marked tables at once.
- When replication is disabled for a LOB column, any part of an operation that affects that column will not be recorded in the transaction log, even if the operation also affects other columns for which replication is enabled.

pdb_dflt_object_repl

Determines whether replication is enabled by default when objects (tables or stored procedures) are marked.

Default
true

Values
true – enables replication by default (automatically) when objects are marked.

false – disables replication by default when objects are marked.

Comments

- If the value of the **pdb_dflt_object_repl** parameter is false when a table is marked for replication, no transactions can be replicated from that table until replication is explicitly enabled with the **pdb_setreptable** command.
- If the value of the **pdb_dflt_object_repl** parameter is false when a stored procedure is marked for replication, no invocations of that stored procedure can be replicated until replication is explicitly enabled with the **pdb_setrepproc** command.
- You can use the **pdb_setrepproc** or **pdb_setreptable** command to enable or disable replication for all marked stored procedures or tables at once.
- When replication is disabled for a table, no operations that affect that table will be recorded in the transaction log.
- When replication is disabled for a stored procedure, no invocations of that stored procedure are recorded in the transaction log.

pdb_ignore_unsupported_anydata

Determines whether Replication Agent ignores data of unsupported datatypes stored in columns of type ANYDATA.

Default
false

Values
true – Replication Agent ignores data of unsupported datatypes stored in columns of type ANYDATA, sending no data for these columns to Replication Server.

false – Replication Agent sends the string `type not supported` to Replication Server for data of unsupported datatypes stored in columns of type ANYDATA. This causes Replication Server failure, after which corrections must be made at Replication Server or the replicate database for each table row containing unsupported data.

Comments

- Replication Agent does not replicate data of these Oracle datatypes stored in a column of type ANYDATA:
  - BFILE
  - NESTED TABLE
  - REF
  - UROWID
  - VARRAY
- Replication Agent checks the setting of **pdb_ignore_unsupported_anydata** only when an object is marked for replication. To reset **pdb_ignore_unsupported_anydata** and change Replication Agent behavior for a marked object, you must unmark the object before you reset **pdb_ignore_unsupported_anydata**. The change to **pdb_ignore_unsupported_anydata** takes effect once you re-mark the object.
- If **pdb_ignore_unsupported_anydata** is set to true and the replicate table has a default column value for the corresponding ANYDATA columns, the primary and replicate tables will be inconsistent. If the replicate table has no default column value for the corresponding ANYDATA columns, Replication Server fails, even though **pdb_ignore_unsupported_anydata** is set to true.
- To recover from a Replication Server failure caused by data of unsupported datatypes found in an ANYDATA column, do one of the following:
  - Alter the corresponding replicate table so that the table has a default value for columns of type ANYDATA. You can do this only if **pdb_ignore_unsupported_anydata** is set to true.

- Create a trigger in the corresponding replicate table to provide a default value for columns of type `ANYDATA`. You can do this only if **pdb_ignore_unsupported_anydata** is set to true.
- Customize a Replication Server function string to provide a default value for columns of type `ANYDATA` in the replicate table.

pdb_include_archives

Enables or disables the use of Oracle archive log files.

Default
true

Values
true – allows reading of the archived Oracle redo log files from the path specified by **pdb_archive_path**. The configuration of Oracle automatic archiving is supported under this mode. Use **pdb_archive_remove** to remove old archives logs that are no longer needed to support replication.

false – only online redo logs files are read. Oracle automatic archiving must be disabled. Replication Agent executes Oracle archive commands to archive the redo logs once they are no longer needed for replication.

Comments

- Set the configuration to true when use of archive logs is preferred or when Oracle must be configured to perform automatic archiving. Set this value to false if accessing only the on-line redo logs is preferred.
- Set this value to false if using only the online redo logs is preferred.

See also

- *pdb_archive_path* on page 167
- *truncation_interval* on page 211
- *truncation_type* on page 211

pdb_skip_missing_user

Determines whether or not Replication Agent skips the processing of any command for which there is no matching session or user information.

Default
false

Values
true – Replication Agent skips the processing of any command for which there is no matching session or user information. A message is logged identifying the skipped operation.

false – Replication Agent continues to process any command for which there is no matching session or user information. The default session user is sent as "missing."

pdb_support_large_identifier

Supports replication of large identifiers up to 255 characters in length with Replication Server 12.6 and later.

Default
false

Value
true – objects containing large identifiers may be marked for replication.

false – objects containing large identifiers may *not* be marked for replication.

Comments

- If **pdb_support_large_identifier** value is **false**, when an object (Table/Procedure/Function) is being marked for replication, the object is checked for any identifiers that are longer than 30 characters. An error is returned and the object is not marked for replication if the object has identifiers longer than 30 characters.
- This parameter may be set to **true** if the Replication Server being used is at version 12.6 or later and the replicate database can support large identifiers.
- When **pdb_support_large_identifier** is set to **true**, objects being marked for replication are not checked for identifiers longer than 30 characters.

pdb_timezone_file

Specifies the file to read at Replication Agent initialization to obtain Oracle time zone information.

Default
<not configured>

Value
A valid path to the Oracle time zone file including the `timezone` file name.

Comments

- If the value is not specified, it will default to the Oracle installation `oracore/zoneinfo/timezone.dat` file. For example,

```
$ORACLE_HOME/oracore/zoneinfo/timezone.dat
```

- The `timezone` file specified must be for the same release and platform as the primary Oracle database. For example, an Oracle 9i timezone file is not compatible with an Oracle 10g primary database, and a Windows `timezone` file is not compatible with a UNIX version of Oracle.

pdb_xlog_device

The primary database device on which Replication Agent transaction log objects are created.

Note: This parameter has been deprecated. Use the **ra_admin_device** parameter instead.

Default
NULL

Value
A valid primary database device name or NULL.

Comments

- The value of the **pdb_xlog_device** parameter is the device specification of the primary database device to be used in SQL scripts generated by Replication Agent to create transaction log objects.
- The **pdb_xlog_device** parameter allows you to specify a single device on which all Replication Agent transaction log objects will be created, even if the database uses multiple devices.
- If the value of the **pdb_xlog_device** parameter is **NULL**, no device is specified in the SQL **create** statements, and Replication Agent transaction log objects are placed on the primary data server system-defined default device.

See also

- *ra_admin_device* on page 186

pdb_xlog_prefix

The prefix string used in database object names to identify Replication Agent transaction log objects.

Note: This parameter has been deprecated. Use the **ra_admin_instance_prefix** parameter instead.

Default
ra_

Value
A character string of 1 to 3 characters.

Comments

- When Replication Agent generates database object names for transaction log components in the primary database, it uses the value of the **pdb_xlog_prefix** parameter as an object name prefix.
- Replication Agent uses the value of the **pdb_xlog_prefix** parameter to recognize its transaction log objects in the primary database. Therefore, if you change the value of the **pdb_xlog_prefix** parameter after the transaction log objects are created, Replication Agent will not be able to find its transaction log objects.
- The value of the **pdb_xlog_prefix** parameter is case-insensitive and any lowercase character is stored as an uppercase character.
- The value of the **pdb_xlog_prefix_chars** parameter specifies the non-alphabetic characters that can be used in the prefix string.

See also

- *ra_admin_instance_prefix* on page 186

pdb_xlog_prefix_chars

Non-alphabetic characters that are allowed in the database object name prefix string that identifies Replication Agent transaction log objects.

Note: This parameter has been deprecated. Use the **ra_admin_prefix_chars** parameter instead.

Default
_#@&1234567890

Value
A string of characters with no separators.

Comments

- The default value of the **pdb_xlog_prefix_chars** parameter depends on the type of primary database that the Replication Agent instance was created for. The default value is based on the standard, non-alphabetic characters allowed by each non-Sybase database.
- When you set or change the value of the **pdb_xlog_prefix_chars** parameter, the new value replaces any existing value; it does not add or append the new value to a previous value.
- When you use the **ra_config** command to set the value of the **pdb_xlog_prefix** parameter, any non-alphabetic characters specified on the command line are validated against the value of the **pdb_xlog_prefix_chars** parameter.
- Alphabetic characters a-z are always valid in the **pdb_xlog_prefix** parameter, and they need not be specified.

- Replication Agent does not support delimited names for transaction log objects, so you cannot use a space character in the value of the **pdb_xlog_prefix** parameter.
- The value you specify for the **pdb_xlog_prefix_chars** parameter is not validated. There are no restrictions on the characters you can include.

Note: The primary data server may restrict the characters used in certain positions in a database object name. Refer to the documentation for your primary data server for more information.

See also

- *ra_admin_prefix_chars* on page 188

pds_connection_type

The type of connectivity driver used on the primary database connection.

Default

One of the following values is set automatically when the Replication Agent instance is created.

Values

ORAJDBC – Replication Agent uses the Oracle JDBC™ **UDBJDBC** driver to connect to the primary Oracle database.

Comments

- The value of the **pds_connection_type** parameter is set automatically at the time a Replication Agent instance is created. The specific value depends on the type of Replication Agent instance created.

Note: Do not change the default value of the **pds_connection_type** parameter.

- The value of the **pds_connection_type** parameter determines which of several other Replication Agent configuration parameters related to the primary database connection must also have values specified.
  - **ORAJDBC** requires corresponding values for the following parameters:
    - **pds_host_name**
    - **pds_port_number**
    - **pds_database_name**
  - For the `tnsnames.ora` file, the following parameters are required:
    - **pds_tns_filename**
    - **pds_tns_connection**
  - The value of the **pds_connection_type** parameter is automatically set when a Replication Agent instance is created.

pds_database_name

The name of the primary database.

Default
<not_configured>

Value
A valid database name.

Note: For Oracle, if **pds_tns_connection** is set, you cannot set **pds_database_name** .

Comments

- The value of the **pds_database_name** parameter is the name of the primary database on the primary data server.

  Note: Some primary data servers may not support multiple databases in a single instance of the data server. In that case, the value of the **pds_database_name** parameter should be the name of the data server instance.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

pds_host_name

The name of the primary data server host machine.

Default
<not_configured>

Value
A valid host name.

Note: You cannot set **pds_host_name** if the **pds_tns_connection** is set.

Comments

- The value of the **pds_host_name** parameter is the network name of the host machine on which the primary data server resides.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

pds_password

The password that Replication Agent uses for primary data server access.

Default
"" (empty string)

Value
A valid password.

Comments

- The value of the **pds_password** parameter is the password for the user login name that Replication Agent uses to access the primary data server.
- The value of the **pds_password** parameter is encrypted in the Replication Agent instance configuration file.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

pds_port_number

The client port number for the primary data server.

Default
1111

Value
A valid port number on the primary data server host machine.

Note: You cannot set **pds_port_number** if **pds_tns_connection** is set.

Comments

- The value of the **pds_port_number** parameter is the client port number for the primary data server.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

pds_retry_count

The number of times Replication Agent tries to establish a connection to the primary database.

Default
5

Value
An integer from 0 to 2,147,483,647.

Comments

- The value of the **pds_retry_count** parameter is the number of times that Replication Agent tries to establish a network connection to the primary database after a connection failure.
- Sybase recommends a setting of 5 for this parameter.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

pds_retry_timeout

The number of seconds Replication Agent waits between retry attempts to connect to the primary database.

Default
10

Value
An integer from 0 to 3600.

Comments

- The value of the **pds_retry_timeout** parameter is the number of seconds that Replication Agent waits between retry attempts to establish a network connection to the primary database after a connection failure.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

pds_ssl_sc_dn

pds_ssl_sc_dn parameter is the distinguished name (DN) of the primary data server certificate.

Default
<not configured>

Value
A valid server certificate DN.

Comments

- This parameter is only valid if **pds_use_ssl** is set.
- If this parameter is set, the DN field in the server certificate is verified to match this parameter. If it does not match, the connection to the primary data server fails.

pds_tns_connection

The Oracle connection name that identifies the primary database connection in the Oracle `tnsnames.ora` file.

Default
Not configured.

Value
A valid primary database connection name from the Oracle `tnsnames.ora` file specified by the **pds_tns_filename** configuration parameter.

Comments

- Setting of the configuration parameter overrides settings of the configuration parameters **pds_host_name**, **pds_database_name**, and **pds_port_number**.
- This configuration parameter is required when the Oracle data server instance to be replicated is part of a Real Application Cluster (RAC) configuration.

See also

- *pds_tns_filename* on page 184

pds_tns_filename

The fully-qualified file name identifying the Oracle `tnsnames.ora` file that contains connection parameters for the primary Oracle data server.

Default
Not configured.

Value
A valid Oracle `tnsnames.ora` file that contains the connection parameters to the primary Oracle data server. This file normally resides in the `ORACLE_HOME\network\admin` directory.

Comments

- When configured, Replication Agent will use the connection information specified in the `tnsnames.ora` file to connect to the primary database and the **pds_host_name** and the **pds_port_number** are ignored. The **pds_tns_connection** name should be configured to the entry name in the Sybase interfaces file when **pds_tns_filename** is configured.
- Setting of the configuration parameter is required when the Oracle data server instance to be replicated is part of a Real Application Clusters (RAC) configuration.

Warning! The Replication Agent process must have read permission to this file. Access failures will prevent Replication Agent from connecting to the Oracle server.

See also

- *pds_tns_connection* on page 184

pds_use_ssl

pds_use_ssl indicates whether to use SSL to connect to the primary data server.

Default
False

Value
True | False

Comments

- If this parameter is set to true, the parameter **pds_port_number** must be set to Oracle SSL port, 2484. You can also choose a different SSL port number.

pds_username

The user login name that Replication Agent uses for primary data server access.

Default
<not_configured>

Value
A valid user name.

Comments

- The value of the **pds_username** parameter is the login name that Replication Agent uses to log in to the primary data server.

  This login name must be defined in the primary data server, with appropriate privileges or authority in the primary database.
- Replication Agent uses this login to access primary database objects and to create, remove, and manage its transaction log objects in the primary database.
- **rs_ticket** requires that the user name specified by **pds_username** be different from the user ID specified in the connection to Replication Server (the maintenance user). You can get the name of the maintenance user by executing **ra_maintid**.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

ra_admin_device

The primary database device on which Replication Agent system objects are created.

Default
NULL

Value
A valid primary database device name or NULL.

Comments

- The value of the **ra_admin_device** parameter is the device specification of the primary database device to be used in SQL scripts generated by Replication Agent to create system objects.
- The **ra_admin_device** parameter allows you to specify a single device on which all Replication Agent system objects will be created, even if the database uses multiple devices.
- If the value of the **ra_admin_device** parameter is NULL, no device is specified in the SQL **create** statements, and Replication Agent system objects are placed on the primary data server system-defined default device.

See also

- *pdb_xlog_device* on page 178

ra_admin_instance_prefix

The prefix string used to identify Replication Agent system objects specific to one Replication Agent instance.

Default
ra_

Value
A character string of 1 to 3 characters.

Comments

- When Replication Agent generates system objects in the primary database, it uses the value of the **ra_admin_instance_prefix** parameter as an object name prefix.
- Replication Agent uses the value of the **ra_admin_instance_prefix** parameter to recognize its system objects in the primary database. Therefore, if you change the value of the **ra_admin_instance_prefix** parameter after the system objects are created, Replication Agent will not be able to find its objects.

- The value of the **ra_admin_instance_prefix** parameter is case-insensitive and any lowercase character is stored as an uppercase character.
- The value of the **ra_admin_instance_prefix** parameter specifies the non-alphabetic characters that can be used in the prefix string.
- Each Replication Agent instance in a Replication Agent group must be configured with its own unique value for **ra_admin_instance_prefix**.

See also

- *pdb_xlog_prefix* on page 178
- *ra_admin_prefix* on page 187

ra_admin_prefix

The prefix string used to identify shared Replication Agent system objects.

Default
ra_

Value
A character string of 1 to 3 characters.

Comments

- When Replication Agent generates shared database object names for transaction log components in the primary database, it uses the value of the **ra_admin_prefix** parameter as an object name prefix.
- The value of the **ra_admin_prefix** parameter specifies the non-alphabetic characters that can be used in the prefix string.
- Replication Agent uses the value of the **ra_admin_prefix** parameter to recognize its system objects in the primary database. Therefore, if you change the value of the **ra_admin_prefix** parameter after a Replication Agent instance has been created, Replication Agent will not be able to find its objects.
- Each Replication Agent instance in a Replication Agent group must be configured with the same value for **ra_admin_prefix**.

See also

- *ra_admin_instance_prefix* on page 186
- *ra_admin_owner* on page 188

ra_admin_prefix_chars

Non-alphabetic characters that are allowed in the database object name prefix string that identifies Replication Agent system objects.

Default
_#@&1234567890

Value
A string of characters with no separators.

Comments

- The default value of the **ra_admin_prefix_chars** parameter depends on the type of primary database that the Replication Agent instance was created for. The default value is based on the standard, non-alphabetic characters allowed by each non-Sybase database.
- When you set or change the value of the **ra_admin_prefix_chars** parameter, the new value replaces any existing value; it does not add or append the new value to a previous value.
- When you use the **ra_config** command to set the value of the **ra_admin_instance_prefix** parameter, any non-alphabetic characters specified on the command line are validated against the value of the **ra_admin_prefix_chars** parameter.
- Alphabetic characters a-z are always valid in the **ra_admin_instance_prefix** parameter, and they need not be specified.
- Replication Agent does not support delimited names for system objects, so you cannot use a space character in the value of the **ra_admin_instance_prefix** parameter.
- The value you specify for the **ra_admin_prefix_chars** parameter is not validated. There are no restrictions on the characters you can include.

Note: The primary data server may restrict the characters used in certain positions in a database object name. Refer to the documentation for your primary data server for more information.

See also

- *pdb_xlog_prefix_chars* on page 179

ra_admin_owner

The owner of all Replication Agent system objects, including shared and instance-specific system objects.

Default
<not_configured>

Value
A valid user name.

Comments

- When **ra_admin_owner** is not configured, it uses the value of the **pds_username** parameter.
- The user name specified by **ra_admin_owner** must be defined in the primary data server.
- You cannot change the value of the **ra_admin_owner** parameter after a Replication Agent instance has been created.
- Each Replication Agent instance in a Replication Agent group must be configured with the same value for **ra_admin_owner**.

See also

- *ra_admin_instance_prefix* on page 186
- *ra_admin_prefix* on page 187

ra_retry_count

The number of times Replication Agent attempts to restart replication after a failure.

Default
2

Value
An integer greater than 0.

Comments

- The value of the **ra_retry_count** parameter is the number of times that the Log Transfer Manager component will try to get the Replication Agent instance back into Replicating state after a failure or error causes the instance to go to Replication Down state.
- When a network connection fails, Replication Agent attempts to re-establish the connection, using the values stored in its connection configuration parameters for that connection.
- If Replication Agent is unable to re-establish a connection after the number of retries specified in the **pds_retry_count** or **rs_retry_count** parameter, then the Replication Agent instance goes to Replication Down state and the Log Transfer Manager component attempts to return the Replication Agent instance to Replicating state, based on the settings of the **ra_retry_count** and **ra_retry_timeout** parameters.

ra_retry_timeout

The number of seconds Replication Agent waits between attempts to restart replication after a failure.

Default
10

Value
An integer greater than 0.

Comment
The value of the **ra_retry_timeout** parameter is the number of seconds that the Log Transfer Manager component will wait between its attempts to get the Replication Agent instance back into Replicating state after a failure causes the instance to go to Replication Down state.

rasd_backup_dir

The directory path for Replication Agent System Database (RASD) backup files.

Default
The path to the RASD `backup` directory created automatically when the Replication Agent instance was created. For example:

- On Microsoft Windows platforms:
  ```
  %SYBASE%\RAX-15_5\inst_name\repository\backup
  ```
  where:
  - *%SYBASE%* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.
- On UNIX platforms:
  ```
  $SYBASE/RAX-15_5/inst_name/repository/backup
  ```
  where:
  - *$SYBASE* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.

Value
A valid path on the Replication Agent host machine.

Comments

- When you create a Replication Agent instance, an RASD `backup` directory is created automatically as part of the instance directory structure. The default value of the **rasd_backup_dir** parameter points to that directory.
- If you specify any valid path as the value of the **rasd_backup_dir** parameter, Replication Agent places its RASD backup files in that directory during RASD backup operations, and it looks in that directory for the RASD backup files during restore operations.

rasd_database

The directory path for the Replication Agent System Database (RASD) database file.

Default
The path to the RASD database file created automatically when the Replication Agent instance was created. For example:

- On Microsoft Windows platforms:
  ```
  %SYBASE%\RAX-15_5\inst_name\repository\inst_name.db
  ```
  where:
  - *%SYBASE%* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.
- On UNIX platforms:
  ```
  $SYBASE/RAX-15_5/inst_name/repository/inst_name.db
  ```
  where:
  - *$SYBASE* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.

Value
A valid path and RASD database file name on the Replication Agent host machine.

Comments

- When you create a Replication Agent instance, the `repository` directory and the RASD database file are created automatically. The default value of the **rasd_database** parameter points to the RASD database file in that directory.
- If you specify any valid path and RASD database file name as the value of the **rasd_database** parameter, the Replication Agent instance looks in that directory for its RASD database file, with the file name you specified.

rasd_mirror_tran_log

Enables or disables Replication Agent System Database (RASD) transaction log mirroring.

Default
false

Values
true – enables mirroring the RASD transaction log to another file.

false – disables mirroring of the RASD transaction log.

Comment
Setting the value of the **rasd_mirror_tran_log** parameter to **true** provides additional recovery options in the event of a device failure on the Replication Agent host machine.

rasd_trace_log_dir

The directory path for the Replication Agent System Database (RASD) trace log file.

Default

The path to the `repository` directory created automatically when the Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

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

  where:
  - *%SYBASE%* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.
- On UNIX platforms:

  ```
  $SYBASE/RAX-15_5/inst_name/repository
  ```

  where:
  - *$SYBASE* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.

Value

A valid path on the Replication Agent host machine.

Comments

- When you create a Replication Agent instance, the `repository` directory is created automatically as part of the instance directory structure. The default value of the **rasd_trace_log_dir** parameter points to that directory.
- If you specify any valid path as the value of the **rasd_trace_log_dir** parameter, the Replication Agent instance writes its RASD trace log file in that directory.

rasd_tran_log

The directory path for the Replication Agent System Database (RASD) transaction log file.

Default

The path to the RASD transaction log file created automatically when the Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

  ```
  %SYBASE%\RAX-15_5\inst_name\repository\inst_name.log
  ```

  where:

- *%SYBASE%* is the path to the Replication Agent installation directory.
- *inst_name* is the name of the Replication Agent instance.

- On UNIX platforms:

```
$SYBASE/RAX-15_5/inst_name/repository/inst_name.log
```

  where:
  - *$SYBASE* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.

Value

A valid path on the Replication Agent host machine.

Comments

- When you create a Replication Agent instance, the `repository` directory and RASD transaction log file are created automatically. The default value of the **rasd_tran_log** parameter points to that transaction log file.
- If you specify any valid path and RASD transaction log file name as the value of the **rasd_tran_log** parameter, the Replication Agent instance looks in that directory for its RASD transaction log file, with the name you specified.

rasd_tran_log_mirror

The directory path for the Replication Agent System Database (RASD) transaction log file mirror.

Default

The path to the RASD transaction log file mirror in the `tran_log_mirror` directory created automatically when the Replication Agent instance was created. For example:

- On Microsoft Windows platforms:

```
%SYBASE%\RAX-15_5\inst_name\repository\tran_log_mirror
\inst_name.log
```

  where:
  - *%SYBASE%* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.
- On UNIX platforms:

```
$SYBASE/RAX-15_5/inst_name/repository/tran_log_mirror/
inst_name.log
```

  where:
  - *$SYBASE* is the path to the Replication Agent installation directory.
  - *inst_name* is the name of the Replication Agent instance.

Value
A valid path on the Replication Agent host machine.

Comment
If you specify any valid path and transaction log file name as the value of the **rasd_tran_log_mirror** parameter, the Replication Agent instance looks in that directory for its RASD transaction log file mirror, with the name you specified.

ra_standby

Determines whether or not Replication Agent functions in standby mode.

Default
false

Values
true – Replication Agent functions in standby mode.

false – Replication Agent functions in normal mode.

Comments

- In standby mode, Replication Agent:
  - scans the transaction log and keeps the Replication Agent System Database (RASD) current.
  - does not send any Log Transfer Language (LTL) to Replication Server.
  - continues to perform log truncation.
- To function in standby mode, Replication Agent should:
  - have the **rs_source_ds** and **rs_source_db** parameters configured as physical connections to Replication Server.
  - enable or disable the replication of DDL statements as desired using the **pdb_setrep_ddl** command.
  - set the **pdb_auto_create_repdefs**, **pdb_dflt_column_repl**, **pdb_dflt_object_repl**, and **pdb_automark_tables** parameters to true.

See also

- *rs_source_ds* on page 201
- *rs_source_db* on page 201
- *pdb_auto_create_repdefs* on page 169
- *pdb_dflt_column_repl* on page 173
- *pdb_dflt_object_repl* on page 174
- *pdb_automark_tables* on page 170

ra_statrack_interval

Determines the interval, in seconds, at which statistics are sampled by **ra_statrack**.

Default
60

Values
An integer from 5 - 86400.

See also

- *ra_statistics* on page 104
- *ra_statrack* on page 111
- *ra_statrack_list* on page 111

rman_enabled

Determines whether or not Replication Agent truncates old archive log files using the Oracle **RMAN** utility.

Default
false

Values
true – enables truncation of archive log files using the Oracle **RMAN** utility.

false – disables truncation of archive log files using the Oracle **RMAN** utility.

Comments

- If both the **rman_enabled** and **pdb_archive_remove** parameters are set to true, Replication Agent uses the Oracle **RMAN** utility to remove old archive redo log files from the path specified by the **pdb_archive_path** parameter. If the **rman_enabled** parameter is set to false but the **pdb_archive_remove** parameter is set to true, Replication Agent uses file system operations to remove old archive redo log files from the path specified by the **pdb_archive_path** parameter.
- When the **rman_enabled** parameter is set to true, the ORACLE_HOME environment variable must be set in the runtime context of the Replication Agent process, and the **rman_username** and **rman_password** parameters must be properly configured.
- The Oracle **RMAN** utility must be installed on the same machine as Replication Agent and must be compatible with the Oracle database that contains the archive log files being truncated.
- A change to the **rman_enabled** parameter will take effect only after Replication Agent is suspended and resumed.

See also

- *pdb_archive_remove* on page 168
- *rman_password* on page 196
- *rman_username* on page 196

rman_password

Contains the password used with **rman_username** to connect to the Oracle **RMAN** utility.

Default
"" (empty string)

Value
A valid password.

Comments

- Setting the **rman_password** configuration parameter is required only if **rman_enabled** is set to true.
- The value of the **rman_password** configuration parameter is encrypted in the configuration file for the Replication Agent instance.

See also

- *rman_enabled* on page 195
- *rman_username* on page 196

rman_username

Contains the login name used with **rman_password** to connect to the Oracle **RMAN** utility.

Default
Not configured.

Value
A valid Oracle user name with **sysdba** privileges.

Comments

- Setting the **rman_username** configuration parameter is required only if **rman_enabled** is set to true.
- The value of **rman_username** is the login name that Replication Agent uses to connect to the Oracle **RMAN** utility and manage archive log files. This login name must be defined in the Oracle primary data server and have **sysdba** privileges. If a login name lacking **sysdba** privileges is set by **ra_config**, Replication Agent returns an error.

See also

- *rman_enabled* on page 195
- *rman_password* on page 196

rs_charset

The character set used in communication with the primary Replication Server.

The Replication Agent default character set must be set to match the primary database's character set. The value of the **rs_charset** parameter must be set to match the Replication Server character set. If they differ, Replication Agent will do character set conversion before sending data to Replication Server.

Note: If Replication Agent can connect to Replication Server 15.0.1 or later, the **rs_charset** in Replication Agent is ignored and the **RS_charset** in Replication Server is used.

If the character set on your Replication Agent is different from the one on your primary database, you need to set the RA_JAVA_DFLT_CHARSET environment variable. The Replication Agent character set must be the same as that of the primary database. For more information on setting the RA_JAVA_DFLT_CHARSET environment variable, see Chapter 2 of the *Replication Agent Administration Guide*.

Note: Setting this parameter to anything other than the character set of the primary Replication Server causes Replication Agent to do character set conversion before sending data to Replication Server. This will degrade Replication Agent performance.

Default

Defaults to empty string (**""**).

Value

Any valid Sybase character set supported by the Java VM on the Replication Agent host machine.

Comments

- Use the exact same value as that of the **RS_charset** parameter in the Replication Server configuration (`.cfg`) file which is located at: `$SYBASE/REP-15_0/install/<instance>.cfg`. For example, `iso_1`.
- Configure the primary data server and primary Replication Server to use the same character set.

  Note: If **rs_charset** is not set at the time you try to resume replication, Replication Agent returns an error.

  When the Replication Agent instance is created, the **rs_charset** parameter is set to its default value **""** (empty string).

- If you specify a valid character set for the value of the **rs_charset** parameter, the Replication Agent instance sends replicated transaction data from the primary database to the primary Replication Server in that character set.
- If you do not specify a valid character set name for the value of the **rs_charset** parameter (including the default **rs_charset** value **""**), the Replication Agent instance will not allow you to resume replication.
- If the values of the **rs_charset** and the system default character set are valid but not the same value, Replication Agent converts the replicated transaction data from the system-defined database character set to the Replication Server character set before sending it to the primary Replication Server.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_host_name

The name of the primary Replication Server host machine.

Default
<not_configured>

Value
A valid host name.

Comments

- The value of the **rs_host_name** parameter is the name of the host machine for the primary Replication Server.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_packet_size

The network packet size on the connection to the primary Replication Server.

Default
2048

Value
An integer from 2048 to 65536.

Comments

- The value of the **rs_packet_size** parameter is the maximum size (in bytes) of the network packets handled by the TCP/IP network protocol.
- The Replication Agent **rs_packet_size** parameter is equivalent to the Replication Server **rs_packet_size** parameter.

- When the network packet size is smaller, more packets must be processed to transmit a given amount of data to Replication Server. When the network packet size is larger, more system resources are consumed to process the packets.
- The optimum value of the **rs_packet_size** parameter is based on the nature of the typical data replicated. If the typical operation is very large, a larger packet size is more efficient.
- A larger value of the **rs_packet_size** parameter is more efficient when the value of the **lti_batch_mode** parameter is **true**.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_password

The password that Replication Agent uses for Replication Server access.

Default
"" (empty string)

Value
A valid password.

Comments

- The value of the **rs_password** parameter is the password for the user login name that Replication Agent uses to log in to the primary Replication Server.
- The value of the **rs_password** parameter is encrypted in the Replication Agent instance configuration file.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_port_number

The client port number of the primary Replication Server.

Default
1111

Value
A valid port number on the Replication Server host machine.

Comments

- The value of the **rs_port_number** parameter is the client port number of the primary Replication Server.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_replicate_owner_required

Indicates if the owner is always included with the replicate table clause when generating replication definitions.

Default
true

Value
true – the owner is always included in the replicate table clause.

false – the owner is not included in the replicate table clause unless the table is marked with the owner mode value set to **on**.

Comments
For additional information, see the **rs_create_repdef** command.

rs_retry_count

The number of times Replication Agent retries establishing a connection to the primary Replication Server.

Default
5

Value
An integer greater than 0.

Comments

- The value of the **rs_retry_count** parameter is the number of times that Replication Agent will try to establish a network connection to Replication Server after a connection failure.
- Sybase recommends a setting of 5 for this parameter.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_retry_timeout

The number of seconds Replication Agent waits between attempts to connect to the primary Replication Server.

Default
10

Value
An integer greater than 0.

Comments

- The value of the **rs_retry_timeout** parameter is the number of seconds that Replication Agent waits between its retry attempts to establish a network connection to the primary Replication Server after a connection failure.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_source_db

The name of the database identified in the Replication Server primary database connection.

Default
<not_configured>

Value
A valid database name.

Comments

- The value of the **rs_source_db** parameter is the name of the primary database by which the primary Replication Server recognizes the primary database transaction log.
- The value of the **rs_source_db** parameter must match the name of the database specified in the Replication Server **create connection** command for the primary database.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_source_ds

The name of the data server identified in the Replication Server primary database connection.

Default
<not_configured>

Value
A valid server name.

Comments

- The value of the **rs_source_ds** parameter is the name of the primary data server by which the primary Replication Server recognizes the primary database transaction log.
- The value of the **rs_source_ds** parameter must match the name of the data server specified in the Replication Server **create connection** command for the primary database.
- The value of the **rs_source_ds** parameter should not be the same as the name of the Replication Agent instance.

- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rs_ssl_sc_dn

rs_ssl_sc_dn is the distinguished name (DN) of the Replication Server certificate and is only valid if **rs_use_ssl** is set to true.

Default
<not configured>

Value
A valid Replication Server certificate DN.

Comments

- If **rs_use_ssl** is set to true, the DN field in the Replication Server certificate is verified to match this parameter. If it does not match, the connection to the Replication Server fails.

rs_ticket_version

Determines whether Replication Agent records the primary database time or the primary database date and time into the **rs_ticket marker**.

Default
1

Value

- 1 – Replication Agent records only the primary database time.
- 2 – Replication Agent records both the primary database date and time.

Comments

- If the value is set to **1**, Replication Agent records only the primary database time into **rs_ticket** marker. For example, 13:20:19.368.
- If the value is set to **2**, Replication Agent records both the primary database date and time into **rs_ticket** marker. For example, 12/14/07 13:20:19.368.

See also

- *rs_ticket* on page 129

rs_use_ssl

rs_use_ssl indicates whether Replication Agent (as a client) should use SSL to connect to Replication Server.

Default
False

Value
True | False

Comments

- By default, this parameter is set to false, whether SSL is enabled from Replication Agent (as a client) into Replication Server.

rs_username

The user login name that Replication Agent uses for Replication Server access.

Default
<not_configured>

Value
A valid user name.

Comments

- The value of the **rs_username** parameter is the user login name that Replication Agent uses to log in to the primary Replication Server.
- The value of the **rs_password** parameter is the password for the login name specified by the **rs_username** parameter.
- The user login name that Replication Agent uses to log in to Replication Server must have **connect source** permission in Replication Server.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.
- The **rs_username** user must have **create object** permission before Replication Agent can use it to create replication definitions from Replication Server. You must grant this permission manually from the RSSD.

rssd_charset

The character set used in communication with the RSSD of the primary Replication Server.

Default
"" (empty string)

Value
Any valid Sybase character set supported by the Java VM on the Replication Agent host machine.

Comments

- The value of the **rssd_charset** parameter must match (or be compatible with) the RSSD character set. The RSSD character set is usually the same as the Replication Server default character set identified by the Replication Server **rs_charset** configuration parameter.
- If you specify a valid character set for the value of the **rssd_charset** parameter, the Replication Agent instance communicates with the RSSD using that character set.
- If you do *not* specify a valid character set name for the value of the **rssd_charset** parameter (including the default **rssd_charset** value **""**), Replication Agent communicates with the RSSD using the RSSD charset.
- The **rssd_charset** parameter does not need to be set if the Replication Agent **use_rssd** parameter is set to **false**.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rssd_database_name

The database name of the RSSD of the primary Replication Server.

Default
<not_configured>

Value
A valid database name.

Comments

- The value of the **rssd_database_name** parameter is the database name of the RSSD of the primary Replication Server.
- The **rssd_database_name** parameter does not need to be set if the Replication Agent **use_rssd** parameter is set to **false**.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rssd_host_name

The name of the machine on which the RSSD of the primary Replication Server resides.

Default
<not_configured>

Value
A valid host name.

Comments

- The value of the **rssd_host_name** parameter is the name of the host machine on which the RSSD of the primary Replication Server resides.
- The **rssd_host_name** parameter does not need to be set if the Replication Agent **use_rssd** parameter is set to false.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rssd_password

The password that Replication Agent uses for access to the RSSD of the primary Replication Server.

Default
"" (empty string)

Value
A valid password.

Comments

- The value of the **rssd_password** parameter is the password for the user login name that Replication Agent uses to access the RSSD of the primary Replication Server.
- The value of the **rssd_password** parameter is encrypted in the Replication Agent instance configuration file.
- The **rssd_password** parameter need not be set if the Replication Agent **use_rssd** parameter is set to false.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rssd_port_number

The client port number of the Replication Server System Database (RSSD) of the primary Replication Server.

Default
1111

Value
A valid port number on the RSSD host machine.

Comments

- The value of the **rssd_port_number** parameter is the client port number of the RSSD data server.
- The **rssd_port_number** parameter need not be set if the Replication Agent **use_rssd** parameter is set to false.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

rssd_username

The user login name that Replication Agent uses to access the RSSD of the primary Replication Server.

Default
<not_configured>

Value
A valid user login name in the RSSD data server.

Comments

- The value of the **rssd_username** parameter is the user login name that Replication Agent uses to access the RSSD.
- The **rssd_username** parameter need not be set if the Replication Agent **use_rssd** parameter is set to false.
- See the *Replication Agent Administration Guide* for more information about setting up Replication Agent connection configuration parameters.

scan_fetch_size

The number of rows to fetch from the primary database when Oracle LogMiner is scanning the log in each network round-trip.

Default
1

Value
An integer from 1 to 20.

Comments

- The value of the **scan_fetch_size** parameter is a trade-off between the number of network round-trips and latency:

    - Higher value increases latency because Oracle LogMiner waits for the specified number of log records to be filled after reaching the end of the log file.
    - Lower value reduces latency but requires more network round-trips for Oracle LogMiner to return the scan results to the Replication Agent.
- If the primary database is continuously updated, you can specify a higher **scan_fetch_size** parameter value. Otherwise, specify a lower value.

scan_sleep_increment

The number of seconds to add to each wait interval before scanning the transaction log, after a previous scan yields no transaction to be replicated.

Default
5

Value
An integer from 0 to 3600.

Comments

- The value of the **scan_sleep_increment** parameter is the number of seconds added to each wait interval before the Log Reader component scans the log for a transaction to be replicated, after a previous scan yields no such transaction.
- The number of seconds specified by the **scan_sleep_increment** parameter is added to each wait interval, until the wait interval reaches the value specified by the **scan_sleep_max** parameter.
- For optimum Replication Agent performance, the value of the **scan_sleep_increment** parameter should be balanced with the average number of operations in the primary database over a period of time. In general, better performance results from reading more operations from the transaction log during each Log Reader scan.
- With a primary database that is less frequently updated, increasing the value of the **scan_sleep_increment** parameter may improve overall performance.
- If the database is continuously updated, the value of the **scan_sleep_increment** parameter may not be significant to Replication Agent performance.

scan_sleep_max

The maximum wait interval between Log Reader transaction log scans.

Default
60

Value
An integer from 1 to 3600.

Comments

- The value of the **scan_sleep_max** parameter is the maximum number of seconds that can elapse before the Log Reader component scans the transaction log for a transaction to be replicated, after a previous scan yields no such transaction.
- For reduced replication latency in an infrequently updated database, use lower number settings for the **scan_sleep_max** parameter.
- If the primary database is continuously updated, the value of the **scan_sleep_max** parameter is not significant to Replication Agent performance.

skip_lr_errors

Determines whether Replication Agent ignores log record processing errors.

Default
false

Values
true – enables Replication Agent to skip log record processing errors and continue replication.

false – disables Replication Agent from skipping log record processing errors.

Comments

- If you configure **skip_lr_errors** to true, Replication Agent logs the log record processing error encountered and also logs a warning that the error has been skipped. If the transaction ID, operation ID and locator of the log record are available at the time of the error, these are also logged. Replication Agent continues processing transaction log records.
- If you configure **skip_lr_errors** to false, Replication Agent throws an exception, stops all replication processing, and transitions to the Replication Down state.
- **skip_lr_errors** is intended only for troubleshooting with assistance from Sybase Technical Support.
- You can change **skip_lr_errors** only when Replication Agent is in the Admin or Replication Down state.

Warning! Use of this parameter does not guarantee that there will be no transaction loss, nor does it guarantee that the RASD is synchronized with the primary database when log record processing errors are skipped.

skip_ltl_errors

Determines whether Replication Agent ignores Log Transfer Language (LTL) error messages.

Warning! Using the **skip_ltl_errors** parameter incorrectly may cause data inconsistencies between the primary and replicate databases.

Default
false

Values
true – enables skipping LTL errors to continue replication.

false – disables skipping LTL errors.

Comments

- If the **skip_ltl_errors** configuration parameter is set to true, the Replication Agent instance logs any LTL error messages returned by Replication Server, along with the offending LTL commands, and then it continues processing transaction log records.
- If the **skip_ltl_errors** configuration parameter is set to false, the Replication Agent instance stops all of its replication processing and goes to Replication Down state when it receives an LTL error message and the error is unrecoverable.
- The **skip_ltl_errors** parameter is intended for troubleshooting only, with assistance from Sybase Technical Support.

ssl_identity_filename

ssl_identity_filename indicates the path to the Replication Agent instance identity file, a PKCS #12 file containing an asymmetric key pair used for SSL communication.

Default
`$SYBASE/RAX-15_5/instance_name/certificates/instance_name.p12`

Value
The path of the identity file.

Comments

- Adaptive Server® and Sybase OCS include **openssl**, **certreq**, **certauth**, and **certpk12** utilities in `$SYBASE/OCS-15_0/bin`. Sybase recommends that you use **openssl** to create the PKCS #12 file.

ssl_identity_password

ssl_identity_password is the passsword to access a Replication Agent instance identity file.

Default
<not configured>

Value
A valid password.

ssl_certificates_filename

ssl_certificates_filename indicates the path of the file containing Certificate Authority (CA) certificates included with the Replication Agent installation.

Default
`$SYBASE/RAX-15_5/config/trusted.txt`

Value
The path of the `trusted.txt`.

Comments

- `trusted.txt` contains a few global recognized CA certificates, including Thawte, Entrust, Baltimore, Verisign and RSA.
- After installing Replication Agent, `trusted.txt` is available in `$SYBASE/RAX-15_5/config`.
- This parameter need not be set unless you want to specify a different location or file.

structured_tokens

Determines whether Replication Agent uses LTL structured tokens.

Default
true

Values
true – enables LTL structured tokens.

false – disables LTL structured tokens.

Comments

- If the **structured_tokens** configuration parameter is set to **true**, the Log Transfer Interface (LTI) component uses LTL structured tokens when it generates LTL commands.
- Using structured tokens in the LTL can significantly improve overall replication system performance.
- Using structured tokens in the LTL can improve Replication Server performance, especially when non-Sybase datatypes in the primary database must be translated by Replication Server.
- To replicate columns that have one or more spaces in the column name, you must set the value of the **structured_tokens** parameter to true.

truncation_interval

Specifies a time interval between automatic truncations of the Replication Agent transaction log.

Warning! Replication Agent deletes the archive log files that it no longer needs. For more information, see the *Replication Agent Primary Database Guide*.

Default
0

Value
An integer from 0 to 720.

Comments

- The value of the **truncation_interval** parameter is the number of minutes between automatic transaction log truncations.
- Automatic transaction log truncation based on the value of the **truncation_interval** parameter takes place only when the value of the **truncation_type** parameter is **interval**.
- The maximum truncation interval is 720 minutes, or 12 hours.
- If the value of the **truncation_interval** parameter is 0 (zero) and the value of the **truncation_type** parameter is **interval**, automatic truncation is disabled.
- To truncate the transaction log manually, use the **pdb_truncate_xlog** command.

See also

- *pdb_archive_path* on page 167
- *truncation_type* on page 211
- *pdb_archive_remove* on page 168

truncation_type

Configures transaction log truncation behavior of Replication Agent.

Warning! Replication Agent deletes the archived log files that it no longer needs. For more information, see the *Replication Agent Primary Database Guide*.

Default
locator_update

Values
command – Replication Agent truncates the transaction log only when the **pdb_truncate_xlog** command is invoked.

When the value of the **truncation_type** parameter is **command**, the only way you can truncate the transaction log is by invoking the **pdb_truncate_xlog** command. No automatic truncation takes place when the value of the **truncation_type** parameter is **command**.

interval – Replication Agent automatically truncates the transaction log when determined by a configurable interval of time.

locator_update – Replication Agent automatically truncates the transaction log whenever it receives a new LTM Locator value from the primary Replication Server.

When the value of the **truncation_type** parameter is **locator_update**, the transaction log is automatically truncated when Replication Agent receives a new LTM Locator from the primary Replication Server.

Comments

Note: Truncation of the archive log files that Replication Agent no longer needs from the **pdb_archive_path** directory is performed only if the **pdb_archive_remove** parameter is **true**.

- Regardless of the value of the **truncation_type** parameter, you can truncate the Replication Agent transaction log manually at any time by invoking the **pdb_truncate_xlog** command.
- If the value of the **truncation_interval** parameter is 0 (zero) and the value of the **truncation_type** parameter is **interval** (the default values for both parameters), automatic truncation is disabled.
- Replication Agent receives a new LTM Locator based on the values of the **lti_update_trunc_point** parameter.

See also

- *pdb_archive_path* on page 167
- *pdb_archive_remove* on page 168
- *truncation_interval* on page 211

use_rssd

Determines whether Replication Agent uses replication definitions.

Default
true

Values
true – enables using replication definitions.

false – disables using replication definitions.

Comments

- If the value of the **use_rssd** parameter is true, the Replication Agent instance connects to the Replication Server System Database (RSSD) to retrieve replication definitions for the primary database automatically whenever it goes from Replication Down state to Replicating state (for example, when the **resume** command is invoked).
  - Each time it retrieves replication definitions, Replication Agent stores the information in a cache. Replication Agent uses replication definitions stored in its cache when it generates Log Transfer Language (LTL) commands.
  - If the Log Transfer Interface (LTI) component encounters an operation on a database object for which it does not have a cached replication definition, Replication Agent reconnects to the RSSD to update its replication definition cache.
  - If a replication definition still cannot be found for the operation, the Replication Agent instance suspends all of its replication operations and goes to Replication Down state.
- Replication Agent can use information in table and function replication definitions (that is, replication definitions for individual primary database objects) stored in the RSSD to generate more efficient LTL, and thus improve throughput in the LTI component and Replication Server.

  Accessing replication definitions in the RSSD enables the LTI component to improve performance by:
  - Omitting column names in LTL. When columns are sent in the order specified in the replication definition, column images can be sent without column names (headings), which reduces LTL overhead.
  - Omitting unneeded columns in LTL. When columns are sent as specified in the replication definition, images for unchanged columns need not be sent, which reduces LTL overhead.
  - Sending data for each column in the datatype specified by the replication definition. This allows data to be handled more efficiently all the way through the replication system.
  - Sending database object names in the same character case as defined in the replication definition.
- If the value of the **use_rssd** parameter is **false**, none of the previously described performance improvements are possible. In that case, Replication Agent sends all data as a `char` datatype in the LTL.

use_ssl

use_ssl indicates whether clients must use SSL to connect to Replication Agent.

Default
False

Value
True | **False**

Comments

- If this parameter is set to true, any clients must use SSL to connect to Replication Agent administration port. The Replication Agent certificate and private key in its identity file is used during SSL handshake.
- For **isql** as a client, if this parameter is set to true, the Replication Agent master and query entries in the `interfaces` file (UNIX) or `sql.ini` (Windows) must be appended with ssl. For example:

```
[SRVRA1]
master=NLWNSCK,localhost,13010,ssl
query=NLWNSCK,localhost,13010,ssl
```

Obtaining Help and Additional Information

Use the Sybase Getting Started CD, Product Documentation site, and online help to learn more about this product release.

- The Getting Started CD (or download) – contains release bulletins and installation guides in PDF format, and may contain other documents or updated information.
- Product Documentation at *http://sybooks.sybase.com/* – is an online version of Sybase documentation that you can access using a standard Web browser. You can browse documents online, or download them as PDFs. In addition to product documentation, the Web site also has links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, Community Forums/Newsgroups, and other resources.
- Online help in the product, if available.

To read or print PDF documents, you need Adobe Acrobat Reader, which is available as a free download from the *Adobe* Web site.

Note: A more recent release bulletin, with critical product or document information added after the product release, may be available from the Product Documentation Web site.

Technical Support

Get support for Sybase products.

If your organization has purchased a support contract for this product, then one or more of your colleagues is designated as an authorized support contact. If you have any questions, or if you need assistance during the installation process, ask a designated person to contact Sybase Technical Support or the Sybase subsidiary in your area.

Downloading Sybase EBFs and Maintenance Reports

Get EBFs and maintenance reports from the Sybase Web site or the SAP® Service Marketplace (SMP). The location you use depends on how you purchased the product.

- If you purchased the product directly from Sybase or from an authorized Sybase reseller:
  a) Point your Web browser to *http://www.sybase.com/support*.
  b) Select **Support > EBFs/Maintenance**.
  c) If prompted, enter your MySybase user name and password.
  d) (Optional) Select a filter, a time frame, or both, and click **Go**.
  e) Select a product.

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

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

- If you ordered your Sybase product under an SAP contract:
  a) Point your browser to *http://service.sap.com/swdc* and log in if prompted.
  b) Select **Search for Software Downloads** and enter the name of your product. Click **Search**.

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.

Glossary

This glossary describes Replication Server Options terms.

- **Adaptive Server® –** the brand name for Sybase relational database management system (RDBMS) software products.
  - Adaptive Server® Enterprise manages multiple, large relational databases for high-volume online transaction processing (OLTP) systems and client applications.
  - Sybase®IQ manages multiple, large relational databases with special indexing algorithms to support high-speed, high-volume business intelligence, decision support, and reporting client applications.
  - SQL Anywhere® (formerly Adaptive Server Anywhere) manages relational databases with a small DBMS footprint, which is ideal for embedded applications and mobile device applications.

  See also *DBMS* and *RDBMS*.
- **atomic materialization –** a materialization method that copies subscription data from a primary database to a replicate database in a single, atomic operation. No changes to primary data are allowed until the subscription data is captured at the primary database. See also *bulk materialization* and *nonatomic materialization*.
- **BCP utility –** a bulk copy transfer utility that provides the ability to load multiple rows of data into a table in a target database. See also *bulk copy*.
- **bulk copy –** an Open Client™ interface for the high-speed transfer of data between a database table and program variables. Bulk copying provides an alternative to using SQL **insert** and **select** commands to transfer data.
- **bulk materialization –** a materialization method whereby subscription data in a replicate database is initialized outside of the replication system. You can use bulk materialization for subscriptions to table replication definitions or function replication definitions. See also *atomic materialization* and *nonatomic materialization*.
- **client –** in client/server systems, the part of the system that sends requests to servers and processes the results of those requests. See also *client application*.
- **client application –** software that is responsible for the user interface, including menus, data entry screens, and report formats. See also *client*.
- **commit –** an instruction to the DBMS to make permanent the changes requested in a transaction. See also *transaction*. Contrast with *rollback*.
- **database –** a collection of data with a specific structure (or schema) for accepting, storing, and providing data for users. See also *data server*, *DBMS*, and *RDBMS*.
- **database connection –** a connection that allows Replication Server to manage the database and distribute transactions to the database. Each database in a replication system

can have only one database connection in Replication Server. See also *Replication Server* and *route*.

- **data client –** a client application that provides access to data by connecting to a data server. See also *client*, *client application*, and *data server*.
- **data distribution –** a method of locating (or placing) discrete parts of a single set of data in multiple systems or at multiple sites. Data distribution is distinct from data replication, although a data replication system can be used to implement or support data distribution. Contrast with *data replication*.
- **data replication –** the process of copying primary data to remote locations and synchronizing the copied data with the primary data. Data replication is different from data distribution. Replicated data is a stored copy of data at one or more remote sites throughout a system, and it is not necessarily distributed data. Contrast with *data distribution*. See also *transaction replication*.
- **data server –** a server that provides the functionality necessary to maintain the physical representation of a table in a database. Data servers are usually database servers, but they can also be any data repository with the interface and functionality a data client requires. See also *client*, *client application*, and *data client*.
- **datatype –** a keyword that identifies the characteristics of stored information on a computer. Some common datatypes are: *char*, *int*, *smallint*, *date*, *time*, *numeric*, and *float*. Different data servers support different datatypes.
- **DBMS –** an abbreviation for database management system, a computer-based system for defining, creating, manipulating, controlling, managing, and using databases. The DBMS can include the user interface for using the database, or it can be a standalone data server system. Compare with *RDBMS*.
- **ERSSD –** an abbreviation for Embedded Replication Server System Database, which manages replication system information for a Replication Server. See also *Replication Server*.
- **failback –** a procedure that restores the normal user and client access to a primary database, after a failover procedure switches access from the primary database to a replicate database. See also *failover*.
- **failover –** a procedure that switches user and client access from a primary database to a replicate database, particularly in the event of a failure that interrupts operations at the primary database, or access to the primary database. Failover is an important fault-tolerance feature for systems that require high availability. See also *failback*.
- **function –** a data server object that represents an operation or set of operations. Replication Server distributes operations to replicate databases as functions. See also *stored procedure*.
- **function string –** a string that Replication Server uses to map a function and its parameters to a data server API. Function strings allow Replication Server to support heterogeneous replication, in which the primary and replicate databases are different types, with different SQL extensions and different command features. See also *function*.

- **gateway –** connectivity software that allows two or more computer systems with different network architectures to communicate.
- **inbound queue –** a stable queue managed by Replication Server to spool messages received from a Replication Agent. See also *outbound queue* and *stable queue*.
- **interfaces file –** a file containing information that Sybase Open Client and Open Server™ applications need to establish connections to other Open Client and Open Server applications. See also *Open Client* and *Open Server*.
- **isql –** an Interactive SQL client application that can connect and communicate with any Sybase Open Server application, including Adaptive Server, Replication Agent, and Replication Server. See also *Open Client* and *Open Server*.
- **Java –** an object-oriented programming language developed by Sun Microsystems. A platform-independent, "write once, run anywhere" programming language.
- **Java VM –** the Java Virtual Machine. The Java VM (or JVM) is the part of the Java Runtime Environment (JRE) that is responsible for interpreting Java byte codes. See also *Java* and *JRE*.
- **JDBC –** an abbreviation for Java Database Connectivity. JDBC is the standard communication protocol for connectivity between Java clients and data servers. See also *data server* and *Java*.
- **JRE –** an abbreviation for Java Runtime Environment. The JRE consists of the Java Virtual Machine (Java VM or JVM), the Java Core Classes, and supporting files. The JRE must be installed on a machine to run Java applications, such as Replication Agent. See also *Java VM*.
- **LAN –** an abbreviation for "local area network," a computer network located on the user premises and covering a limited geographical area (usually a single site). Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary can be subject to some form of regulation. Contrast with *WAN*.
- **latency –** in transaction replication, the time it takes to replicate a transaction from a primary database to a replicate database. Specifically, latency is the time elapsed between committing an original transaction in the primary database and committing the replicated transaction in the replicate database.

  In disk replication, latency is the time elapsed between a disk write operation that changes a block or page on a primary device and the disk write operation that changes the replicated block or page on a replicate device.

  See also *transaction replication*.
- **LOB –** an abbreviation for large object, a large collection of data stored as a single entity in a database.
- **Log Reader –** an internal component of Replication Agent that interacts with the primary database to capture transactions for replication. See also *Log Transfer Interface* and *Log Transfer Manager*.

- **Log Transfer Interface –** an internal component of Replication Agent that interacts with Replication Server to forward transactions for distribution to Replication Server. See also *Log Reader* and *Log Transfer Manager*.
- **Log Transfer Language –** the proprietary protocol used between Replication Agent and Replication Server to replicate data from the primary database to Replication Server. See also *Log Reader* and *Log Transfer Interface*.
- **Log Transfer Manager –** an internal component of Replication Agent that interacts with the other Replication Agent internal components to control and coordinate Replication Agent operations. See also *Log Reader* and *Log Transfer Interface*.
- **maintenance user –** a special user login name in the replicate database that Replication Server uses to apply replicated transactions to the database. See also *replicate database* and *Replication Server*.
- **materialization –** the process of copying the data from a primary database to a replicate database, initializing the replicate database so that the replication system can begin replicating transactions. See also *atomic materialization*, *bulk materialization*, and *nonatomic materialization*.
- **Multi-Path Replication™ –** Replication Server feature that improves performance by enabling parallel paths of data from the source database to the target database. These multiple paths process data independently and are applicable when sets of data can be processed in parallel without transactional consistency requirements between them.
- **nonatomic materialization –** a materialization method that copies subscription data without a lock on the primary database. Changes to primary data are allowed during data transfer, which may cause temporary inconsistencies between the primary and replicate databases. Contrast with *atomic materialization*. See also *bulk materialization*.
- **ODBC –** an abbreviation for Open Database Connectivity, an industry-standard communication protocol for clients connecting to data servers. See also *client*, *data server*, and *JDBC*.
- **Open Client –** a Sybase product that provides customer applications, third-party products, and other Sybase products with the interfaces needed to communicate with Open Server applications. See also *Open Server*.
- **Open Client application –** An application that uses Sybase Open Client libraries to implement Open Client communication protocols. See also *Open Client* and *Open Server*.
- **Open Server –** a Sybase product that provides the tools and interfaces required to create a custom server. See also *Open Client*.
- **Open Server application –** a server application that uses Sybase Open Server libraries to implement Open Server communication protocols. See also *Open Client* and *Open Server*.
- **outbound queue –** a stable queue managed by Replication Server to spool messages to a replicate database. See also *inbound queue*, *replicate database*, and *stable queue*.
- **primary data –** the data source used for replication. Primary data is stored and managed by the primary database. See also *primary database*.

- **primary database –** the database that contains the data to be replicated to another database (the replicate database) through a replication system. The primary database is the source of replicated data in a replication system. Sometimes called the active database. Contrast with *replicate database*. See also *primary data*.
- **primary key –** a column or set of columns that uniquely identifies each row in a table.
- **primary site –** the location or facility at which primary data servers and primary databases are deployed to support normal business operations. Sometimes called the active site or main site. See also *primary database* and *replicate site*.
- **primary table –** a table used as a source for replication. Primary tables are defined in the primary database schema. See also *primary data* and *primary database*.
- **primary transaction –** a transaction that is committed in the primary database and recorded in the primary database transaction log. See also *primary database*, *replicated transaction*, and *transaction log*.
- **quiesce –** to cause a system to go into a state in which further data changes are not allowed. See also *quiescent*.
- **quiescent –** in a replication system, a state in which all updates have been propagated to their destinations. Some Replication Agent and Replication Server commands require that you first quiesce the replication system.

  In a database, a state in which all data updates are suspended so that transactions cannot change any data, and the data and log devices are stable.

  This term is interchangeable with quiesced and in quiesce. See also *quiesce*.
- **RASD –** an abbreviation for Replication Agent System Database. Information in the RASD is used by the primary database to recognize database structure or schema objects in the transaction log.
- **RCL –** an abbreviation for Replication Command Language, the command language used to manage Replication Server. See also *Replication Server*.
- **RDBMS –** an abbreviation for relational database management system, an application that manages and controls relational databases. Compare with *DBMS*. See also *relational database*.
- **relational database –** a collection of data in which data is viewed as being stored in tables, which consist of columns (data items) and rows (units of information). Relational databases can be accessed by SQL requests. Compare with *database*. See also *SQL*.
- **replicate data –** A set of data that is replicated from a primary database to a replicate database by a replication system. See also *primary database*, *replication system*, and *replicate database*.
- **replicate database –** a database that contains data replicated from another database (the primary database) through a replication system. The replicate database is the database that receives replicated data in a replication system. Contrast with *primary database*. See also *replicate data*, *replicated transaction*, and *replication system*.

- **replicated transaction –** a primary transaction that is replicated from a primary database to a replicate database by a transaction replication system. See also *primary database*, *primary transaction*, *replicate database*, and *transaction replication*.
- **replicate site –** the location or facility at which replicate data servers and replicate databases are deployed to support normal business operations during scheduled downtime at the primary site. Contrast with *primary site*. See also *replicate database*.
- **Replication Agent –** an application that reads a primary database transaction log to acquire information about data-changing transactions in the primary database, processes the log information, and then sends it to a Replication Server for distribution to a replicate database. See also *primary database* and *Replication Server*.
- **replication definition –** a description of a table or stored procedure in a primary database, for which subscriptions can be created. The replication definition, maintained by Replication Server, includes information about the columns to be replicated and the location of the primary table or stored procedure. See also *Replication Server* and *subscription*.
- **Replication Server –** a Sybase software product that provides the infrastructure for a transaction replication system. See also *Replication Agent*.
- **replication system –** a data processing system that replicates data from one location to another. Data can be replicated between separate systems at a single site, or from one or more local systems to one or more remote systems. See also *transaction replication*.
- **rollback –** an instruction to a database to back out of the changes requested in a unit of work (called a transaction). Contrast with *commit*. See also *transaction*.
- **route –** A one-way message stream from a primary Replication Server to a replicate Replication Server. Routes carry data-changing commands (including those for RSSDs) and replicated functions (database procedures) between separate Replication Servers. See also *Replication Server*.
- **RSSD –** an abbreviation for Replication Server System Database, which manages replication system information for a Replication Server. See also *Replication Server*.
- **SQL –** an abbreviation for Structured Query Language, a nonprocedural programming language used to process data in a relational database. ANSI SQL is an industry standard. See also *transaction*.
- **stable queue –** a disk device-based, store-and-forward queue managed by Replication Server. Messages written into the stable queue remain there until they can be delivered to the appropriate process or replicate database. Replication Server provides a stable queue for both incoming messages (the inbound queue) and outgoing messages (the outbound queue). See also *database connection*, *Replication Server*, and *route*.
- **stored procedure –** a data server object that represents an operation or set of operations. This term is often used interchangeably with *function*.
- **subscription –** a request for Replication Server to maintain a replicated copy of a table, or a set of rows from a table, in a replicate database at a specified location. See also *replicate database*, *replication definition*, and *Replication Server*.

- **table –** in a relational DBMS, a two-dimensional array of data or a named data object that contains a specific number of unordered rows composed of a group of columns that are specific for the table. See also *database*.
- **transaction –** a unit of work in a database that can include zero, one, or many operations (including **insert**, **update**, and **delete** operations), and that is either applied or rejected as a whole. Each SQL statement that modifies data can be treated as a separate transaction, if the database is so configured. See also *SQL*.
- **transactional consistency –** A condition in which all transactions in the primary database are applied in the replicate database, and in the same order that they were applied in the primary database.
- **transaction log –** generally, the log of transactions that affect the data managed by a data server. Replication Agent reads the transaction log to identify and acquire the transactions to be replicated from the primary database. See also *Replication Agent*, *primary database*, and *Replication Server*.
- **transaction replication –** a data replication method that copies data-changing operations from a primary database to a replicate database. See also *data replication*.
- **UDB –** IBM DB2 Universal Database (formerly IBM DB2 for Linux, UNIX, and Windows).
- **WAN –** an abbreviation for "wide area network," a system of local-area networks (LANs) connected together with data communication lines. Contrast with *LAN*.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W