Primary Database Guide

Replication Agent™ 15.7.1 SP200

Linux, Microsoft Windows, and UNIX

DOCUMENT ID: DC00269-01-1571200-01
LAST REVISED: March 2014

Contents

Conventions

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

Style conventions

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

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

Run:

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

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

Syntax conventions

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

Case-sensitivity

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

Terminology

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

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

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

Replication Agent for Oracle

Review the features of Replication Agent that are unique to Replication Agent for Oracle.

The term "Replication Agent for Oracle" refers to an instance of Replication Agent software that is installed and configured for a primary database that resides in an Oracle data server.

Note: For information on the basic features and operation of Replication Agent, see the *Replication Agent Administration Guide* and *Replication Agent Reference Manual*.

Oracle-Specific Considerations

These general issues and considerations are specific to using Replication Agent with the Oracle data server.

Unsupported Oracle Software Features

Review the Oracle features that are not supported by Replication Agent.

These features are not supported:

- Oracle virtual columns
- Oracle label security
- Oracle-packaged stored procedures and functions (standalone procedures and functions are supported)
- Oracle schema objects in encrypted tablespaces
- SAP® Replication Server® parallel DSI
- SAP Replication Server **rs_init** utility
- SAP Replication Server **rs_subcomp** utility
- SAP Replication Server automatic materialization
- SAP Replication Server when replicating in an environment where other vendors are replicating

Replication of Deferred Updates on Primary Keys

Updates to the unique column index in a table is not supported by traditional replication, and the Replication Server reports errors.

The replication of updates to the unique column index in a table is not supported, and Replication Server reports errors. For example, table `t` has a unique index on column `c`, with values 1, 2, 3, 4 and 5. A single **update** statement is applied to the table:

```
update t set c = c+1
```

Using traditional replication, this statement results in:

```
update t set c = 2 where c = 1
update t set c = 3 where c = 2
```

```
update t set c = 4 where c = 3
update t set c = 5 where c = 4
update t set c = 6 where c = 5
```

The first update attempts to insert a value of c=2 into the table. However, this value already exists in the table. Replication Server displays error 2601—an attempt to insert a duplicate key.

The Replication Server DSI stops working if you attempt to replicate updates to a non-SAP table that has a unique column index. To work around this problem, broaden the unique index definition.

Unsupported Oracle Datatypes, Data, and Structures

Review the Oracle datatypes that are not supported by Replication Agent.

These datatypes are not supported:

- Oracle-supplied datatypes:
  - "Any" Types (`SYS.ANYTYPE`, `SYS.ANYDATASET`), except for `SYS.ANYDATA`.
  - `MLSLABEL`
  - Spatial Types (`MDSYS.SDO_GEOMETRY`, `SDO_TOPO_GEOMETRY`, `SDO_GEORASTER`)
  - Media Types (`ORDSYS.ORDAudio`, `ORDSYS.ORDImage`, `ORDSYS.ORDImageSignature`, `ORDSYS.ORDVideo`, `ORDSYS.ORDDoc`, `SI_StillImage`, `SI_Color`, `SI_AverageColor`, `SI_ColorHistogram`, `SI_PositionalColor`, `SI_Texture`, `SI_FeatureList`)
  - Datatypes used with Oracle Expression Filter
  - `ANYDATA`, if it is being replicated to a non-`ANYDATA` column or if the data exceeds 16KB, which is the size constraint of the Replication Server `OPAQUE` datatype.
  - `BFILE`, `UROWID`, or `REF` data stored in an `ANYDATA` column
  - `BLOB` or object user-defined datatype data being replicated to a replicate database other than SAP® HANA® database, Oracle, or SAP® Adaptive Server® Enterprise (SAP® ASE). When replicating to Oracle, use ExpressConnect for Oracle since data of these types is not supported by Enterprise Connect™ Data Access (ECDA) for Oracle.
  - `REF`
  - `ROWID` argument is not supported, if `ROWID` is used as a searchable value in the stored procedure. Because `ROWID` is the physical location of a row which is unique to the primary database, not unique to the replicate database.
  - `UROWID`
- User-defined datatypes containing LOB data
- User-defined datatypes defined as NOT FINAL
- Partial updates to Oracle LOBs are not supported.
- Virtual columns – Replication Agent supports the replication of tables containing computed (or virtual) columns in Oracle 11g. However, the replication of individual

computed columns is not supported. You can mark tables with virtual columns for replication using the **force** option, but the virtual columns are not replicated.

Data stored in the Oracle XML DB Repository is not supported if it is accessed with these protocols or methods:

- Internet protocols like FTP, HTTP, HTTPS, and WebDAV
- The Oracle XML DB Repository API

These user-defined object types and structures are not supported:

- Associative arrays
- Nested tables
- `VARRAY`
- Nested tables or `VARRAY` stored in an `ANYDATA` column

Predefined PL/SQL Numeric Datatypes

Replication Agent does not support marking procedures containing the `PLS_INTEGER` or `BINARY_INTEGER` predefined PL/SQL numeric datatypes.

Replication Agent does support marking procedures with the `SIMPLE_INTEGER` datatype, which is a subtype of `PLS_INTEGER`. However, Replication Agent does not support marking procedures containing any other subtypes of the aforementioned types, including `NATURAL`, `NATURALN`, `POSITIVE`, `POSITIVEN`, and `SIGNTYPE`.

Supported Oracle Datatypes, Data, and Structures

Replication Agent supports the replication of encrypted data, compressed data, and SecureFiles.

Encrypted Data

Replication Agent transparently supports the replication of data in encrypted columns, partitions, and tablespaces for Oracle versions 11g Release 1 and later.

If your primary Oracle database version is earlier than 11g Release 1, you can still mark tables containing encrypted columns, partitions, and tablespaces using the **force** option, but the columns are not replicated.

Compressed Data

Replication Agent supports the replication of compressed tables, compressed tablespaces, and data compressed for Oracle direct-load operations (bypassing I/O buffers) for Oracle versions 11g Release 2 and later.

Note: If you want to replicate tables compressed for online transaction processing (OLTP), apply first the Oracle patch# 129050503.

SecureFile Data

Replication Agent supports the replication of Oracle data stored with the SecureFile option.

Replication Agent supports the replication of Oracle LOB data stored with the SecureFile option for Oracle versions 11g Release 2 and later. If your primary Oracle database version is earlier than 11g Release 2, you can still mark tables containing SecureFile LOB columns using the **force** option, but the columns are not replicated.

Replication Agent Connectivity

Replication Agent for Oracle uses JDBC™ for communications with all replication system components.

The Oracle JDBC driver must be installed on the Replication Agent host machine, and the directory this driver is installed in must be in the CLASSPATH environment variable.

Install the `xstreams.jar` for Oracle XStream JDBC driver in the CLASSPATH environment variable. See *Setting Up an Oracle XStream JDBC Driver* in the *Replication Agent Installation Guide*.

The TNS Listener Service must be installed and running on the primary database so the Replication Agent instance can connect to it. See the *Oracle Database Net Services Administrator's Guide*. See the *Replication Agent Installation Guide* for the specific JDBC driver and version to install.

Replication Agent Permissions

Replication Agent for Oracle uses the **pds_username** to connect to Oracle and must have certain Oracle permissions.

These permissions are required:

- **GRANT ALTER ANY PROCEDURE** – required to manage procedures for replication.
- **GRANT ALTER DATABASE** – required for Replication Agent to read from a Data Guard standby database transaction log.
- **GRANT ALTER ON** *TABLE_NAME* – required to replicate user-defined datatypes if table-level supplemental logging has not been enabled for the specified *TABLE_NAME*.
- **GRANT ALTER SYSTEM** – required to perform redo log archive operations.
- **GRANT CREATE ANY PROCEDURE** – required to mark procedures for replication.
- **GRANT CREATE PROCEDURE** – required to create **rs_marker** and **rs_dump proc** procedures.
- **GRANT CREATE PUBLIC SYNONYM** – required to create synonyms for created tables in the primary database.
- **GRANT CREATE SEQUENCE** – required to support replication.
- **GRANT CREATE SESSION** – required to connect to Oracle.
- **GRANT CREATE TABLE** – required to create tables in the primary database.
- **GRANT DROP PUBLIC SYNONYM** – required to drop created synonyms.
- **GRANT EXECUTE_CATALOG_ROLE** – required to use Oracle LogMiner.

- **GRANT EXECUTE ON DBMS_FLASHBACK** – required to execute **DBMS_FLASHBACK.get_system_change_number.**
- **GRANT EXECUTE ON SYS.DBMS_LOCK** – required to generate **commit** log records at the primary database.
- **GRANT SELECT ANY TRANSACTION** – required to use Oracle LogMiner.
- **GRANT SELECT_CATALOG_ROLE** – required to select from the **DBA_* views**.
- **GRANT SELECT ON SYS.OPQTYPE$** – required for DDL replication and `XMLTYPE` data replication.
- **GRANT SELECT ON SYS.RECYCLEBIN$** – required to use Oracle Flashback with Replication Agent.
- **GRANT SELECT ON SYS.ARGUMENT$** – required to process procedure DDL commands.
- **GRANT SELECT ON SYS.CCOL$** – required to support table replication (column constraint information).
- **GRANT SELECT ON SYS.CDEF$** – required to support table replication (constraint information).
- **GRANT SELECT ON SYS.COL$** – required to support table replication (column information).
- **GRANT SELECT ON SYS.COLLECTION$** – required to support table replication.
- **GRANT SELECT ON SYS.COLTYPE$** – required to support table replication.
- **GRANT SELECT ON SYS.CON$** – required to support table replication (constraint information).
- **GRANT SELECT ON SYS.DEFERRED_STG$** – required to suppress replication of compressed tables on Oracle 11g Release 2, on which LogMiner does not support compressed tables.
- **GRANT SELECT ON SYS.IND$** – required to identify indexes.
- **GRANT SELECT ON SYS.INDCOMPART$** – required to identify indexes.
- **GRANT SELECT ON SYS.INDPART$** – required to identify indexes.
- **GRANT SELECT ON SYS.INDSUBPART$** – required to identify indexes.
- **GRANT SELECT ON SYS.LOB$** – required for LOB replication support.
- **GRANT SELECT ON SYS.LOBCOMPPART$** – required to support partitioned LOB replication.
- **GRANT SELECT ON SYS.LOBFRAG$** – required to support partitioned LOB replication.
- **GRANT SELECT ON SYS.MLOG$** – required to filter materialized view log tables.
- **GRANT SELECT ON SYS.NTAB$** – required to support table replication.
- **GRANT SELECT ON SYS.OBJ$** – required for processing procedure DDL commands in the repository.
- **GRANT SELECT ON SYS.PROCEDUREINFO$** – required for procedure replication support.
- **GRANT SELECT ON SYS.SEG$** – required to suppress replication of compressed tables on versions of Oracle where LogMiner does not support compressed tables.
- **GRANT SELECT ON SYS.SEQ$** – required to support sequence replication.

- **GRANT SELECT ON SYS.SNAP$** – required to filter out materialized view tables.
- **GRANT SELECT ON SYS.TAB$** – required to support table replication.
- **GRANT SELECT ON SYS.TABCOMPART$** – required to support partitioned table replication.
- **GRANT SELECT ON SYS.TABPART$** – required to support partitioned table replication.
- **GRANT SELECT ON SYS.TABSUBPART$** – required to support partitioned table replication.
- **GRANT SELECT ON SYS.TS$** – required to identify tablespace encryption in Oracle 11g.
- **GRANT SELECT ON SYS.TYPE$** – required to process Oracle predefined and user-defined types.
- **GRANT SELECT ON SYS.USER$** – required for Oracle user identification.
- **GRANT SELECT ON SYS.ATTRIBUTE$** – required to process Oracle types.
- **GRANT SELECT ON V_$LOGMNR_CONTENTS** – required to use Oracle LogMiner.
- **GRANT SELECT ON V_$LOGMNR_LOGS** – required to use Oracle LogMiner.
- **GRANT SELECT ON SYS.PARTOBJ$** – required to support partitioned table replication.
- **GRANT SELECT ON SYS.ICOL$** – required to support the use of a unique index on columns as the primary key of the replication definition when there is no primary key defined for that table.

For supplemental logging at table level, the script generated from either **ra_admin prepare** for a new instance, or from **ra_admin supplemental_logging_level, table**, turns on table-level supplemental logging of these Oracle system tables:

- SYS.ARGUMENT$
- SYS.ATTRIBUTE$
- SYS.COLLECTION$
- SYS.COLTYPE$
- SYS.DEFERRED_STG$
- SYS.INDCOMPART$
- SYS.INDPART$
- SYS.INDSUBPART$
- SYS.LOB$
- SYS.LOBCOMPPART$
- SYS.LOBFRAG$
- SYS.MLOG$
- SYS.NTAB$
- SYS.OPQTYPE$
- SYS.PROCEDUREINFO$
- SYS.RECYCLEBIN$
- SYS.SEQ$
- SYS.SNAP$

- SYS.TABPART$
- SYS.TABCOMPART$
- SYS.TABSUBPART$
- SYS.TYPE$

If the Replication Agent is configured to remove old archive files, the user must have **UPDATE** authority to the directory and the archive log files.

Replication Agent for Oracle requires the **ALTER SYSTEM** privilege to issue the **ALTER SYSTEM ARCHIVE LOG** command. If Replication Agent is configured to access only online Oracle redo logs, Replication Agent issues the **ALTER SYSTEM ARCHIVE LOG SEQUENCE** command when the online redo log is no longer needed for replication (as when all data from the log has been replicated). Regardless of online or archive log processing, Replication Agent uses the **ALTER SYSTEM** privilege to issue the **ALTER SYSTEM ARCHIVE LOG CURRENT** command when Replication Agent is instructed to move processing to the end of the Oracle log. By issuing the **ALTER SYSTEM ARCHIVE LOG CURRENT** command, Replication Agent ensures that the current redo log file does not contain old data. Replication Agent moves processing to the end of the Oracle redo log when requested by the **move_truncpt** option of the **ra_locator** command. Replication Agent may also move processing to the end of the Oracle redo log during migration from one version of Replication Agent to another.

Oracle 10g, 11g, and12c Privileges for the ra_admin_owner User
If you configure the **ra_admin_owner** user, these permissions are required:

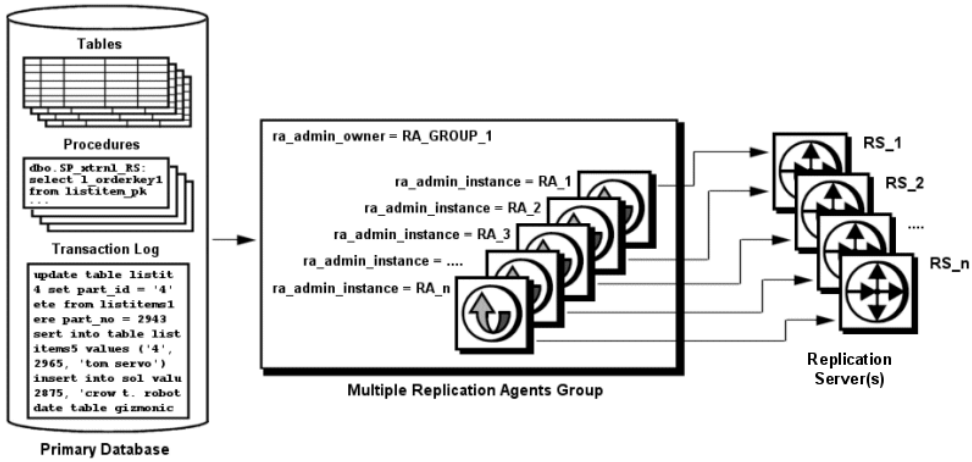
- **GRANT CREATE SESSION**
- **GRANT CREATE TABLE**
- **GRANT CREATE SEQUENCE**
- **GRANT CREATE ANY PROCEDURE**
- **GRANT SELECT_CATALOG_ROLE**

Multiple Replication Agents

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

Using multiple Replication Agent instances with Replication Server Multi-Path Replication™ enables the replication of data through different streams, maintaining data consistency within a path but not adhering to the commit order across different paths. Replication Agent for Oracle supports the use of multiple Replication Agent instances to replicate Oracle tables and procedures.

Figure 1: Multipath Replication



Note: All Replication Agent instances in a common Replication Agent group must be of the same version.

For a description of end-to-end Multi-Path Replication scenarios, see the *Replication Server Heterogeneous Replication Guide*.

Configuration

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

- Every Replication Agent for Oracle instance in a Replication Agent group must use the same value for the **ra_admin_owner** and **ra_admin_prefix** parameters.
- Each Replication Agent for Oracle instance in a Replication Agent group must use a unique value for the **ra_admin_instance_prefix** parameter. This parameter distinguishes Replication Agent for Oracle instances within a group.

See the *Replication Agent Reference Manual*.

DDL Replication with Multiple Replication Agents

You can configure each Replication Agent instance to replicate all DDL operations or no DDL operations in the primary database.

In a Replication Agent group, you can also configure each Replication Agent instance to replicate:

- DDL operations for all objects marked for replication by the instance
- DDL operations for all objects marked for replication by the instance and DDL operations for objects not marked for replication by any instances

To avoid DDL synchronization errors with multiple Replication Agents, use these keywords with the **pdb_setrepddl** command:

- **all** specifies that all DDL operations are to be replicated by the instance.
- **marked** specifies that only DDL operations for objects that have been marked for replication by the instance are to be replicated.
- **unmarked** specifies that DDL operations for objects that have not been marked for replication by any instance are to be replicated. Nonschema DDL operations are also replicated by an instance that specifies this keyword.

Example

A Replication Agent group containing three instances replicates DDL operations performed on these tables in the primary database:

- `table_A` is marked for replication by instance `ra_instance_1`.
- `table_B` is marked for replication by instance `ra_instance_2`.
- `table_C` is marked for replication by instance `ra_instance_3`.
- `table_D` is not marked for replication by any instance.

The Replication Agent instances use the **pdb_setrepddl** command to specify which DDL to replicate:

```
ra_instance_1>pdb_setrepddl enable, marked
ra_instance_1>pdb_setrepddl enable, unmarked
ra_instance_2>pdb_setrepddl enable, marked
ra_instance_3>pdb_setrepddl enable, marked
```

The DDL is replicated as follows:

Replication Agent Instance	DDL Replicated
`ra_instance_1`	DDL for `table_A`, `table_D`, and nonschema DDL operations
`ra_instance_2`	DDL for `table_B`
`ra_instance_3`	DDL for `table_C`

High Availability

A primary data server may employ a high-availability solution, such as failover clustering, to minimize downtime in the event of hardware or software failure.

Although Replication Agent does not provide any high-availability solutions, it works with a third-party high-availability solution for the primary database if:

- Replication Agent is installed on a shared file system, such as OCFS, a Network File System (NFS), or a Veritas Cluster Server (VCS). The Replication Agent binaries, configuration files, and RASD files must be installed on the system.

- A third-party cluster-management solution—such as Sun Cluster Manager, Veritas Cluster Manager, or Oracle Cluster Ready Services (CRS) —is used to automatically start Replication Agent in the event of failover.

Redo and Archive Log Setup

By default, you can access both online and archive logs. You can configure Replication Agent to access only the online logs, but doing so requires you to turn Oracle automatic archiving off and requires Replication Agent to issue manual archive log commands to Oracle.

Note: If Replication Agent for Oracle is configured to truncate Oracle archive logs directly —**rman_enabled** is set to false and **pdb_archive_remove** is set to true—it must be installed on a host that has direct access to the Oracle archive log files.

When you add or delete the Oracle archive logs without updating their statuses in `v$archived_log` view, the Replication Agent for Oracle Log Reader cannot control log files that are being loaded.

Therefore, once you remove or add Oracle archive logs, you must execute **RMAN crosscheck archivelog all** command to make sure the `v$archived_log` view is up-to-date.

Archive Log Access

When you are using the default, archive log files will be accessed. The Replication Agent for Oracle uses the Oracle LogMiner to capture replicated transactions. The LogMiner automatically loads the log files. By default, an Oracle instance creates multiple directories under the flash recovery area specified by the **DB_RECOVERY_FILE_DEST** parameter of the Oracle **ALTER SYSTEM** command, each directory corresponding to and named after a separate day.

When there are multiple archive path, log file are loaded from one of the archive paths being in a prioritized order such as in **LOG_ARCHIVE_DEST_n** or alphabetically. If a log file is missing from the first prioritized path, LogMiner loads the log file from the next prioritized archive path.

Tip: To be sure that Replication Agent for Oracle does not read the archived redo logs from the incorrect location, set the **configuration pdb_archive_path** to read the first prioritized path in the Replication Agent.

Note: To prevent conflicts with other archive file processes, you may want to configure Oracle to duplex the archive log files into an additional destination directory that is used only for replication.

For information on specifying archive log destinations for your Oracle environment, see the Oracle **ALTER SYSTEM** command and **LOG_ARCHIVE_DEST_n** parameter.

Note: Accessing Oracle archived redo logs that are stored as file-system files is discussed in this section. If the archived redo logs are stored using the Oracle ASM, see the discussion of Automatic Storage Management.

Replication Agent for Oracle requires that redo log archiving is enabled at your Oracle database:

```
alter database ARCHIVELOG;
```

Note: If you are using Oracle Real Application Clusters (RAC), you must enable redo log archiving for each instance in the cluster.

Verify that log archiving is enabled:

```
select log_mode from v$database;
```

If you are using Oracle RAC, use this SQL statement to verify that log archiving has been enabled:

```
select instance, name, log_mode from gv$database;
```

If ARCHIVELOG (ARCHIVELOG or MANUAL in Oracle 10g) is returned, then log archiving is enabled.

Archive Log File Access

In the Replication Agent, set the **pdb_archive_path** configuration property to the expected location of archived redo log files. You can also set the Replication Agent **pdb_archive_remove** configuration parameter to true to allow the Replication Agent to remove these archive log files when they are no longer needed to support replication.

The **rman_enabled** parameter enables Replication Agent to use the Oracle **RMAN** utility to truncate old archive log files. See the *Replication Agent Reference Manual*.

Replication Agent Archive Setup

When **pdb_include_archives** is set to true (the default), Replication Agent does not archive, and SAP recommends that you configure Oracle to perform automatic archiving of redo logs.

When the **pdb_include_archives** configuration parameter is set to false, Replication Agent for Oracle also requires you to disable automatic archiving of Oracle redo logs. Archiving is performed manually by Replication Agent as the data in the online redo log files is replicated.

Replication Agent for Oracle requires these settings in your Oracle database depending on the Oracle version.

See also

- *Automatic Storage Management* on page 49

Querying Oracle Archive Log History

To determine the performance of Replication Agent for Oracle, review the number of archive logs that are generated by Oracle in a period of time and the size of those archive logs.

The archive log history information indicates how much data Replication Agent for Oracle requires to process a replication request.

To determine the archive log history, execute the query against the primary Oracle database by replacing the completion time with the desired time:

```
set linesize 1024
set pagesize 1000
select
      sequence# LSN,
      to_char(first_time, 'mm/dd/yy hh24:mi:ss') first_time,
      to_char(next_time, 'mm/dd/yy hh24:mi:ss') next_time,
      to_char(completion_time, 'mm/dd/yy hh24:mi:ss')
completion_time,
      blocks,
      block_size,
      first_change#,
      next_change#,
     name
from v$archived_log
where completion_time > to_date('01/01/14 00:00:00', 'mm/dd/yy
hh24:mi:ss')
and name like '/oracle/PRD/oraarch%'
order by sequence#;
```

where `01/01/14 00:00:00` is the desired time.

See the Oracle documentation for parameter descriptions.

Disabling Automatic Archiving for Oracle 12c

Disable automatic archiving for Oracle 12c.

1. To change the LOG_ARCHIVE_START parameter, either manually edit the server start-up parameter file, or enter:

   ```
   alter system set log_archive_start=false scope=spfile;
   ```

2. To check the setting of the LOG_ARCHIVE_START parameter, enter:

   ```
   select value from v$system_parameter where name =
   'log_archive_start';
   ```

3. If false is returned, the value in the server parameter file has been correctly modified to prevent automatic archiving when you restart the Oracle server. For information about the LOG_ARCHIVE_START parameter or the ALTER SYSTEM commands, see the *Oracle Database Reference Guide*.

4. Automatic archiving must be disabled in the active server and when you restart the Oracle server. To stop automatic archiving in the active server, enter:

   ```
   alter system archive log stop;
   ```

5. To disable automatic archiving when you restart the Oracle server, change the value of the server LOG_ARCHIVE_START parameter to false.

   Note: If **pdb_include _archives** is set to false: For redo log file processing after Replication Agent for Oracle is initialized, automatic archiving must never be enabled, even temporarily. If automatic archiving is enabled or if manual archiving is performed, causing a redo log file not yet processed by the Replication Agent to be overwritten, the

data in the lost redo log file is not replicated. You can recover from this situation by reconfiguring the Replication Agent to access archive log files. Set **pdb_include_archives** to true, set **pdb_archive_path** to the directory location that contains the archive of the file that has been overwritten, and resume. After catching up, suspend the Replication Agent, and reset **pdb_include_archives** to false.

Disabling Automatic Archiving for Oracle 11g

Disable automatic archiving for Oracle 11g.

1. To change the LOG_ARCHIVE_START parameter, either manually edit the server start-up parameter file, or enter:

   ```
   alter system set log_archive_start=false scope=spfile;
   ```

2. To check the setting of the LOG_ARCHIVE_START parameter, enter:

   ```
   select value from v$system_parameter where name =
   'log_archive_start';
   ```

3. If false is returned, the value in the server parameter file has been correctly modified to prevent automatic archiving when you restart the Oracle server. For information about the LOG_ARCHIVE_START parameter or the ALTER SYSTEM commands, see the *Oracle Database Reference Guide*.
4. Automatic archiving must be disabled in the active server and when you restart the Oracle server. To stop automatic archiving in the active server, enter:

   ```
   alter system archive log stop;
   ```

5. To disable automatic archiving when you restart the Oracle server, change the value of the server LOG_ARCHIVE_START parameter to false.

Note: If **pdb_include _archives** is set to false: For redo log file processing after Replication Agent for Oracle is initialized, automatic archiving must never be enabled, even temporarily. If automatic archiving is enabled or if manual archiving is performed, causing a redo log file not yet processed by the Replication Agent to be overwritten, the data in the lost redo log file is not replicated. You can recover from this situation by reconfiguring the Replication Agent to access archive log files. Set **pdb_include_archives** to true, set **pdb_archive_path** to the directory location that contains the archive of the file that has been overwritten, and resume. After catching up, suspend the Replication Agent, and reset **pdb_include_archives** to false.

Disabling Automatic Archiving for Oracle 10g

Disable automatic archiving for Oracle 10g.

1. Make sure you have **sysdba** administrator privileges, and close the database.
2. Enter:

   ```
   alter database ARCHIVELOG MANUAL;
   ```

3. To verify that log archiving is disabled, enter:

   ```
   select log_mode from v$database;
   ```

If MANUAL is returned, then automatic log archiving is disabled.

Forcing the Logging of All Database Changes

SAP recommends that you enable forced logging of all database changes to the Oracle redo log file to ensure that all data that should be replicated is logged.

1. In the primary database, execute:

   ```
   alter database FORCE LOGGING;
   ```

2. Verify the current setting by executing:

   ```
   select force_logging from v$database;
   ```

UNC Paths for Windows Archive and Online Redo Log Paths

If Replication Agent for Oracle is running as a Windows service and the primary Oracle data server is installed on a separate machine, configure the archive and online redo log paths according to the Microsoft Windows Universal Naming Convention (UNC):

```
\\oracle_server_machine\oracle_log_path
```

where *oracle_server_machine* is where the primary Oracle data server resides, and *oracle_log_path* is the archive or redo log file. For example, to set the location of archive redo log files to the `oracle` directory on the machine named `labratx64`, enter:

```
1> ra_config pdb_archive_path, \\labratx64\oracle
2> go
```

Configuring Oracle LogMiner

Configure Oracle LogMiner on the primary Oracle database.

1. Go to `$ORACLE_HOME/rdbms/admin`.
2. Log in as a “sys as sysdba” user.
3. Execute the Oracle LogMiner installation script:

   ```
   @dbmslm.sql
   ```

4. 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 step is required if you are using Oracle 10g.

5. Grant these privileges to **pds_username**:

   - **EXECUTE_CATALOG_ROLE**
   - **SELECT ON V_$LOGMNR_CONTENTS**
   - **SELECT ON V_$LOGMNR_LOGS**
   - **SELECT ANY TRANSACTION**

Note: The **ra_migrate** command verifies that these privileges have been granted to **pds_username**. 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.

Supplemental Logging

Enable supplemental logging and supplemental logging of primary key data and index columns.

To enable supplemental logging, execute these Oracle commands:

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE
 INDEX) COLUMNS;
```

To verify that minimal supplemental logging and supplemental logging of primary key and unique index information is enabled, enter:

```
select SUPPLEMENTAL_LOG_DATA_MIN, SUPPLEMENTAL_LOG_DATA_PK,
SUPPLEMENTAL_LOG_DATA_UI from v$database;
```

If YES is returned for each column, supplemental logging of primary key information is enabled.

Table-Level Supplemental Logging

To replicate updates to user-defined object type attributes, Replication Agent must enable table-level supplemental logging.

Manually enable table-level supplemental logging by entering:

```
ALTER TABLE THE_TABLE ADD SUPPLEMENTAL LOG DATA (ALL)
COLUMNS;
```

where *THE_TABLE* is the name of the table on which supplemental logging is being enabled. Verify that table-level supplemental logging has been enabled by executing:

```
select count(*) from ALL_LOG_GROUPS where
LOG_GROUP_TYPE='ALL COLUMN LOGGING' and OWNER=THE_OWNER
and TABLE_NAME=THE_TABLE
```

where *THE_OWNER* is the table owner. If this command returns a value of 1, table-level supplemental logging has been enabled for this table.

You can also enable supplemental logging from Replication Agent for Oracle using the **ra_set_autocorrection** command as described in the *Replication Agent Reference Manual*.

DDL Replication

Replication of data definition language (DDL) commands is supported, but only to Oracle databases. You cannot replicate DDL commands from Oracle to non-Oracle replicate databases.

Replication of DDL commands is enabled or disabled in Replication Agent using the **pdb_setrepddl** command. Replication Agent for Oracle can disable or enable the replication of specific DDL commands by object, owner, statement, or user. Replication Server uses the **ddl_username** parameter to execute DDL commands in the replicate database as the same user who executed the DDL commands in the primary database.

Note: By default, Replication Agent for Oracle filters SYS user DDL and if you want to replicate the SYS user DDL, you can manually remove the filter.

See *Replication Agent Reference Manual > Command Reference > pdb_setrepddl* and *Replication Agent Reference Manual > Configuration Parameters > ddl_username* for details on using **pdb_setrepddl** and **ddl_username**.

Oracle 10g, 11g, and 12c Privileges for DDL Replication

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

Different versions of Oracle have different permission requirements. For Oracle 10g, 11g, and 12c grant the DDL user permission to execute these commands:

- **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**

DDL Parameters

Set the **ddl_username** and **ddl_password** parameters.

To replicate DDL in Oracle, use **pdb_setrepddl** to set filtering rules accordingly. You must also set the Replication Agent **ddl_username** and **ddl_password** parameters. **ddl_username** is the database user name used to execute the replicated DDL command at the target database. This user must have permission to execute all replicated DDL commands at the target database. **ddl_password** is the corresponding password for **ddl_username**. In addition, the **ddl_username** database user must have permission to issue the **ALTER SESSION SET CURRENT_SCHEMA** command for any primary database user that might issue a DDL command to be replicated. See the *Replication Agent Reference Manual*.

Special Usage Notes

You do not need to set the **ddl_username** and **ddl_password** parameters if the Replication Server parameter **dsi_replication_ddl** is **on**.

The value of **ddl_username** cannot be the same as the maintenance user defined in Replication Server for the replicate connection. If these names are the same, a Replication Server error results.

The value of the **ddl_username** parameter is sent in the LTL for all replicated DDL statements. When DDL is replicated, Replication Server connects to the replicate database using the user

ID and password specified by the **ddl_username** and **ddl_password** parameters. Replication Server then issues:

```
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 user ID specified in **ddl_username** does not have permission to issue the **ALTER SESSION SET CURRENT_SCHEMA** or to execute the DDL command against the user schema, the command fails.

Note: To replicate DDL, Replication Server must have a database-level replication definition with **replicate DDL** set in the definition. See the *Replication Server Reference Manual*.

DDL Commands and Objects Filtered from Replication

Some Oracle DDL commands and objects are not replicated.

These DDL commands are not replicated:

- **alter database**
- **alter rollback segment**
- **alter session**
- **alter snapshot**
- **alter snapshot log**
- **alter system**
- **alter tablespace**
- **analyze**
- **audit**
- **create control file**
- **create database link**
- **create pfile from spfile**
- **create rollback segment**
- **create schema authorization**
- **create snapshot**
- **create snapshot log**
- **create spfile from pfile**
- **create tablespace**
- **drop database link**
- **drop rollback segment**
- **drop snapshot**
- **drop snapshot/log**
- **drop tablespace**
- **explain**

- **lock table**
- **no audit**
- **rename**
- **set constraints**
- **set role**
- **set transaction**

Any objects that are owned by SYS are not replicated. Any object owned by users defined in the list of nonreplicated users is not replicated. You can modify this list using the **pdb_ownerfilter** command. In addition, SAP has provided a default list of owners whose objects are not replicated. However, you cannot remove the SYS owner. Use the **pdb_ownerfilter** command to return, add, or remove the list of owners whose objects are not replicated. See the *Replication Agent Reference Manual*.

Note: The **truncate table** command is replicated as **rs_truncate**.

Character Case of Database Object Names

Database object names must be delivered to the primary Replication Server in the same format as specified in replication definitions; otherwise, replication fails. For example, if a replication definition specifies a table name in all lowercase, then that table name must appear in all lowercase when it is sent to the primary Replication Server by the Replication Agent.

To control the way Replication Agent treats the character case of database object names sent to the primary Replication Server, set the **ltl_character_case** configuration parameter to one of these values:

- **asis** – (the default) database object names are passed to Replication Server in the same format as stored in the primary data server.
- **lower** – database object names are passed to Replication Server in all lowercase, regardless of how they are stored in the primary data server.
- **upper** – database object names are passed to Replication Server in all uppercase, regardless of how they are stored in the primary data server.

In the Oracle data server, by default, database object names are stored in all uppercase. However, if you create a case-sensitive name, the case-sensitivity is retained in Oracle.

These examples use the **asis** option:

- `create table tabA` is stored as `TABA`
- `create table Tabb` is stored as `TABB`
- `create table 'TaBc'` is stored as `TaBc`

These examples use the **upper** option:

- `create table tabA` is rendered in LTL as `TABA`
- `create table Tabb` is rendered in LTL as `TABB`

- `create table 'TaBc'` is rendered in LTL as `TABC`

Format of Origin Queue ID

Each record in the transaction log is identified by an origin queue ID that consists of 64 hexadecimal characters (32 bytes). The format of the origin queue ID is determined by the Replication Agent instance and varies according to the primary database type.

Table 1. Replication Agent for Oracle Origin Queue ID Format

Character	Bytes	Description
0–3	2	Database generation ID
4–15	6	System change number
16–19	2	System change number generation ID
20–23	2	Redo log thread
24–43	10	Redo log record block address
44–55	6	System change number of the oldest active transaction begin
56–63	4	Locator ID

LTL Origin Commit Time Granularity

For Oracle, the precision of the origin commit time does not include milliseconds.

Replication Agent retrieves the origin commit time from the Oracle redo log. Timestamps in the redo log have a granularity only of seconds, not milliseconds.

Replication Server and RSSD Scripts

Replication Agent provides supplemental scripts to support additional Replication Server user-defined datatypes for Oracle datatypes and the replication of DDL commands.

These Replication Server scripts are shipped with Replication Agent and must be applied when the installed Replication Server is version 15.0.1 or earlier:

- `$SYBASE/RAX-15_5/scripts/oracle/hds_oracle_new_setup_for_replicate.sql`
- `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_1_rs.sql`
- `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_2_rssd.sql`
- `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_error_class_3_rs.sql`

Manually run these Replication Server scripts against the RSSD when the installed Replication Server is version 15.0.1 or earlier:

- `$SYBASE/RAX-15_5/scripts/oracle/hds_oracle_funcstrings.sql`
- `$SYBASE/RAX-15_5/hds_oracle_udds.sql`
- `$SYBASE/RAX-15_5/hds_clt_ase_to_oracle.sql`

Applying Script Changes for User-Defined Datatypes

Apply these script changes to use Oracle user-defined datatypes.

To use Oracle user-defined datatypes:

1. If your Replication Server is version 15.0.1 or earlier, apply this script to support replication of DDL to an Oracle replicate database:

   ```
   $SYBASE/RAX-15_5/scripts/oracle/
   hds_oracle_new_setup_replicate.sql
   ```

   This script defines Replication Server objects that must be created in the replicate database. Use this script instead of the `hds_oracle_setup_replicate.sql` script provided in the Replication Server install directory. This revised script contains additional changes to support Oracle-to-Oracle DDL replication.
2. To correctly define the Oracle error class for Replication Server 15.0.1 or an earlier version:
   - Apply this script at Replication Server:
     `$SYBASE/RAX-15_5/scripts/oracle/`
     `oracle_create_error_class_1_rs.sql`
   - Apply this script against your RSSD:
     `$SYBASE/RAX-15_5/scripts/oracle/`
     `oracle_create_error_class_2_rssd.sql`
   - Apply this script at Replication Server:
     `$SYBASE/RAX-15_5/scripts/oracle/`
     `oracle_create_error_class_3_rs.sql`

See *Replication Server Heterogeneous Replication Guide > Oracle Primary Data Server Issues*.

Oracle Datatype Compatibility

Replication Agent for Oracle processes Oracle transactions and passes data to the primary SAP Replication Server. In turn, the primary SAP Replication Server uses the datatype formats specified in the replication definition to receive the data from Replication Agent for Oracle.

You can define a column that is CHAR or VARCHAR2 as char or varchar datatype respectively in the replication definition. If the primary database character set is different from the SAP Replication Server character set, you must define the column as varchar datatype in the replication definition and make sure that the size (in bytes) is large enough to accommodate maximum size of characters in the primary database column.

For example, if the character encoding is UTF-16 for the primary database (Oracle) and UTF-32 for SAP Replication Server, and if a column is CHAR(10) or VARCHAR2(10) in the primary database, define the datatype as varchar(20) in the replication definition.

If you are replicating approximate datatypes (`real` or `float`) in Oracle, define the datatype as `rs_oracle_float` in the replication definition.

If the replication definition is automatically created by Replication Agent, Replication Agent can measure the correct size of the datatype and perform accurate datatype mapping.

Table 2. Recommended Oracle Datatype Mapping

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`BINARY_DOUBLE`	9 bytes, 64-bit single precision floating point number datatype	`double`	8 bytes	• Maximum positive finite value is 1.79769313 486231E +308. • Minimum positive finite value is 2.22507485 850720-308.
`BINARY_FLOAT`	5 bytes, 32-bit single precision floating point number datatype	`rs_oracle_float`	4 or 8 bytes, depending on precision	• Maximum positive finite value is 3.40282E +38F. • Minimum positive finite value is 1.17549E-38 F.
`BLOB`	4GB, variable-length binary large object	`image`	2GB	

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
BOOLEAN	1 byte	rs_oracle_decimal	17 bytes.	The BOOLEAN datatype is only for use with PL/SQL.
CHAR	2000 bytes	char	32K	**Note:** If the primary database character set is different from the Replication Server character set, define CHAR column as varchar datatype in the replication definition.
CLOB	4GB, variable-length character large object	image or unitext	2GB	For Replication Server 15.0 and later versions, the CLOB datatype maps to unitext. For earlier versions of Replication Server, the NCLOB datatype maps to image.

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
DATE	8 bytes, fixed-length	datetime or rs_oracle_datetime	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 9999 AD. **Note:** Use the Replication Server heterogeneous datatype support (HDS) feature for datatype conversion and translation.
INTERVAL DAY(n) TO SECOND(n)	Variable-length	rs_oracle_interval		
INTERVAL YEAR(n) TO MONTH	Variable-length	rs_oracle_interval		
LONG	2GB, variable-length character data	text		
LONG RAW	2GB, variable-length binary data	image		

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
NCHAR	2000 bytes, multibyte characters	unichar	32K	
NCLOB	4GB, variable-length multibyte character large object	unitext or text	2GB	For Replication Server 15.0 and later versions, the NCLOB datatype maps to unitext. For earlier versions of Replication Server, the NCLOB datatype maps to image.

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
NUMBER (p,s)	21 bytes, variable-length numeric data	`float`, `int`, `real`, `number`, `decimal`, or `rs_oracle_decimal`	`float` is 4 or 8 bytes. `int` is 4 bytes. `real` is 4 bytes. `number` and `decimal` are 2 to 17 bytes.	The `float` datatype can convert to scientific notation if the range is exceeded. Integers (`int`) are truncated if they exceed the Replication Server range of 2,147,483,647 to -2,147,483,648 or $1x10^{-130}$ to $9.99x10^{25}$. The `number` and `decimal` datatypes are truncated if they exceed the range of -10^{38} to $10^{38}-1$. Oracle precision ranges from 1 to 38 digits. Default precision is 18 digits. Oracle scale ranges from -84 to 127. Default scale is 0.

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
NVARCHAR2	4000 bytes, variable-length, multibyte character data	univarchar or varchar	32K	
RAW	2000 bytes, variable-length binary data	rs_oracle_binary	32K	
ROWID	10 bytes, binary data representing row addresses	rs_oracle_rowid	32K	
SIMPLE_INTEGER	4 bytes representing signed integers	integer		SIMPLE_INTEGER is new as of Oracle 11g and is only for use with PL/SQL. **Note:** Marking procedures with PLS_INTEGER and predefined PL/SQL numeric datatypes other than SIMPLE_INTEGER is not supported.

Oracle Datatype	Oracle Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
TIMESTAMP(n)	21-31 bytes, variable-length	datetime or rs_oracle_timestamp9	8 bytes	Replication Server supports dates from January 1, 1753 to December 31, 9999. Oracle supports dates from January 1, 4712 BC to December 31, 4712 AD. **Note:** Use the Replication Server heterogeneous datatype support (HDS) feature for datatype conversion and translation.
TIMESTAMP(n) WITH [LOCAL] TIME ZONE	Variable-length	rs_oracle_timestamptz		
UDD object type	Variable-length character data	rs_rs_char_raw	32K	
VARCHAR2	4000 bytes, variable-length character data	varchar	32K	
XMLTYPE	4GB, variable-length character large object	text	2GB	XMLTYPE data is implicitly handled as Oracle CLOB data.

See also

- *Oracle User-Defined Types* on page 40

Replication Server 15.0 Unsigned Datatype Mapping

For Replication Server 15.0 and later, unsigned datatypes are supported and can be specified in the replication definitions.

Table 3. Unsigned Integer Replication Definition Datatype Mapping

RepServer 15.0 Unsigned Datatypes	Replication Definition Datatypes
`unsigned bigint`	`numeric (20)`
`unsigned int`	`numeric (10)`
`unsigned smallint`	`int`
`unsigned tinyint`	`tinyint`

Oracle ANYDATA Datatype Compatibility

Replication Agent supports the replication of data stored in `ANYDATA` columns.

- Both the primary and replicate databases must be Oracle databases.
- Both the primary database table and replicate database tables must have the same `ANYDATA` columns.

The **pdb_ignore_unsupported_anydata** configuration parameter determines how Replication Agent handles data of unsupported datatypes stored in columns of type `ANYDATA`. See the *Replication Agent Reference Manual*.

Oracle XMLTYPE Datatype Compatibility

Replication Agent supports the replication of `XMLTYPE` column data stored `CLOB` or `XML` from an Oracle 10g, 11g, or 12c primary database to an Oracle 10g, 11g, 12c or SAP Adaptive Server Enterprise replicate database.

However, the `XMLTYPE` datatype is supported only for replication to Oracle 10g replicate databases. The `XMLTYPE` stored as `CLOB` is supported for replication to Oracle 11g or 12c replicate databases. The `XMLTYPE` stored as `text` is supproted for replication to SAP Adaptive Server Enterprise replicate databases.

Note: The support for the `XMLTYPE` stored as `CLOB` is deprecated in Oracle 12c version. However, Replication Agent for Oracle supports `XMLTYPE` stored as `CLOB` for existing columns.

See also

- *Oracle 10g, 11g, and 12c XMLTYPE Restrictions* on page 35
- *XMLTYPE Data Replication* on page 67

Oracle Datatype Restrictions

SAP Replication Server and Replication Agent impose some constraints on the Oracle `NUMBER` datatype.

See the *SAP Replication Server Options Release Bulletin* for the latest information on datatype restrictions.

These constraints are:

- In the integer representation:
  - The corresponding SAP `int` datatype has a smaller absolute maximum value.

    The Oracle `NUMBER` absolute maximum value is 38 digits of precision, between 9.9 x 10^{125} and 1 x 10^{-130}. The SAP `int` value is between 2^{31} - 1 and -2^{31} (2,147,483,647 and -2,147,483,648), inclusive.
  - Oracle `NUMBER` values greater than the SAP `int` maximum are rejected by SAP Replication Server.
- In the floating point representation:
  - The precision of the floating point representation has the same range limitation as the integer representation.
  - If the floating point value is outside the SAP range of 2^{31} - 1 and -2^{31} (2,147,483,647 and -2,147,483,648), Replication Agent for Oracle converts the number into exponential format to make it compatible with SAP Replication Server. No loss of precision or scale occurs.

Replication Agent does not support replication of these special values for `BINARY_FLOAT` and `BINARY_DOUBLE` datatypes:

- NaN (not a number)
- Inf (positive infinity)
- -Inf (negative infinity)

SAP Replication Server and Replication Agent impose these constraints on the Oracle `TIMESTAMP WITH [LOCAL] TIME ZONE` datatype.

When a `TIMESTAMP WITH TIME ZONE` datatype is replicated, the time zone information is used to resolve the timestamp value to the “local” time zone and then the resolved value is replicated. The time zone information itself is not replicated.

For example, if a `TIMESTAMP WITH TIME ZONE` datatype is recorded in Oracle as “01-JAN-05 09:00:00.000000 AM -8:00” and the “local” time zone is -6:00, the replicated value is “01-JAN-05 11:00:00.000000”. The timestamp value is adjusted for the difference between the recorded time zone of -8:00 and the local time zone of -6:00, and the adjusted value is replicated.

Oracle ANYDATA Datatype Restrictions

Replication Agent does not support the replication of data stored in `ANYDATA` columns under some circumstances.

Replication of `ANYDATA` is not supported when:

- The replicate database table column is not of type `ANYDATA`. An attempt to replicate data stored in an `ANYDATA` column to a column that is not of the `ANYDATA` type causes the Replication Server Data Server Interface (DSI) thread to fail.
- The size of the data stored in an `ANYDATA` column exceeds the maximum size of the Replication Server `opaque` datatype, which is 16K.
- Replication Agent does not replicate data of these Oracle datatypes or structures stored in a column of type `ANYDATA`:
  - `BFILE`
  - Nested tables
  - `REF`
  - `UROWID`
  - `VARRAY`

The **pdb_ignore_unsupported_anydata** configuration parameter determines how Replication Agent handles data of unsupported datatypes stored in columns of type `ANYDATA`.

See the *Replication Server Reference Manual* for information on replication definitions and **create replication definition**.

See the Oracle SQL Reference guide for a complete list of Oracle-supplied types.

See the *Replication Agent Reference Manual* for information on **pdb_ignore_unsupported_anydata**.

Oracle 10g, 11g, and 12c XMLTYPE Restrictions

Replication Agent supports the replication of `XMLTYPE` columns and tables if they are stored as `CLOB` or `XML` data. Replication Agent does not support the replication of `XMLTYPE` data in object-relational XML storage or binary XML storage.

Replication Agent supports the replication of `XMLTYPE` columns from an Oracle primary database to an Oracle or SAP Adaptive Server Enterprise replicate database, but no other platforms are supported. Replication Agent replicates `XMLTYPE` tables only from an Oracle primary database to an Oracle replicate database.

Note: The `XMLTYPE` datatype is supported only for replication to Oracle 10g replicate databases from Oracle 10g, 11g, and 12c primary databases.

See also

- *Oracle XMLTYPE Datatype Compatibility* on page 33

- *XMLTYPE Data Replication* on page 67

Oracle ROWID Datatype Restrictions

When Replication Agent replicates `ROWID` data, the value replicated always represents the value stored in the table in the primary database and has no relationship to the `ROWID` value in the replicate database. There is no attempt to convert or adjust `ROWID` data to match the data in the replicate database.

Oracle Large Object (LOB) Support

Oracle LOB data can exist in several formats in Oracle.

The LOB datatypes in Oracle are:

- Character:
  - `LONG`
  - `CLOB`
  - `NCLOB`
- Binary:
  - `LONG RAW`
  - `BLOB`
  - `BFILE`–points to file contents stored outside of the Oracle database

For those types stored in the database (all types except `BFILE`), Oracle records the content of the LOB in the redo log. The Replication Agent reads the LOB data from the redo log and submits the data for replication.

Because `BFILE` type data is stored outside of the database, the `BFILE` contents are not recorded in the redo log. To replicate the content of a `BFILE`, the Replication Agent connects to the primary Oracle database and issues a query to select the data from the `BFILE`. Selecting the `BFILE` data separately from other data in the redo log can provide a temporary out-of-sync condition if the `BFILE` contents are changed multiple times.

See also

- *Replication of LOB Columns* on page 36

Replication of LOB Columns

Oracle logs all LOB data (except for `BFILE` datatypes) in the Oracle redo log. This allows the Replication Agent to apply each individual LOB change. However, for `BFILE` data, the same technique is used and the same limitation exists—`BFILE` data is not logged but read from the database at the time the rest of the transaction is processed.

For instructions on enabling and disabling replication for LOB columns, see the *Replication Agent Administration Guide*.

Transaction Integrity and LOB Data

Because of the way Replication Agent processes the LOB column data when replicating transactions, transaction integrity may be compromised. For example, if two transactions change the data in a LOB column and the Log Reader does not process the first transaction until after the second transaction has been committed, when the LOB data is read from the primary database, the value of that data is the result of the second transaction. In this event, the value of the LOB data in the first transaction is never sent to the replicate database. After the second transaction is processed by the Log Reader, the primary and replicate databases are synchronized again, but for a period of time between processing the first and second transactions, the replicate database contains data that does not match the originating transaction.

This problem occurs only when a LOB column is changed more than once by a sequence of transactions. The period of time over which the problem exists may be significant if the replication system throughput is slow or if a replication system component fails. As soon as the last transaction that changes the LOB column is processed at the replicate site, the problem is corrected.

Large Object Replication Limitation

Replication Agent does not support the replication of partial updates to LOB columns.

For example, use of the Oracle **DBMS_LOB.WRITE()** function, which updates LOB data from a specified offset, is not replicated.

Special Handling for Off-Row Large Objects

Learn how Replication Agent handles off-row stored LOBs.

LOB types that are stored within the Oracle database (`BLOB`, `CLOB`, and `NCLOB`) may be defined with certain storage characteristics. One of those characteristics, “disable storage in row,” indicates that the data for the LOB should always be recorded separately from the rest of the data in the row the LOB belongs to. This off-row storage requires special handling for replication of updates to these LOB values.

When an off-row LOB value is updated, the change recorded in the redo log is for the index that holds the LOB data; the row the LOB belongs to is not changed. As a result, information is missing from the redo log to identify which row in the table the LOB belongs to.

For example, when a non-LOB column is updated in a table, the column data that identifies the changed values and lookup columns is recorded. The command `updated myTable set col2 = 2 where col1 = 1` records values in the redo log for the values of both “col2” and “col1.”

In contrast, a command that only updates a LOB that has been defined with the **disable storage in row** clause records only the LOB data change to its index, and not the table that holds the LOB. So the command `updated myTable set ClobColumn = 'more`

`data' where col1 = 1` only records the value changed, and does not include the value of “col1”.

Because the value of the columns in the **where** clause are not logged in that update, there is insufficient information to build the correct **where** clause to be used to apply the data at the replicate site. To resolve this problem, Replication Agent for Oracle requires that an update to a LOB column defined with **disable storage in row** must be immediately accompanied by an **insert** or **update** to the same row in the table the LOB belongs to.

The Replication Agent uses the additional column data from the associated operation to correctly build the **where** clause required to support replication.

For example, these transaction sequences support replication of updates to LOB column “ClobColumn” when it has been defined with the **disable storage in row** clause:

```
begin
insert into myTable (col1, col2, ClobColumn, updated)
values (1,1,empty_clob(), sysdate);
update myTable set ClobColumn = 'more data' where col1 = 1;
commit

begin
update myTable set updated = sysdate() where col1 = 1;
update myTable set ClobColumn = 'more data' where col1 = 1;
commit

begin
update myTable set ClobColumn = 'more data' where col1 = 1;
update myTable set updated = sysdate() where col1 = 1;
commit
```

Note: For purposes of replication, LOB objects populated with the **empty_clob** or **empty_lob** function are replicated as NULL values. Replication definitions for LOB columns should therefore include the “null” keyword as part of the column definition.

These transaction sequences are not supported for LOB columns defined with the **disable storage in row** clause and result in a failure to supply the LOB data to the replicate site:

- Missing accompanying change to the same row:

  ```
  begin
  update myTable set ClobColumn = 'more data' where col1 = 1;
  commit
  ```

- Accompanying change for the same row is not immediately adjacent to the LOB change:

  ```
  begin
  update myTable set updated = sysdate where col1 = 1;
  update myTable set col2 = 5 where col1 = 5;
  update myTable set ClobColumn = 'more data' where col1 = 1;
  commit
  ```

This limitation applies only to LOB columns that have been defined with the **disable storage in row** clause.

You can identify the LOB columns in your database that have this constraint using this query against your Oracle database:

```
select owner, table_name, column_name from dba_lobs
where in_row = 'NO';
```

Replicating CLOB and NCLOB Datatypes

Oracle `NCLOB` (National Character Large Object) is a datatype that stores large character data using a multibyte national character set. Similarly, the `CLOB` datatype may also store character data using a multibyte national character set, when the Oracle database is defined with a double-byte or variable-width character set.

By default, the byte order of the multibyte characters stored in the `NCLOB` datatype (and `CLOB` when the database is defined with a double-byte or variable-width character set) is converted during replication to big-endian byte order. This allows the data to be transmitted over networks using big-endian order, which is the common network byte order.

The datatype in a replication definition for an `NCLOB` or `CLOB` should be `unitext`. This prevents Replication Server from attempting character set conversion on the data. If the Replication Server version does not support `unitext`, use the `image` datatype.

If the target database that is to receive this `NCLOB` or `CLOB` data is installed on a little-endian platform, the database may not automatically convert the replicated data from the sent big-endian order to the little-endian order. To support replicating `NCLOB` or `CLOB` data to a database server that does not provide the necessary conversion from big-endian (network order) to little-endian, force the byte order to be sent by the Replication Agent using the **lr_ntext_byte_order** parameter to set a value of **big** (for big-endian) or **little** (for little-endian).

The **lr_ntext_byte_order** parameter is available for Microsoft SQL Server and Oracle, andis important for replication between two databases that reside on different platforms. For example, for replication between Oracle and Microsoft SQL Server, the primary database stores the data in big-endian byte order, but the replicate database stores data in little-endian byte order because Microsoft SQL Server only runs on Windows. Therefore, set the **lr_ntext_byte_order** parameter to **little** to force the Replication Agent to convert the data to little-endian (the format expected by SQL Server). However, if the replicate database is not a Microsoft SQL Server, determine its byte order and set the **lr_ntext_byte_order** parameter accordingly.

Note: The default behavior of Replication Agent for Oracle is to force any Unicode data to big-endian order as defined by the **ltl_big_endian_unitext** configuration parameter. To allow the **lr_ntext_byte_order** configuration parameter to successfully override the Oracle byte order, you must also set the **ltl_big_endian_unitext** configuration parameter to false whenever the **lr_ntext_byte_order** parameter is used.

The **ltl_big_endian_unitext** parameter specifies whether `unitext` data should be converted from little-endian to big-endian before sending LTL to the Replication Server. Valid values are true and false. When setting this parameter, you must know how **lr_ntext_byte_order** is set. If

lr_ntext_byte_order is set to send the correct byte order for the replicate database, the **ltl_big_endian_unitext** parameter must be set to false so that the byte order is not changed. **ltl_big_endian_unitext** is true, by default. The **ltl_big_endian_unitext** and **lr_ntext_byte_order** configuration parameters have differences:

- When **ltl_big_endian_unitext** is true, Replication Agent for Oracle sends all Unicode data in big-endian order.
- When **ltl_big_endian_unitext** is false, Replication Agent for Oracle allows Unicode data to be sent in a byte order that is used when the data is stored in the transaction log file.

lr_ntext_byte_order forces the result of Unicode data that is read from the transaction log to be in the correct byte order, regardless of how it normally exists in the transaction log file.

Oracle User-Defined Types

User-defined datatypes (UDDs) use Oracle built-in datatypes and other user-defined datatypes as building blocks that model the structure and behavior of data in applications.

Replication Agent for Oracle supports replication of user-defined object types. Object types are abstractions of real-world entities, such as purchase orders, that application programs deal with. An object type is a schema object with three kinds of components:

- A name, which identifies the object type uniquely within that schema.
- Attributes, which are built-in types or other user-defined types. Attributes model the structure of the object.
- Methods, which are functions or procedures written in PL/SQL and stored in the database, or written in a language such as C or Java and stored externally. Methods implement operations the application can perform on the object.

Creating a Datatype Definition in Replication Server

Create a definition for your user-defined datatype.

Prerequisites

You must have Replication Server administrator privileges or permission. Also, if you are using Replication Server 15.1 or earlier, see "Replication Server and RSSD Scripts" first.

Task

To replicate user-defined datatypes in Oracle, the datatype specified in the replication definition must be `rs_char_raw`.

1. Log in to the RSSD.
2. Add a row to the **rs_datatype** table using this example as a guide:

```
/*  rs_oracle_udd_raw - char with no delimiters */
insert into rs_datatype values(
0,                  /* prsid */
0x0000000001000008,  /* classid */
```

```
'rs_oracle_udd',      /* name */
0x0000000000010210,   /* dtid */
0,                    /* base_coltype */
255,                  /* length */
0,                    /* status */
1,                    /* length_err_act */
'CHAR',               /* mask */
0,                    /* scale */
0,                    /* default_len */
'',                   /* default_val */
0,                    /*-delim_pre_len-*/
'',                   /* delim_pre */
0,                    /*-delim_post_len-*/
'',                   /* delim_post */
0,                    /* min_boundary_len */
'',                   /* min_boundary */
3,                    /* min_boundary_err_act */
0,                    /* max_boundary_len */
'',                   /* max_boundary_err_act */
0                     /* rowtype */
)
go
```

3. Restart Replication Server.
4. In Replication Server, test the new type:

```
admin translate, 'The quick brown fox jumped over the lazy
dog.', 'char(255)', 'rs_oracle_udd'
```

```
go
```

```
Delimiter Prefix   Translated                  Value Delimiter
Postfix
```

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

```
NULL               The quick brown fox jumped over the lazy dog.
NULL
```

The new type is defined correctly if the sentence translates correctly.

See also

- *Replication Server and RSSD Scripts* on page 24

Example: Create a Replication Definition

This example demonstrates how to create a replication definition using the `rs_char_raw` type defined in Replication Server.

These Oracle table and type definitions are used in the example:

- Oracle UDD object type name: NAME_T
- Oracle table name: USE_NAME_T

- Oracle table columns: PKEY INT, PNAME NAME_T

```
create replication definition use_name_t_repdef
with primary at ra_source_db.ra_source_ds
with all tables named 'USE_NAME_T'
(
   PKEY int,
   PNAME rs_rs_char_raw
)
primary key (PKEY)
searchable columns (PKEY)
go
```

Note: For this example, **ltl_character_case** must be **upper**.

Object Type Attribute Replication

To replicate updates to user-defined object type attributes, Replication Agent must enable table-level supplemental logging. Table-level supplemental logging can be enabled manually.

Replication Agent also attempts to enable this logging when marking a table that contains a user-defined object type. However, for Replication Agent to mark such a table, there must already be an Oracle user specified by the **pds_username** parameter that has ALTER permission granted for the table.

If table-level supplemental logging has not been enabled for a table containing a user-defined object type and Replication Agent encounters an update log record in the Oracle log, Replication Agent changes its status from Replicating to Admin with this error:

```
There is insufficient column data in the log to support Oracle UDD
update command processing. Please make sure table-level supplemental
logging is enabled.
```

In this case, use the **pdb_skip_op** to skip this log record. See the *Replication Agent Reference Manual*.

Sequence Marking and Unmarking

Support for Oracle sequence replication is supported only for replication to Oracle. No support is provided for replicating a sequence value to a non-Oracle replicate database.

Replication Agent supports replication of sequences in the primary database. To replicate a sequence invoked in a primary database, the sequence must be marked for replication, and replication must be enabled for that sequence. This is analogous to marking and enabling replication for tables.

Note: Marking a sequence for replication is separate from enabling replication for the sequence. If the value of the **pdb_dflt_object_repl** parameter is true, replication is enabled automatically at the time a sequence is marked.

Oracle does not log information every time a sequence is incremented. Sequence replication occurs when the Replication Agent captures the system table updates that occur when the sequence's cache is refreshed. Therefore, the sequence value replicated when a sequence is

marked for replication is the “next” sequence value to be used when the current cache expires. The result is that not every individual increment of a sequence is replicated, but the replicate site always has a value greater than the primary site's currently available cached values.

To temporarily suspend replication of a marked sequence, you can disable replication for the sequence.

See also

- *Sequence Replication Enabling and Disabling* on page 45

Replication Server Changes to Support Sequence Replication

By default, Replication Server does not support replication of Oracle sequence objects. You must make changes to Replication Server and the replicate Oracle database before you can replicate Oracle sequences.

For Replication Server, you must create a replication definition that defines a stored procedure to assist with sequence replication. Execute the `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_rs_sequence_repdef.sql` script against your primary Replication Server after editing the script to replace values *{pds}* and *{pdb}* with the name of your primary Replication Server connection. You can find these values in the **rs_source_ds** and **rs_source_db** Replication Agent configuration properties.

Note: The replication definition assumes that a database replication definition exists. You may need to alter the definition if a database replication definition does not exist. For details, see comments in the `oracle_create_rs_sequence_repdef.sql` script.

In the replicate Oracle database, you must create a stored procedure to support sequence replication. Log in to the replicate Oracle database as the maintenance user defined in your Replication Server connection to the replicate database. Execute the `$SYBASE/RAX-15_5/scripts/oracle/oracle_create_replicate_sequence_proc.sql` script to create the necessary stored procedure.

Note: The maintenance user defined in the Replication Server connection to the replicate database must have sufficient privileges to execute functions in the Oracle DBMS_SQL package. Also, this maintenance user must have authority at the replicate Oracle database to update any replicated sequence.

Marking a Sequence for Replication

Mark a sequence for replication.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine whether or not the sequence is marked in the primary database:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence you want to mark for replication.

- If **pdb_setrepseq** returns information that the specified sequence is marked, you do not need to continue this procedure.
- If **pdb_setrepseq** returns information that the specified sequence is not marked, continue this procedure to mark the sequence for replication.

3. Mark the sequence for replication.

   pdb_setrepseq allows you to mark the primary sequence to be replicated and specify a different sequence name to use in the replicate database.

   - If the sequence name you want to increment at the replicate site has the same name as at the primary site, use this command to mark the sequence for replication:

     ```
     pdb_setrepseq pdb_seq, mark
     ```

     Note: Replicating a sequence with a different name than is provided is consistent with other marking commands but is not a typical configuration.

   - To mark the sequence for replication using a different sequence name, use:

     ```
     pdb_setrepseq pdb_seq, rep_seq, mark
     ```

     where *rep_seq* is the name of the sequence that you want to increment in the replicate database.

     Note: Replicating sequence values to a sequence with a different name at the replicate site assumes that the replicate site sequence has the same attributes and starting value as the primary site's sequence.

     - If the value of **pdb_dflt_object_repl** is true, the sequence marked for replication with **pdb_setrepseq** is ready for replication after you successfully invoke **pdb_setrepseq**.
     - If the value of **pdb_dflt_object_repl** is true (the default value), skip step 4.
     - If the value of **pdb_dflt_object_repl** is false, you must enable replication for the sequence before replication can take place.

4. Enable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, enable
   ```

After replication is enabled for the sequence, you can begin replicating invocations of that sequence in the primary database.

Note: To replicate a sequence, you must also run the `oracle_create_replicate_sequence_proc.sql` script in the `$SYBASE/RAX-15_5/scripts/oracle` directory at the replicate site to create a procedure named **rs_update_sequence**.

Unmarking a Sequence

Unmark a sequence.

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

2. Determine whether or not the sequence is marked in the primary database:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence you want to unmark.

   - If **pdb_setrepseq** returns information that the specified sequence is marked, continue this procedure to unmark the sequence.
   - If **pdb_setrepseq** does not return information that the specified sequence is marked, you do not need to continue this procedure.
3. Disable replication for the sequence:

   ```
   pdb_setrepseq pdb_seq, disable
   ```
4. Remove the replication marking from the sequence:

   ```
   pdb_setrepseq pdb_seq, unmark
   ```

   To force the unmarking, use:

   ```
   pdb_setrepseq pdb_seq, unmark, force
   ```
5. Confirm that the sequence is no longer marked for replication:

   ```
   pdb_setrepseq pdb_seq
   ```

Sequence Replication Enabling and Disabling

To temporarily suspend replication of a sequence, use **pdb_setrepseq** to disable replication for the marked sequence. When you are ready to resume replication of the marked sequence, use **pdb_setrepseq** again to enable replication.

Note: By default, no sequences are marked for replication.

To replicate updates of a sequence in the primary database, the sequence must be marked for replication and, replication must be enabled for that sequence.

Marking a sequence for replication is separate from enabling replication for the sequence.

See also

- *Marking a Sequence for Replication* on page 43

Enabling Replication for a Marked Sequence

Enable replication for a marked sequence.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine whether or not replication is enabled for the sequence:

   ```
   pdb_setrepseq pdb_seq
   ```

   where *pdb_seq* is the name of the sequence for which you want to enable replication.

   If **pdb_setrepseq** returns information that the specified sequence is marked and has replication disabled, continue this procedure to enable replication for the sequence.

Note: A sequence must be marked for replication before replication can be enabled or disabled for the sequence.

3. Enable replication for the sequence:

```
pdb_setrepseq pdb_seq, enable
```

   After replication is enabled for the sequence, any invocation of that sequence is replicated.

4. Use **pdb_setrepseq** again to verify that replication is now enabled for the sequence:

```
pdb_setrepseq pdb_seq
```

Disabling Replication for a Marked Sequence

Disable replication for a marked sequence.

1. Log in to the Replication Agent instance with the administrator login.
2. Determine whether or not replication is enabled for the sequence:

```
pdb_setrepseq pdb_seq
```

   where *pdb_seq* is the name of the sequence for which you want to disable replication.

   If **pdb_setrepseq** returns information that the specified sequence is marked and has replication enabled, continue this procedure to disable replication for the sequence.

   Note: A sequence must be marked for replication before replication can be enabled or disabled for the sequence.

3. Disable replication for the sequence:

```
pdb_setrepseq pdb_seq, disable
```

   After replication is disabled for the sequence, any invocation of that sequence is not captured for replication until replication is reenabled.

4. Use **pdb_setrepseq** again to verify that replication is now disabled for the sequence:

```
pdb_setrepseq pdb_seq
```

Setting Up Replication Agent and Oracle on Different Machines

Run Replication Agent and the primary data server on different machines.

1. Install Replication Agent on a machine of the same hardware and operating system as the machine on which the primary data server is running.
2. Install the JDBC driver on the same machine as Replication Agent.
3. If the `timezone.dat` file is not accessible to both machines, copy the `$ORACLE_HOME/oracle/timezone.dat` file to the Replication Agent machine.

   Make a copy of the timezone file available on the machine hosting the Replication Agent. To determine which timezone file is used by the primary Oracle instance, connect to the Oracle instance and execute:

```
select * from V$TIMEZONE_FILE
```

4. Set the Replication Agent **pdb_timezone_file** configuration parameter to the full path name of the `timezone.dat` file.
5. If Replication Agent for Oracle is configured to truncate Oracle archive logs directly, make sure the Oracle archive logs are accessible to both machines. Use the **ra_devicepath** command to point Replication Agent to the log files.

Real Application Clusters (RAC)

Replication Agent for Oracle provides support for Oracle 10g and 11g RAC environments. When a Replication Agent for Oracle instance is initialized, the Oracle database is queried to determine how many nodes are supported by the cluster. Based on this information, Replication Agent automatically configures itself to process the redo log information from all nodes.

To process the redo log data from all nodes in an Oracle RAC cluster, the Replication Agent must execute from a location that has access to the same shared storage used by the Oracle nodes to store their redo data.

Configure Replication Agent to connect to a single Oracle instance by supplying the required host, port, and Oracle SID values to the **pds_host_name**, **pds_port_number** and **pds_database_name** configuration parameters. However, in an Oracle RAC environment, Replication Agent must be able to connect to any node in the cluster in the event that a node fails or otherwise becomes unavailable. To support the configuration of multiple node locations, Replication Agent supports connectivity to all possible RAC nodes by obtaining needed information from an Oracle `tnsnames.ora` file for one specified entry. As a result, instead of configuring individual host, port and instance names for all nodes, Replication Agent only requires the location of a `tnsnames.ora` file and the name of the TNS connection to use.

It is recommended that you point Replication Agent to a `tnsnames.ora` entry that contains the address for all nodes in the cluster.

For example, if this entry exists in a `tnsnames.ora` file for a three-node cluster, instruct Replication Agent to use that entry by providing the `tnsnames.ora` file location to the **pds_tns_filename** configuration property and specifying **RAC10G** as the value for the **pds_tns_connection** configuration property:

```
RAC10G =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOAD_BALANCE = yes)
      (FAILOVER = ON)
      (ADDRESS = (PROTOCOL = TCP)(HOST = www.xxx.yyy.zz1)
      (PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = www.xxx.yyy.zz2)
      (PORT = 1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST = www.xxx.yyy.zz3)
      (PORT = 1521))
```

```
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = rac10g)
    )
  )
```

The `tnsnames.ora` file must also contain a connect descriptor for each node in the cluster:

```
NODE1-VIP =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = www.xxx.yyy.zz1)
    (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = rac10g)
      (INSTANCE_NAME = node1-vip)
    )
  )
NODE2-VIP =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = www.xxx.yyy.zz2)
    (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = rac10g)
      (INSTANCE_NAME = node2-vip)
    )
  )
NODE3-VIP =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = www.xxx.yyy.zz3)
    (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = rac10g)
      (INSTANCE_NAME = node3-vip)
    )
  )
```

If the `tnsnames.ora` file does not contain a connect descriptor for each node, Replication Agent cannot generate commit records correctly.

See the *Replication Agent Reference Manual* for details about **pds_tns_filename** and **pds_tns_connection**.

Note: Replication Agent must have read access to the `tnsnames.ora` file.

pdb_archive_path

The **pdb_archive_path** configuration parameter identifies where Replication Agent expects to find archived Oracle redo log files.

If Replication Agent for Oracle is configured to truncate archive redo logs without using the Oracle Recovery Manager (RMAN), make sure Replication Agent has permission to truncate

the Oracle archive redo logs. Oracle can be configured to archive to different locations and Oracle finds these locations using the destination column of the V$ARCHIVE_DEST system table. Replication Agent must have read access to the archive redo logs and the directories containing them.

Note: Archived redo logs can also be stored within ASM.

You can configure Replication Agent to remove archived logs from the location specified by **pdb_archive_path**, using the **pdb_archive_remove** configuration parameter. This allows Replication Agent to remove archived log files that are no longer needed to support replication. If **pdb_archive_remove** is set to true, Replication Agent must have update authority to the archive log directory and delete authority on the individual archive log files. If **rman_enabled** or the **pdb_archive_path** configuration parameter is pointed to an ASM path, then update authority to the archive log directory and delete authority on the individual archive logs are not required.

Note: The **rman_enabled** parameter enables Replication Agent to use the Oracle **RMAN** utility to truncate old archive log files. See the *Replication Agent Reference Manual*.

See also

- *Automatic Storage Management* on page 49
- *Marking a Sequence for Replication* on page 43

Oracle Instance Failover

If the Oracle instance to which Replication Agent is connected fails for any reason, Replication Agent attempts to reconnect to any surviving instance, choosing from the list of instances defined in the `tnsnames.ora` file entry.

No manual intervention or configuration is required. If none of the instances are available, Replication Agent reports an error and continues processing as long as redo log file information is still available.

Automatic Storage Management

Replication Agent for Oracle supports the use of the Oracle Automatic Storage Management (ASM) feature. ASM provides file system and volume management support for an Oracle database environment. ASM can be used in both Real Application Cluster (RAC) and non-RAC environments.

Archive Log Removal and Configuration

Archive logs that are managed by ASM can be removed from ASM when they are no longer needed by Replication Agent for Oracle.

When **pdb_archive_remove** is set to true and the archive logs are managed by ASM, **pdb_archive_path** must be set to the name of the ASM disk group in which the archive logs are stored. The disk group name must be preceded with a plus sign (+) indicating that the path is an ASM path. For example:

```
pdb_archive_remove=true
pdb_archive_path=+DISK_GROUP1
```

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, **pdb_archive_path** must be set 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
```

In addition to automatic removal of archive logs from ASM, manual removal is supported with **pdb_truncate_xlog**. **pdb_archive_path** must be set to the ASM disk group name and preceded with a plus (+) sign for archive logs to be manually removed.

Note: rman_enabled enables Replication Agent to use the Oracle **RMAN** utility to truncate old archive log files. See the *Replication Agent Reference Manual*.

Configuration Parameters

Some configuration parameters must be set if your log files are being managed by ASM.

These configuration parameters are:

- **asm_tns_filename** – The fully-qualified file name identifying the Oracle `tnsnames.ora` file that contains the Oracle ASM connection parameters. This configuration parameter is required only when the connection parameter information required for ASM does not exist in the `tnsnames.ora` file pointed to by the **pds_tns_filename** configuration parameter.
- **asm_tns_connection** – The Oracle connection name that identifies the connection parameters for the Oracle Automatic Storage Management (ASM) connection in the Oracle `tnsnames.ora` file. If configuration parameter **asm_tns_filename** is not configured, the `tnsnames.ora` file identified by **pds_tns_filename** is used.
- **asm_username** – The login name that Replication Agent uses to access the Oracle ASM server. The ASM user ID for **asm_username** must have **sysdba** permission. For Oracle 10g, 11g or 12c, set **asm_username** as follows:

  ```
  asm_username="sys as sysdba"
  ```

  Alternately, for Oracle 11g, you can set **asm_username** as follows:

  ```
  asm_username="sys as sysasm"
  ```
- **asm_password** – The password associated with the configuration parameter **asm_username** user to access the Oracle Automatic Storage Management (ASM) server instance.

For example:

```
asm_tns_filename=/u01/app/11.2.0/grid/network/admin/tnsnames.ora
asm_tns_connection=+ASM1
asm_username=sys as sysasm
asm_password=Sybase123
```

See the *Replication Agent Reference Manual*.

Replication Server set autocorrection Command

The Replication Server **set autocorrection** command prevents failures that would otherwise be caused by missing or duplicate rows in a replicated table.

The **set autocorrection** command corrects discrepancies that may occur during materialization by converting each **update** or **insert** operation into a **delete** followed by an **insert**.

To set autocorrection from:

- Replication Agent for one or all tables in the primary database, use the Replication Agent **ra_set_autocorrection** command as described in the *Replication Agent Reference Manual*.
- Replication Server, use the **set autocorrection** command in a replication definition. You must do this from Replication Server, however, because Replication Agent cannot alter the autocorrection setting on a replication definition.

Replication Agent for Oracle does not support use of the autocorrection feature for large-object (LOB), `LONG`, `LONG RAW`, or user-defined datatypes. Also, **pds_username** must have the **ALTER ANY TABLE** privilege to execute these commands:

- `ALTER TABLE tablename ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`
- `ALTER TABLE tablename DROP SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`

Partitioned Tables

Replication Agent supports Oracle partitioning functionality.

Partitioning allows a table, index, or index-organized table to be subdivided into smaller pieces, where each piece of such a database object is called a partition. Each partition has its own name and may optionally have its own storage characteristics. Any table can be partitioned into many separate partitions except those tables containing columns with `LONG` or `LONG RAW` datatypes.

Unstructured data (such as images and documents) stored in a LOB column in the database can also be partitioned. When a table is partitioned, all the columns reside in the tablespace for that partition, with the exception of LOB columns, which can be stored in their own tablespace. For additional information about Oracle partitioning, see the Oracle Database VLDB and Partitioning Guide at *http://download.oracle.com/docs/cd/B28359_01/server.111/b32024/toc.htm*.

Replication of the truncate partition Command

Replication Agent supports replication of the **truncate partition** command.

Replicate the **truncate partition** command either by:

- Using **lr_send_trunc_partition_ddl**
- Wrapping **truncate partition** in a stored procedure and replicating the procedure

*Use **lr_send_trunc_partition_ddl***

Use the Replication Agent configuration parameter **lr_send_trunc_partition_ddl** to determine whether **truncate partition** commands are sent as DDL or DML to the replicate database. The configuration can be:

- true (default) – the truncate partition command is sent as a DDL command (**alter table**). Use this setting to replicate to Oracle.
- false – the truncate partition is sent as a DML operation. Use this setting when replicating to databases that treat **truncate partition** commands as DML (for example, Microsoft SQL Server).

For information about Replication Agent configuration properties, see the *Replication Agent Reference Manual*.

*Wrap the **truncate partition** command*

You can wrap the **truncate partition** command in a stored procedure definition and replicate the procedure.

For example, to replicate **truncate partition** commands from an Oracle primary to an Adaptive Server Enterprise replicate, create this stored procedure at the primary database:

```
create procedure sp_truncate_partition
as
begin
execute immediate 'ALTER TABLE myTable TRUNCATE PARTITION part1';
end;
```

Create a corresponding stored procedure at the replicate database:

```
create proc sp_truncate_partition as
truncate table myTable part1
```

Mark the **sp_truncate_partition** procedure for replication. When **sp_truncate_partition** is executed at the primary database, the **truncate partition** command is replicated to the replicate database.

Materialized Views

A materialized view is a stored view query result.

The data on which the view is defined is referred to as the master table (or tables). The materialized view is stored in its own table, which is refreshed based on changes to the master table. A materialized view may be local, in which it is defined on the same database as the master table, or remote, in which the materialized view is defined on a different database than the master table.

Oracle supports these types of materialized views:

- Read-only – materialized view content is derived from the corresponding master table or tables, and the view content cannot be changed.
- Writeable – materialized view content can be changed temporarily, but any changes are overwritten when the table containing the materialized view is refreshed based on changes to the corresponding master table or tables.
- Updatable – updates made to a materialized view are written back to the corresponding master table or tables when the materialized view is refreshed.

For a complete description of materialized views, see the Oracle documentation.

Replication and Materialized Views

An Oracle materialized view allocates space to hold the result set of its base query. Replication Agent can replicate transactions involving the data on which a materialized view is defined as well as on the materialized view itself.

Materialized View DDL

By default, Replication Agent does not replicate Oracle DDL commands used for materialized views, for example, **CREATE MATERIALIZED VIEW**, **ALTER MATERIALIZED VIEW**, or **DROP MATERIALIZED VIEW**. Materialized view DDL commands are disabled from replication unless otherwise specified using the **pbd_setrepddl** command. See the *Replication Agent Reference Manual > Command Reference > pdb_setrepddl*.

Materialized Views at the Primary and Replicate Databases

A materialized view may exist on both the primary database and the replicate database. Such a situation might arise, for example, if materialized view DDL has been enabled for replication with the **pdb_setrepddl** command or if the replicate database has been materialized from a primary database dump.

If the master table on which the materialized view is defined exists in the primary database, Replication Agent replicates this master table. The materialized view at the replicate database refreshes according to the contents of the replicated master table. Under no circumstances does Replication Agent replicate the table in which a materialized view is stored in the primary database, and you should not attempt to replicate such a table.

If the materialized view is remote—meaning that the master table on which the materialized view is defined does not exist in the primary database—the materialized view at the replicate database must be redirected so that it points to the database on which the master table is located. If the replicate database is not redirected, a refresh of the materialized view fails at the replicate database. In redirecting the replicate database, re-create the Oracle database link that the replicate database uses to connect to the database containing the master table.

Writeable and Updatable Materialized Views

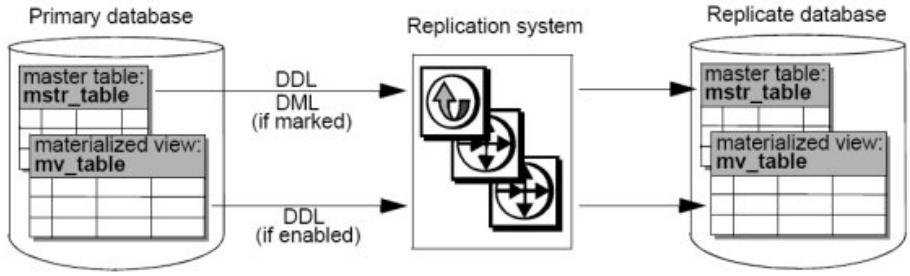
Instead of replicating changes to the table containing a materialized view, Replication Agent replicates changes to the master table if the master table has been marked for replication. Replication Agent therefore does not replicate changes made to a writeable materialized view. However, because changes made to an updatable materialized view are written back to the

corresponding master table or tables when the materialized view is refreshed, Replication Agent replicates changes made to an updatable materialized view on the primary database to the corresponding master table on the replicate database. Changes made to an updatable materialized view on the replicate database affect only the local master table unless bidirectional replication has been enabled.

Materialized View Replication Scenarios

In this figure, a materialized view and a corresponding master table reside on both the primary database and the replicate database.

Figure 2: Master Table and Materialized View on Primary Database



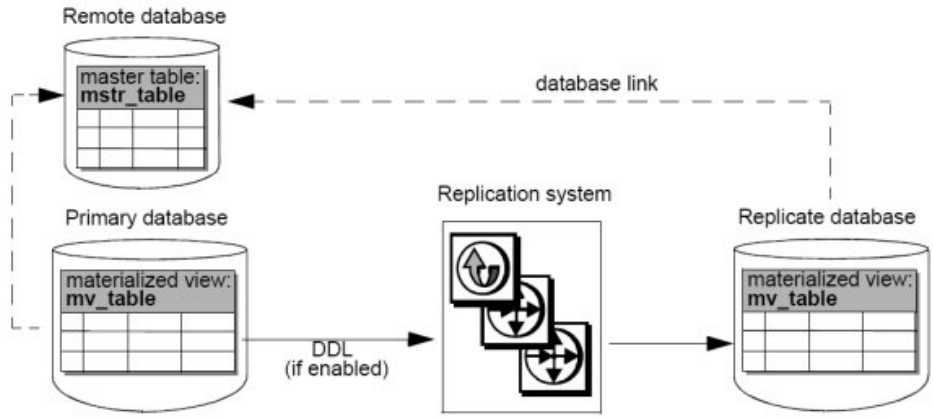
In this situation, DDL commands affecting the master table can be replicated as well as objects affected by the DML that are marked for replication.

DDL commands affecting the materialized view are not replicated unless such DDL is enabled with the **pdb_setrepddl** command. Since a materialized view also exists on the replicate database, all master tables on which the materialized view is defined must also be replicated. Otherwise, the contents of the materialized view on the replicate database may become invalid.

If the materialized view on the primary database is updatable, changes made to this view are written back to the corresponding master table and, if the master table has been marked for replication, replicated to the replicate database. If the materialized view on the replicate database is updatable, changes made to this view are written back to the corresponding master table on the replicate database, but the master table on the primary database is not changed accordingly unless bidirectional replication has been enabled.

In this figure, the master table on which the primary database materialized view is defined resides on a different, or remote, database.

Figure 3: Master Table on Remote Database, Materialized View on Primary Database

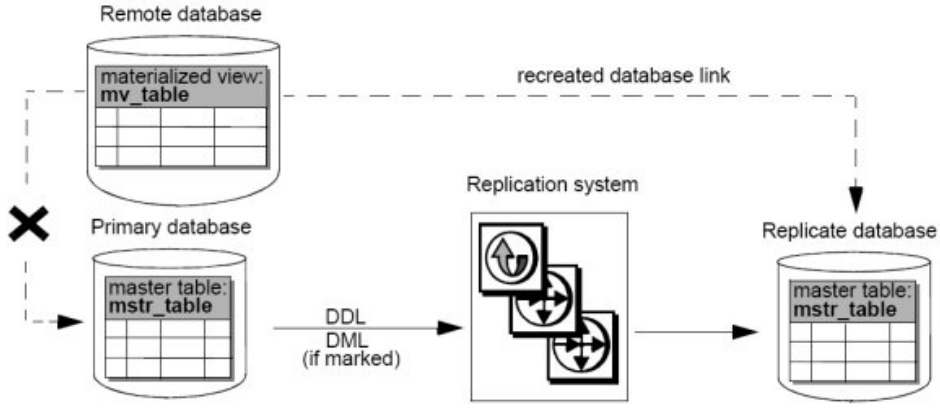


In this situation, neither DML nor DDL affecting the master table is replicated. DDL commands affecting the materialized view are not replicated unless such DDL is enabled with the **pdb_setrepddl** command. Since the materialized view also exists on the replicate database, a database link must be created so that it points to the database containing the master table on which the materialized view is defined.

If the materialized views on the primary and replicate databases are both updatable and are properly linked to the master table at the remote database, changes made to one of these views are written back to the master table, and the changes are reflected in both materialized views upon refresh.

In the figure below, a materialized view resides on a remote database, and the master table on which the materialized view is defined resides on the primary database. A copy of this master table also resides on the replicate database, and the database link between the remote and primary databases is subsequently broken.

Figure 4: Master Table on Primary Database, Materialized View on Remote Database

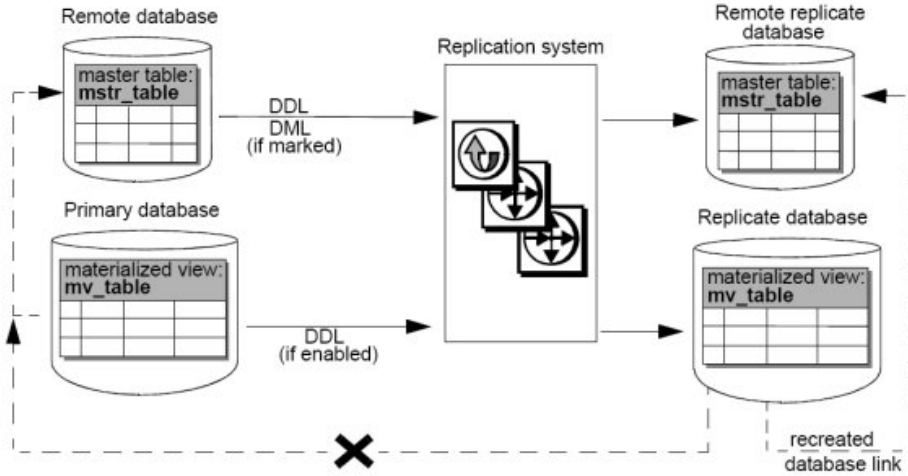


In this situation, DDL commands affecting the master table at the primary database can be replicated as well as DML commands that are marked for replication. DDL commands affecting the materialized view cannot be replicated because there is no corresponding materialized view on the replicate database. If the database link between the remote and primary databases is broken, the remote database must create a link to the replicate database before a refresh occurs.

If the materialized view on the remote database is updatable, changes made to this view are written back to the master table on the database to which the remote database is currently linked.

In the figure below, a master table resides on two different remote databases, one of which is a replicate database. A materialized view resides on the primary database and the replicate database. The materialized view at the replicate database is initially defined by the master table on the remote database, but its database link becomes broken, and the replicate database re-creates a link to the remote replicate database instead.

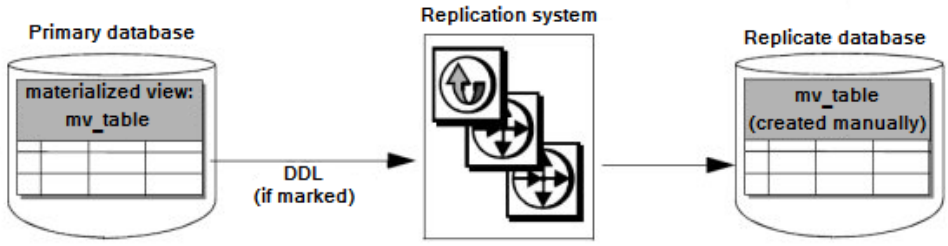
Figure 5: Master Tables on Remote Databases, Materialized Views on Primary and Replicate Databases



If the materialized view on the replicate database is updatable, changes made to this view are written back to the master table on the database to which the replicate database is currently linked. Before the database link between the replicate database and the remote database becomes broken, updates to the materialized view on the replicate database are written back only to the master table on the remote database. After a link is created between the replicate database and the remote replicate database, updates to the materialized view on the replicate database are written back only to the master table on the remote replicate database.

In this figure, the replicate database does not support materialized view. In this situation, you can create a table on the replicate database that has the same structure as the materialized view on the primary database. When you mark the materialized view on the primary database, Replication Agent automatically marks the table containing the materialized view for replication. After the table is marked, any subsequent operations that affect the data in that table are replicated.

Figure 6: Materialized View not Supported on Replicate Database



Index-Organized Tables

Replication Agent can replicate DML for Oracle index-organized tables (IOTs).

DML for these types of index-organized tables can be replicated for Oracle 10g, 11g, and 12c databases:

- Simple IOTs
- IOTs with **including** and **overflow** clauses
- IOTs with composite partitions
- IOTs with mapping tables
- Index-compressed IOTs
- IOTs with row dependencies
- IOTs with large objects
- IOTs with secondary indexes

Replication Agent cannot replicate IOTs with nested tables and columns of type `VARRAY`.

Replicate Database Trigger Execution Control

Trigger execution in a replication system can cause problems when both the transaction that caused a trigger to fire and the transactions that resulted from the trigger are replicated.

This may result in data duplication in the case where a trigger causes data to be recorded twice for a single operation performed on an audited table. It may also result in data inconsistency in the case where a trigger results in DML commands being performed twice: once as a result of the trigger firing at the primary database and a second time as a result of the trigger firing at the replicate database to which trigger-altered data has already been replicated. To avoid data duplication and inconsistency, it is important to control trigger execution in the replication system. However, Oracle provides no session-level command to disable trigger execution.

Replication Server allows you to disable trigger execution at the session or connection level. You can control trigger firing each time a PL/SQL command is executed against the replicate

database. Controlling trigger execution at the replicate database eliminates data duplication and inconsistency caused by the absence of any trigger control at the replicate database.

For a complete description of the Replication Server **rs_triggers_reset** function, see the *Replication Server Reference Manual*. For complete instructions on controlling trigger execution at the replicate database, see "Oracle Replicate Data Server Issues" in the *Replication Server Heterogeneous Replication Guide*.

Alteration of Replication Definitions from the Primary Data Server

You can alter replication definitions from the primary data server.

To avoid having to quiesce the replication system before altering a replication definition, you can issue the Replication Server **alter replication definition** command from the primary data server and make schema changes to primary database objects at the same time. The propagation of changes to a replication definition can be automatically coordinated with data replication without having to stop the replication process.

To issue the Replication Server **alter replication definition** command from the primary data server, create a stored procedure named **rs_send_repserver_cmd** in the primary Oracle database. The SQL for creating this procedure is contained in the appropriate connection profile on Replication Server. For a list of connection profiles, use the Replication Server **admin show_connection_profiles** command.

For a full description of **rs_send_repserver_cmd** and the **alter replication definition** Replication Server command, see the *Replication Server Reference Manual*.

Security Considerations

When the **rs_send_repserver_cmd** procedure is invoked at the primary data server, Replication Agent passes corresponding Replication Command Language (RCL) directly to Replication Server. You should therefore consider carefully to whom execution privileges are assigned for the **rs_send_repserver_cmd** procedure, and assign privileges as appropriate for your environment and security policy.

Limitations

You cannot use the **rs_send_repserver_cmd** procedure to alter replication definitions for tables that contain columns of certain datatypes.

These types are:

- `BINARY ROWID`
- `BINARY UROWID`
- `DATE`
- `INTERVAL YEAR TO MONTH`
- `INTERVAL DAY TO SECOND`
- `MLSLABEL`
- `RAW`

- REF
- TIMESTAMP
- TIMESTAMP WITH LOCAL TZ
- TIMESTAMP WITH TZ

Note: If you manually change a table-level replication definition in Replication Server, you must then suspend and resume replication in the Replication Agent to ensure that the Replication Agent clears and refreshes its cache.

Oracle Data Guard

Data Guard is a disaster-protection architecture consisting of a primary Oracle database and one or more standby Oracle databases to which the primary database copies its data. These standby copies can be used in the event that the primary Oracle database fails. Replication Agent supports replication of data from an Oracle database system that uses Data Guard.

Recommended Configuration

Although you can configure Replication Agent to replicate data from either the Data Guard primary database or a Data Guard standby database, It is recommended that you configure Replication Agent to read from a Data Guard standby database transaction log. This way, if the primary Data Guard database fails over to the standby Data Guard database, Replication Agent is already connected to a working Oracle database, and replication is not disrupted.

If you configure Replication Agent to read from a Data Guard standby database transaction log, **pds_username** must have **alter database** permission.

Database Resynchronization

You can avoid having to quiesce the primary database when you initialize Replication Agent for Oracle if your replication system is configured for database resynchronization.

For information on configuring database resynchronization, see the *Replication Server Heterogeneous Replication Guide > Oracle Replicate Database Resynchronization*.

Oracle Transaction and Operation Troubleshooting

The **ra_dumptran** and **ra_helpop** commands return information for use in troubleshooting a specified Oracle database transaction or database operation, respectively.

The **ra_dumptran** and **ra_helpop** commands use information gathered by Oracle LogMiner to help you troubleshoot Replication Agent for Oracle. Oracle LogMiner consists of Oracle procedures and views that allow you to obtain detailed information about database activities from the Oracle redo logs. To use **ra_dumptran** and **ra_helpop**, you must install Oracle LogMiner, or these commands return errors. For details on using these commands, see the *Replication Agent Reference Manual*.

Setting Up Replication Agent and Oracle to use ra_dumptran and ra_helpop

Troubleshoot using **ra_dumptran** and **ra_helpop**.

1. Go to `$ORACLE_HOME/rdbms/admin`.
2. Log in as a "sys as sysdba" user.
3. Execute the Oracle LogMiner installation script:

   ```
   @dbmslm.sql
   ```

4. 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 step is required if you are using Oracle 10g.

5. Grant these privileges to **pds_username**:

   - **EXECUTE_CATALOG_ROLE**
   - **SELECT ON V_$LOGMNR_CONTENTS**
   - **SELECT ON V_$LOGMNR_LOGS**
   - **SELECT ANY TRANSACTION**

   Note: The **ra_migrate** command verifies that these privileges have been granted to **pds_username**. 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.

6. Use **ra_dumptran** and **ra_helpop** according to instructions provided in the *Replication Agent Reference Manual*.

Stored Procedure Replication with BOOLEAN Arguments

Learn how to replicate Oracle stored procedures with arguments of boolean type.

To replicate an Oracle stored procedure with an argument of type `BOOLEAN`, Replication Agent sends the `BOOLEAN` argument to Replication Server as an integer. Replication Server then uses a function string to convert the argument back to a `BOOLEAN` value so that the stored procedure can be executed on the replicate database. On Replication Server, you must manually create this function string for each Oracle stored procedure that has an argument of type `BOOLEAN`.

Some replicate databases do not support `BOOLEAN` stored procedure arguments. In these cases, the Oracle stored procedure `BOOLEAN` argument should be mapped to an integer argument in the corresponding stored procedure at the replicate database. A function string is then unnecessary.

These examples illustrate how to replicate an Oracle stored procedure with an argument of type `BOOLEAN` to a Oracle replicate database and to a non-Oracle replicate database.

Example: Replicating to an Oracle Replicate Database

Replicate a stored procedure with `BOOLEAN` arguments to an Oracle replicate database.

To replicate a stored procedure defined by these PL/SQL statements:

```
CREATE PROCEDURE boolproc (a IN BOOLEAN, b INT) AS
                BEGIN
                   IF (a = true) THEN
                      DBMS_OUTPUT.PUT_LINE('True');
                   ELSE
                      DBMS_OUTPUT.PUT_LINE('False or NULL');
                   ENDIF;
                END;
```

1. Manually create a replication definition on Replication Server using this RCL command:

   ```
   create function replication definition ra$xxx_boolproc
                           with primary at myprimary.pdb
                           with all functions named boolproc (
                           @"a" rs_oracle_decimal
                           @"b" rs_oracle_decimal )
                           searchable parameters(@"a", @"b")
                           send standby all parameters
   ```

   Note: If the Replication Agent **pdb_auto_create_repdefs** configuration parameter is set to true, a replication definition will be created automatically.

2. Mark the stored procedure for replication:

   ```
   pdb_setrepproc boolproc, mark
   ```

3. Create a function string on Replication Server:

   ```
   create function string ra$xxx_boolproc.boolproc
                           for rs_oracle_function_class
                           output language
                           'begin execute immediate "begin
   ra_user.boolproc
   (?a!param?=1,?b!param?);;end;;";;end;;'
                           go
   ```

4. Create a subscription on Replication Server for the replication definition:

   ```
   create subscription sub_intproc
                           for ra$xxx_boolproc
                           with replicate at myreplicate.rdb
                           go
   ```

The stored procedure is replicated when it is executed at the primary database:

```
EXECUTE boolproc(true,1);
```

Example: Replicating to a Non-Oracle Database

Replicate a stored procedure with `BOOLEAN` arguments to a non-Oracle replicate database.

To replicate a stored procedure defined by these PL/SQL statements:

```
CREATE PROCEDURE boolproc (a IN BOOLEAN, b INT) AS
                BEGIN
                   IF (a = true) THEN
                      DBMS_OUTPUT.PUT_LINE('True');
                   ELSE
                      DBMS_OUTPUT.PUT_LINE('False or NULL');
                   ENDIF;
                END;
```

1. Manually create a replication definition on Replication Server using this RCL command:

   ```
   create function replication definition ra$xxx_boolproc
                           with primary at myprimary.pdb
                           with all functions named boolproc (
                           @"a" rs_oracle_decimal
                           @"b" rs_oracle_decimal )
                           searchable parameters(@"a", @"b")
                           send standby all parameters
   ```

   Note: If the Replication Agent **pdb_auto_create_repdefs** configuration parameter is set to true, a replication definition will be created automatically.

2. Mark the stored procedure for replication:

   ```
   pdb_setrepproc boolproc, mark
   ```

3. Adaptive Server Enterprise does not support `BOOLEAN` stored procedure arguments, so you must map the Oracle stored procedure `BOOLEAN` argument to an integer argument for the corresponding stored procedure at the replicate database.
4. Create a stored procedure, defined by this Transact-SQL® statement, at the replicate database:

   ```
   create proc boolproc (@a int, @b int) as
                           if @a = 1
                              print 'True'
                           else
                              print 'False or NULL'
                           go
   ```

5. Create a subscription on Replication Server for the replication definition:

   ```
   create subscription sub_intproc
                           for ra$xxx_boolproc
                           with replicate at myreplicate.rdb
                           go
   ```

The stored procedure is replicated when it is executed at the primary database:

```
EXECUTE boolproc(true,1);
```

However, the **boolproc** procedure at the replicate Adaptive Server Enterprise will be invoked with an integer value instead of a `BOOLEAN` argument:

```
boolproc 1, 1
go
```

Oracle Warm Standby

Using Replication Server, you can create and maintain a warm standby environment for an Oracle database.

In standby mode, Replication Agent:

- Scans the transaction log and keeps the Replication Agent System Database (RASD) current
- Does not send any LTL to Replication Server
- Continues to perform log truncation

When the active database fails or when you want to perform maintenance on the active database, you can switch to the standby database. For instructions on switching the active and standby database, see "Managing Heterogeneous Warm Standby for Oracle" in the *Replication Server Heterogeneous Replication Guide*.

To function in warm standby mode:

- Replication Agent must be installed on both the primary and standby side and must also be successfully initialized. The Replication Agent on the standby side should be running in standby mode with the **ra_standby** parameter set to true.
- Replication Agent should have the **rs_source_ds** and **rs_source_db** parameters configured as physical connections to Replication Server.
- Replication Agent should enable or disable the replication of DDL statements as desired using the **pdb_setrep_ddl** command.
- Replication Agent should set the **pdb_auto_create_repdefs**, **pdb_dflt_column_repl**, **pdb_dflt_object_repl**, and **pdb_automark_tables** parameters to true.

For details on using these commands and configuration parameters, see the *Replication Agent Reference Manual*. For detailed steps involved in creating and managing warm standby for Oracle, see the *Replication Agent Heterogeneous Replication Guide*.

Oracle Flashback

Replication Agent can replicate Oracle Flashback operations performed at the table level and the transaction level.

- Replication Agent can replicate Flashback Table commands to restore a database back to a System Change Number (SCN), timestamp, or restore point within the threshold specified by the Oracle **UNDO_RETENTION** parameter. If a table has been marked for replication, Replication Agent can replicate any DML changes that result from executing the contents of the `UNDO_SQL` column of the Oracle `FLASHBACK_TRANSACTION_QUERY` view.
- Replication Agent replicates these Oracle DDL commands:
  - **DROP TABLE** *table* (when the Recycle Bin is enabled)
  - **FLASHBACK TABLE** *table* **TO BEFORE DROP**

- **PURGE TABLE** *table*
- **PURGE INDEX** *index*
- **PURGE TABLESPACE** *tablespace*
- **PURGE RECYCLEBIN**
- **PURGE DBA_RECYCLEBIN**

Note: Since the replicate database may not be an exact copy of the primary database, these DDL commands may not execute successfully at the replicate database or may have a different result at the replicate database. For example, the **PURGE DBA_RECYCLEBIN** command purges more objects at the replicate database if the Recycle Bin for the replicate database contains more objects than the primary database Recycle Bin.

- Replication Agent can read from the Oracle Flash Recovery Area if the **pdb_archive_path** configuration parameter is set to that location.

Requirements for Oracle Flashback

Observe these requirements when using Oracle Flashback with Replication Agent.

To use Oracle Flashback with Replication Agent:

- The user ID for **pds_username** must have **select** permission on **SYS.RECYCLEBIN$**.
- For Replication Agent to replicate the Oracle **PURGE DBA_RECYCLEBIN** command, the user ID for **ddl_username** must have **sysdba** permission and should be suffixed with "as sysdba". For example:

```
ra_config ddl_username, "myuser as sysdba"
go
```

- Replication Agent replicates Flashback operations from Oracle 10g and 11g databases but not from earlier versions of Oracle.

Limitations to Oracle Flashback

Replication of Oracle Flashback commands is subject to some limitations.

- If the replicate database does not have the Recycle Bin enabled, Flashback commands, the **PURGE** command, and any command that accesses Recycle Bin objects fails, even if a **DROP TABLE** command has been successfully replicated.
- Replication Agent does not support the translation of DDL commands between a primary and replicate database of different types. Consequently, DDL replication must be disabled when replicating from an Oracle primary database to a non-Oracle replicate database, and Flashback DDL commands cannot be replicated in this case.
- If an Oracle **FLASHBACK TABLE** command is issued with the **RENAME TO** clause, Replication Agent does not automatically update the replication definition with the new table name. You must do this manually.
- Replication Agent reconstructs Flashback commands based on the original object name, not on the object Recycle Bin name. When there are multiple versions of an object in the Recycle Bin, Replication Agent reconstructs a Flashback command to use the most recent version of the object that exists in the Recycle Bin at the replicate database. Consequently,

subsequent Oracle commands that affect the Recycle Bin may result in inconsistency between the primary and replicate databases.

For example, the primary Oracle database contains these dropped versions of table TAB1:

```
SQL> SELECT object_name as recycle_name, original_name,
            FROM recyclebin;
RECYCLE_NAME                     ORIGINAL_NAME  TYPE
-------------------------------  -------------  -----
BIN$zyxwvutsrqponmlkjihgfedcba$1  TAB1           TABLE
BIN$zyxwvutsrqponmlkjihgfedcba$2  TAB1           TABLE
BIN$zyxwvutsrqponmlkjihgfedcba$3  TAB1           TABLE
```

The replicate Oracle database contains these dropped versions of table TAB1:

```
SQL> SELECT object_name as recycle_name, original_name, type
            FROM recyclebin;
RECYCLE_NAME                     ORIGINAL_NAME  TYPE
-------------------------------  -------------  -----
BIN$abcdefghijklmnopqrstuvwxyz$1  TAB1           TABLE
BIN$abcdefghijklmnopqrstuvwxyz$2  TAB1           TABLE
BIN$abcdefghijklmnopqrstuvwxyz$3  TAB1           TABLE
```

This Flashback command is executed at the primary Oracle database:

```
FLASHBACK TABLE "BIN$zyxwvutsrqponmlkjihgfedcba$2" TO BEFORE
DROP;
```

Because Replication Agent reconstructs Flashback commands based on the original object name and uses the most recent version of a dropped object in the Flashback command, this command is executed at the replicate Oracle database:

```
FLASHBACK TABLE "BIN$abcdefghijklmnopqrstuvwxyz$3" TO BEFORE
DROP;
```

If BIN$zyxwvutsrqponmlkjihgfedcba$2 differs in content from BIN $abcdefghijklmnopqrstuvwxyz$3, the primary and replicate databases have become inconsistent.

Disabling Oracle Recycle Bin

If you do not intend to use the Recycle Bin at the replicate Oracle database, you can manually disable it.

Prerequisites

Disabling the Recycle Bin requires the **sysdba** privilege.

Task

Enter this command, and then restart the replicate Oracle database:

```
ALTER SYSTEM SET RECYCLEBIN=OFF SCOPE=SPFILE;
```

Or, if you are using a version of Oracle that does not have the RECYCLEBIN parameter, enter:

```
ALTER SYSTEM SET "_recyclebin"=FALSE SCOPE=BOTH;
```

Note: If you are using Oracle RAC, disable the recycle bin for each instance in the cluster.

Dropped Objects and Article Status

If a marked table is dropped while the Recycle Bin is enabled at the primary database, the **ra_helparticle** command still reports the status of the corresponding article as Current. **ra_helparticle** reports the status of the corresponding article as Dropped only after the dropped table is purged from the primary database Recycle Bin.

Disabling Flashback Replication with Recycle Bin Disabled

Disable the replication of Oracle Flashback DDL commands when the Recycle Bin is disabled.

If the replicate Oracle database does not have the Recycle Bin enabled, any command that accesses Recycle Bin objects at the replicate database fails. You should therefore disable the replication of Flashback DDL commands.

Disable the replication of Flashback DDL commands in one of these ways:

- Use the **pdb_setrepddl** command to prevent the replication of Flashback DDL commands. See the *Replication Agent Reference Manual*.
- Add a "warning" error action for Flashback DDL execution failure to Replication Server so that replication can continue with the replicate Recycle Bin disabled. Run this Replication Server script against the RSSD:
  ```
  $SYBASE/RAX-15_5/scripts/oracle/
  hds_oracle_setup_flashback_errors.sql
  ```
  After you run this script, you must restart Replication Server.

  Note: This script uses the **rs_oracle_error_class** default error class as a template. If you are using a custom error class and want Replication Server to continue replicating without interruption, you must instruct Replication Server to display warning messages 38305 and 38307 in its error log:

  ```
  assign action warn for your_error_class to 38305, 38307
  ```

  where *your_error_class* is the name of your custom error class.

  To remove the "warning" error action for Flashback DDL execution failure to Replication Server, run this Replication Server script against the RSSD:
  ```
  $SYBASE/RAX-15_5/scripts/oracle/
  hds_oracle_remove_flashback_errors.sql
  ```

XMLTYPE Data Replication

When an Oracle table is created with an `XMLTYPE` column but without any XML schema specification, a hidden `CLOB` column is automatically created to store the XML data. The `XMLTYPE` column becomes a virtual column for the hidden `CLOB` column. In the corresponding Oracle base table, the hidden column immediately follows the `XMLTYPE`

column that it represents and is named SYS_NC*nnnnn*$, where *nnnnn* represents the position of the hidden column in the base table.

For example, a table is created in the Oracle database with this DDL command:

```
CREATE TABLE sampletable
( col1 INT,
, col2 INT,
, xml1 XMLTYPE
, xml2 XMLTYPE);
```

The Oracle database creates hidden columns named `SYS_NC00004$` and `SYS_NC00006$`, which respectively correspond to the `xml1` and `xml2` columns. These hidden `CLOB` columns cannot be accessed directly. However, they can be viewed by querying the `col$` and `obj$` base tables of the Oracle data dictionary. See the Oracle documentation.

See also

- *Oracle XMLTYPE Datatype Compatibility* on page 33
- *Oracle 10g, 11g, and 12c XMLTYPE Restrictions* on page 35

Example: Replicating XMLTYPE Column Data from Oracle to Oracle

Replicate XMLTYPE column data from an Oracle primary database to an Oracle replicate database.

To replicate a table defined by this DDL statement:

```
CREATE TABLE sampletable
( col1 INT,
  col2 INT,
  xml1 XMLTYPE,
  xml2 XMLTYPE)
XMLTYPE xml1 STORE AS CLOB
XMLTYPE xml2 STORE AS CLOB;
```

1. Manually create a replication definition on SAP Replication Server this RCL command:

   ```
   create replication definition ra$xxx_sampletable
   with primary at myprimary.pdb
   with all tables named sampletable (
   col1 int,
   col2 int,
   xml1 as SYS_NC00004$ text,
   xml2 as SYS_NC00006$ text )
   primary key (col1, col2)
   go
   ```

   Note: If the Replication Agent **pdb_auto_create_repdefs** configuration parameter is set to true, a replication definition is created automatically.

2. Mark the table for replication:

   ```
   pdb_setreptable sampletable, mark
   ```

3. Because of the hidden `CLOB` columns, you must enable replication for the table using **pdb_setrepcol**:

   ```
   pdb_setrepcol sampletable, enable
   ```

4. Create a corresponding table at the replicate Oracle database:

   ```
   CREATE TABLE sampletable
   ( col1 INT,
     col2 INT,
     xml1 XMLTYPE,
     xml2 XMLTYPE);
   ```

Note: If DDL replication has been enabled, manually create the replicate table.

Example: Replicating XMLTYPE Column Data from Oracle to SAP Adaptive Server Enterprise

Replicate XMLTYPE column data from an Oracle primary database to an SAP Adaptive Server Enterprise (SAP ASE) replicate database.

To replicate a table defined by this DDL statement:

```
CREATE TABLE sampletable
( col1 INT,
  col2 INT,
  xml1 XMLTYPE,
  xml2 XMLTYPE)
XMLTYPE xml1 STORE AS CLOB
XMLTYPE xml2 STORE AS CLOB;
```

1. Manually create a replication definition on SAP Replication Server using this RCL command:

   ```
   create replication definition ra$xxx_sampletable
   with primary at myprimary.pdb
   with all tables named sampletable (
   col1 int,
   col2 int,
   xml1 as SYS_NC00004$ text,
   xml2 as SYS_NC00006$ text )
   primary key (col1, col2)
   go
   ```

   Note: If the Replication Agent **pdb_auto_create_repdefs** configuration parameter is set to true, a replication definition is created automatically.

2. Mark the table for replication:

   ```
   pdb_setreptable sampletable, mark
   ```

3. Because of the hidden `CLOB` columns, you must enable replication for the table using **pdb_setrepcol**:

   ```
   pdb_setrepcol sampletable, enable
   ```

4. Create a corresponding table at the replicate SAP ASE database using the hidden column names from the primary database table:

```
create table sampletable
( col1 int,
col2 int,
SYS_NC00004$ text,
SYS_NC00006$ text)
go
```

Note: The `XMLTYPE` columns from the Oracle primary database table map to `text` columns in the SAP ASE replicate database table.

Example: Replicating an XMLTYPE Table from Oracle to Oracle

Replicate an XMLTYPE table from an Oracle primary database to an Oracle replicate database.

This statement creates a simple `XMLTYPE` table with one implicit `CLOB` column that can be accessed through a default pseudocolumn named `XMLDATA`:

```
CREATE TABLE sampletable OF XMLTYPE XMLTYPE STORE AS CLOB;
```

To replicate a table defined by this DDL statement:

1. Manually create a replication definition on Replication Server using this RCL command and the hidden column `SYS_NC_OID$`, which contains the object ID for `sampletable`:

   ```
   create replication definition ra$xxx_sampletable
   with primary at myprimary.pdb
   with all tables named sampletable (
   SYS_NC_OID$ rs_oracle_binary,
   XMLDATA text )
   primary key (SYS_NC_OID$)
   go
   ```

   The Replication Server name for the Oracle `RAW` datatype is `rs_oracle_binary`.

   Note: If the Replication Agent **pdb_auto_create_repdefs** configuration parameter is set to true, a replication definition is created automatically.

2. Mark the table for replication:

   ```
   pdb_setreptable sampletable, mark
   ```

3. Create a corresponding table at the replicate Oracle database:

   ```
   CREATE TABLE sampletable OF XMLTYPE XMLTYPE STORE AS CLOB;
   ```

Oracle 9i

Replication Agent supports Oracle 9i when Oracle 10g, 11g, or 12c is running in 9i compatibility mode.

Limitations to Oracle 9i

Some Replication Agent features cannot be used with Oracle 9i.

These Replication Agent features cannot be used with Oracle 9i:

- Oracle index-organized tables and the `ANYDATA` datatype
- Autocorrection
- Replication of `XMLTYPE` data
- Flashback
- RAC
- ASM

Oracle 11g Release 2

All of the functionality that Replication Agent supports for Oracle Database 11g Release 1 is also supported by Replication Agent for Oracle Database 11g Release 2. Replication Agent also supports some functionality introduced by Oracle Database 11g Release 2.

Note: If you want to replicate tables compressed for online transaction processing (OLTP), apply first the Oracle patch# 129050503.

Replication Agent supports functionality common to both Oracle Database 11g Release 1 and Oracle Database 11g Release 2, including:

- Oracle DDL and DML replication in systems with and without Automatic Storage Management (ASM) and Real Application Clusters (RAC).
- Use of the Oracle Recovery Manager (RMAN) utility to truncate old archive log files.
- Use of the Oracle Recycle Bin and replication of Oracle Flashback operations.
- Oracle Data Guard.

Replication Agent also supports some features that are new as of Oracle Database 11g Release 2:

- Use of the **FORCE** option with a **CREATE OR REPLACE TYPE** statement on types with type dependencies.
- DDL statements on tables enabled for Flashback Data Archive.
- Version 11.2 time zone files and new time zone behavior.

Network Configuration File Location and Structure

The `tnsnames.ora` file is located in `ORACLE_HOME\network\admin`. In an Oracle Database 11g Release 2 instance running ASM or RAC, the `tnsnames.ora` file is read by default from the grid infrastructure home directory at `Grid_home\network\admin`.

If you are using an Oracle Database 11g Release 2 instance running ASM or RAC, set the Replication Agent **asm_tns_filename** parameter to `Grid_home\network\admin \tnsnames.ora`.

By default, the `tnsnames.ora` file at `Grid_home\network\admin` contains an incomplete ASM entry that lacks information in the DESCRIPTION and SERVICE_NAME fields. If you are using an Oracle Database 11g Release 2 instance running ASM, set the Replication Agent **asm_tns_connection** parameter to the ASM connection name specified in this incomplete ASM entry. Replication Agent completes the DESCRIPTION and SERVICE_NAME fields, and you can then use the `tnsnames.ora` file in `Grid_home\network\admin` to connect to the ASM instance server.

Time Zone File

By default, Oracle Database 11g Release 2 uses the large time zone file, `timezone_11.dat`. This file contains all the time zones defined in the database.

If you are using Oracle Database 11g Release 2, set the Replication Agent **pdb_timezone_file** parameter to the location of the `timezone_11.dat` file:

```
ra_config pdb_timezone_file, $ORACLE_HOME/oracore/zoneinfo/
timezone_11.dat
```

User-Defined Type Dependencies

You can use the **CREATE OR REPLACE TYPE** command to change the definition for an existing user-defined type. However, this command throws an error if the referenced type has table or type dependencies.

Oracle Database 11g Release 2 allows you to use **FORCE** with the **CREATE OR REPLACE TYPE** command to replace a type that has a type dependency:

```
CREATE TYPE mytype1 AS OBJECT (a number) NOT FINAL;
CREATE TYPE mytype2 UNDER mytype1 (b varchar(10));
CREATE OR REPLACE TYPE mytype1 FORCE AS OBJECT (c varchar(20));
```

Oracle Database 11g Release 2 does not allow you to use **FORCE** with the **CREATE OR REPLACE TYPE** command to replace a type that has a table dependency:

```
CREATE TABLE mytable1 (colA mytype1);
CREATE OR REPLACE TYPE mytype1 FORCE AS OBJECT (d number);
```

The last command results in an error because `mytype1` has a table dependency on `mytable1`:

```
ERROR at line 1:
ORA-22866: cannot replace a type with table dependents
```

Replication Agent supports use of the **FORCE** option with the **CREATE OR REPLACE TYPE** command in Oracle Database 11g Release 2 to replace types with type dependencies but not for types with table dependencies.

Flashback Data Archive Support for DDL Commands

Oracle Flashback allows database administrators and users to view past states of database objects and to restore database objects to a previous state without using point-in-time

recovery. Users can query past data, query metadata to build a detailed history of changes, recover data to a previous point in time, and roll back transactions while the database is online.

Replication Agent supports the replication of DDL commands on tables that are being tracked with the Flashback Data Archive in Oracle Database 11g Release 2. These DDL commands include:

- **Add**, **Drop**, **Rename**, **Modify Column**
- **Drop**, **Truncate Partition**
- **Rename**, **Truncate Table**
- **Add**, **Drop**, **Rename**, **Modify Constraint**

Oracle XStream

The Replication Agent for Oracle Log Reader component uses Oracle XStream APIs to capture data changes and committed transactions from an Oracle primary database. The XStream APIs enable you to share data changes between Oracle and non-Oracle data sources, such as SAP® Replication Server®.

The Replication Agent for Oracle supports replication of:

- Supported datatypes specified in *Supported and Unsupported Datatypes* on page 75.
- Stored procedures and marker objects.
- Data in Oracle sequences.
- Data in a materialized view table.
- Compression; the level of compression support is dependent on the level of compression supported by Oracle XStream.
- Encryption; the level of encryption support is dependent on the level of encryption supported by Oracle XStream.

Replication Agent observes the same limitations for Oracle data as XStream APIs. For information about those limitations and Oracle XStream, see the Oracle documentation.

Supported Databases and JDBC Drivers

Check the supported Oracle database, and Java Database Connectivity (JDBC) driver versions required to use Replication Agent for Oracle with XStream APIs.

Database	Version
Oracle	11g Release 2 (patch 3 or later) –11.2.0.3 or later

Replication Agent for Oracle requires a JDBC 4.0-compliant driver for the Oracle primary data server.

Driver	Version
Oracle JDBC driver	11.2.0.3 or later

To use the standard JDBC 4.0 driver for Oracle database version 11.2.0.3:

- Download the `ojdbc6.jar` JDBC driver. See the Oracle documentation for instructions on how to download.
- Install the same version of the JDBC driver in the Oracle database environment in which Replication Agent resides.

Technical Prerequisites

Prerequisites for using the Replication Agent for Oracle with XStream APIs.

- The Replication Agent primary database user **pds_username** must have the XStream administrator privilege to:
  - Create or drop an XStream outbound server
  - Receive or send data changes from an XStream outbound server
  - Capture, synchronize, and apply data changes to an inbound server

  To grant the XStream administrator privilege to the **pds_username**, issue:

  ```
  BEGIN
  DBMS_XSTREAM_AUTH.GRANT_ADMIN_PRIVILEGE( grantee =>
  'pds_username');
  END;
  /
  ```

  See the Oracle documentation for the **DBS_XSTREAM_AUTH** package information.
- When you use XStream APIs to capture data changes, grant read/write access to the primary database archive logs to enable a non-Oracle Recovery Manager (RMAN) archive truncation method.

  If you are using Oracle RMAN for archive truncation, or if the archive truncation method is disabled, read/write access to the archive logs is not required.

  Depending on the configuration, Replication Agent may also access archived transactions logs (default) or may process only online transaction logs. For information about redo log and archive log files, see the *Replication Agent Primary Database Guide*.
- Before creating a Replication Agent instance, if the primary TNS service name is different from the **pds_database_name**, set the **pds_service_name** to the Oracle Transparent Network Substrate (TNS) service name found in the `$ORACLE_HOME/network/admin/tnsnames.ora` file.

  For example, if `tnsnames.ora` contains these values:

  ```
  ORCL112 =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = huyharry)(PORT = 1521))
      (CONNECT_DATA =
        (SERVER = DEDICATED)
  ```

```
      (SERVICE_NAME = orcl112.sybase.com)
    )
```

Set the **pds_service_name** to `orcl112.sybase.com` instead of `orcl112` in the resource file.

- When you use XStream APIs to read from the transaction log, Replication Agent sets the Oracle Call Interface (OCI) server connection to Dedicated mode by default. If you change the OCI connection mode to Shared, the `ORA-26908` exception occurs. For mode descriptions and error exception details, see the Oracle documentation.
- When the **Oracle Database Vault** parameter is enabled at the primary database, you must grant the required privileges to the **pds_username** using the Oracle user login with the **DV_OWNER** role. Otherwise, you cannot execute the **pdb_xlog init** command to create, view, or remove the Replication Agent system objects.

  To grant the required permissions to the **pds_username**:

  1. Check whether **Oracle Database Vault** is enabled at the primary database by issuing:

     ```
     SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database
     Vault';
     ```

     A returned result of true indicates that the Oracle Database Vault is enabled, continue with next steps.

     If the result is false, ignore remaining steps.
  2. Execute **ra_admin prepare** to generate a SQL script called `prepare_pdb_with_vault_for_replication_init.sql` in the `<instance_directory>/scripts/prepare`, where *instance_directory* is the Replication Agent installation directory.
  3. Run the `prepare_pdb_with_vault_for_replication_init.sql` script to grant the required privileges.

  For more information about Oracle Database Vault, see the Oracle documentation.

Supported and Unsupported Datatypes

Datatypes that are supported and not supported by Replication Agent for Oracle 15.7.1 SP120 using XStream APIs.

Replicating Datatypes Out of Oracle

Oracle datatypes that are supported or not supported by Replication Agent for replicating out of Oracle using XStream APIs.

Table 4. Supported and Unsupported Oracle Datatypes

Supported	Unsupported
`binary_double`	`bfile`
`binary_float`	`rowid`

Supported	Unsupported
`blob`	`ref`(user-defined datatype)
`clob`	`varray`(user-defined object)
`char`	`nested tables` (user-defined object)
`date`	`xmltype` (such as `binary xml`)
`interval day to second`	
`interval year to month`	
`number`	
`raw`	
`timestamp`	
`timestamp with [local] time zone`	
`timestamp with time zone`	

Oracle-Supplied Datatype Limitations

Replication Agent cannot replicate some Oracle-supplied datatypes, because XStream APIs do not support them in row Logical Change Records (LCRs).

- "ANY" types (`SYS.ANYTYPE`, `SYS.ANYDATASET`)
- `BFile`
- `ROWID`
- User-defined datatypes, such as `REF`, including these object types:
  - `NESTED TABLE`
  - `VARRAY`
- `XMLType` object stored relationally or as a `binary XML`
- Media types (`ORDSYS.ORDAudio`, `ORDSYS.ORDImage`, `ORDSYS.ORDImageSignature`, `ORDSYS.ORDVideo`, `ORDSYS.ORDDoc`, `SI_StillImage`, `SI_Color`, `SI_AverageColor`, `SI_ColorHistogram`, `SI_PositionalColor`, `SI_Texture`, `SI_FeatureList`)
- Spatial types (`MDSYS.SDO_GEOMETRY, SDO_TOPO_GEOMETRY, SDO_GEORASTER`)
- Uniform Resource Identifier (URI) Types

These datatype restrictions are applicable to both ordinary (heap-organized) tables and index-organized tables. For descriptions about these tables, see the Oracle documentation.

Replication Agent for Oracle Limitations

The limitations for using Replication Agent for Oracle with XStream APIs.

- Because Replication Agent for Oracle does not support multiple XStream outbound servers, Multi-Path Replication™ (MPR) is not supported.
- Because XStream APIs return only committed transactions, Replication Agent for Oracle does not support replication of stored procedures in which autonomous transactions exist.
- Because the data definition language (DDL) LCR does not return the global temporary table details from an XStream outbound server, replication of temporary tables is not supported.
- Because XStream APIs does not support changes to a table created with the reference partition, Replication Agent for Oracle cannot replicate Data Manipulation Language (DML) statements on such tables.

Configuring the Replication Agent for Oracle to Use XStream APIs

Configure the Replication Agent for Oracle to use XStream APIs with a new **oraclexs** database type or a resource file.

Setting Up an Oracle XStream JDBC Driver

To use the Replication Agent for Oracle with XStream APIs, use the **setenv** function to set or modify ORACLE_HOME, LD_LIBRARY_PATH, and CLASSPATH environment variables.

Prerequisites

- Download and install the appropriate XStream JDBC driver from the Oracle Web site for your primary Oracle database environment in which Replication Agent resides.
- Make sure your Java Runtime Environment (JRE) is compatible with your Oracle database for using XStream APIs. For example, if you have installed 64-bit Oracle, use the 64-bit JRE 1.7.x. See the Oracle documentation.

  By default, Replication Agent uses 64bit JRE when creating an instance with the XStream APIs Log Reader component.

Task

1. Set the ORACLE_HOME environment variable to the Oracle installation directory for your platform:

   ```
   setenv $ORACLE_HOME <install_dir>/oracle/<version>/
   ```

   where *install_dir* is the installation directory and *version* is the version number of the Oracle database.

The Oracle installation directory might be different, based on the database version installed on your platform. Set the ORACLE_HOME variable before setting the LD_LIBRARY_PATH and CLASSPATH environment variables.

2. Add the path of the Oracle XStream JDBC driver to the CLASSPATH environment variable for your platform.

   Add only one version of a JDBC driver in the CLASSPATH; otherwise, Replication Agent will have problems connecting to the primary database.

   On Linux:

   ```
   setenv LD_LIBRARY_PATH $ORACLE_HOME/lib:$LD_LIBRARY_PATH
   ```

   ```
   setenv CLASSPATH $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/
   rdbms/jlib/xstreams.jar:$CLASSPATH
   ```

   On HP-UX64:

   ```
   setenv SHLIB_PATH $ORACLE_HOME/lib:$SHLIB_PATH
   ```

   ```
   setenv CLASSPATH $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/
   rdbms/jlib/xstreams.jar:$CLASSPATH
   ```

   On AIX 64:

   ```
   setenv LIBPATH $ORACLE_HOME/lib:$LIBPATH
   ```

   ```
   setenv CLASSPATH $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/
   rdbms/jlib/xstreams.jar:$CLASSPATH
   ```

   On SUNOS-i86pc:

   ```
   setenv LD_LIBRARY_PATH $ORACLE_HOME/lib:$LD_LIBRARY_PATH
   ```

   ```
   setenv CLASSPATH $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/
   rdbms/jlib/xstreams.jar:$CLASSPATH
   ```

   On SUNOS-64:

   ```
   setenv LD_LIBRARY_PATH_64 $ORACLE_HOME/lib:$LD_LIBRARY_PATH_64
   ```

   ```
   setenv CLASSPATH $ORACLE_HOME/jdbc/lib/ojdbc6.jar:$ORACLE_HOME/
   rdbms/jlib/xstreams.jar:$CLASSPATH
   ```

   where:

   - *Oracle_Home* – is the Oracle database installation directory.
   - *path_name* – is the library path where you installed the JDBC driver:
     - *LD_LIBRARY_PATH* on UNIX
     - *SHLIB_PATH* on HP-UX64
     - *LIBPATH* on AIX
   - *driver* – is the name of the JDBC driver; for Oracle 11.2.0.3, `ojdbc6.jar`. The `xstreams.jar` is the name of the XStream library path.

3. (On Windows) Set Oracle environment variables in any of these ways:

Use the Control Panel	• Select **Start** > **Settings** > **Control Panel** > **System** > **Advanced System Settings** > **Environment Variables**.
	• Click **New** in the User variables pane.
	• In the New User Variable dialog, enter:
	  • **Variable name –** is the environment variable that you want to set. For example, ORACLE_HOME, PATH, and CLASSPATH.
	  • **Variable value –** is the directory path that you want to set for the environment variable. For example, `c:\oracle\version` (installation directory), `C:\ORACLE_HOME\bin` (the library path), and `C:\ORACLE_HOME\jdbc\lib\ojdbc6.jar` (the JDBC driver path).

Edit the existing variables	• Select the PATH and CLASSPATH environment variables in the System variables pane, and click **Edit**.
	• In the Edit System Variable dialog, using the semicolon (;) path separator, enter the variable value for:
	  • Path – `%ORACLE_HOME%\bin;%ORACLE_HOME%\LIB;%path%`
	  • CLASSPATH – `%ORACLE_HOME%\jdbc\lib\ojdbc6.jar;%ORACLE_HOME%\rdbms\jlib\xstreams.jar;%CLASSPATH%`

In the command line prompt	Enter: `set CLASSPATH=ORACLE_HOME\jdbc\lib\ojdbc6.jar;ORACLE_HOME\rdbms\jlib\xstreams.jar`

Creating a Replication Agent Instance

Create a Replication Agent instance, which uses XStream APIs to capture replicated transactions from an Oracle primary database.

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

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

For parameter descriptions, see the **ra_admin** command in the *Replication Agent Administration Guide*.

In this example, you create a Replication Agent instance named `raoxs`, which listens to port `8888` and database type is `oraclexs` for Oracle XStream:

```
ra_admin -c raoxs -t oraclexs -p 8888
```

The **oraclexs** parameter initializes the Replication Agent Log Reader component to use XStream APIs to capture committed transactions.

Creating a Replication Agent Instance Using a Resource File

Create a Replication Agent instance using the **oraclexs** resource file template for Oracle XStream.

Invoke **ra_admin** using the command line parameters that create and validate a Replication Agent instance using a resource file:

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

where:

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

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

In addition to creating a Replication Agent instance, the **ra_admin** parameters in the resource file allows you to:

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

To create a Replication Agent instance with XStream APIs, set the **instance_type** configuration parameter in the resource file as `instance_type=oraclexs`.

Note: When you create a Replication Agent instance with a resource file, the **pds_username** and **pds_password** configuration parameter values cannot contain single or double quotes if the **create_pds_username** parameter is set to yes.

Creating and Validating a Replication Agent Resource File

Create a resource file using the **oraclexs** template and validate the file by issuing the **ra_admin** command with the **-vr** option.

Prerequisites

Locate your resource file templates for Oracle database in the `init` subdirectory of the Replication Agent installation directory. For example:

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

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

Note: It is recommended that you to validate each resource file before you create a Replication Agent instance using that resource file.

Task

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

   For example:

   ```
   cp oraclexs.rs raoxs.rs
   ```

   where `raoxs.rs` is the name of the new resource file you want to create.
2. Use a text editor to edit the resource file copy that you created.
3. Validate your resource file by executing:

   ```
   ra_admin -vr raoxs.rs
   ```

   where `raoxs.rs` is the name of the resource file you want to validate.

   Validation results are returned as one of:
   - `Response-file processing completed`.
   - `Response-file processing completed with errors.`

   If the validation is successful, skip step 4, and use the resource file to create a Replication Agent instance.
4. To correct validation errors:
   a) Review the error messages to determine the cause of the failure.
   b) Edit the resource file to correct the appropriate values.
   c) Re-run **ra_admin -vr** specifying the name of the resource file.

   Repeat this step until the resource file is successfully validated.

Creating a Replication Agent Instance with a Resource File

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

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

- If the Replication Agent for Oracle primary database user login does not exist in the primary data server, the utility creates it, if specified in the resource file (`create_pds_username=yes`).
- If the user login exists in the primary data server but does not have all the required privileges, set **create** to yes, to have the utility grant all required permissions.
- If the resource file specifies to initialize new Replication Agent instance (`initialize_instance=yes`), then:
  - The Replication Agent primary database user login must either exist in the primary data server, or be created by **ra_admin** (`create_pds_username=yes`).
  - The resource file must specify the instance type for Oracle XStream (`instance_type=oraclexs`).

- The resource file must specify to start the Replication Agent instance (`start_instance=yes`).

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

Note: We recommend you to validate a new resource file before you create a Replication Agent instance using the new resource file.

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

```
ra_admin -r c:/raoxs.rs
```

where `raoxs.rs` is the name of the resource file.

Results are returned as one of:

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

If the instance creation is successful, the `oraclexs.cfg` configuration file is created in `$SYBASE/RAX/15_5/config` directory. You can begin using the new Replication Agent instance.

The configuration file now includes the XStream specific LogReader and LogAdmin '`ra_log_reader=com.sybase.ra.lr.oracle.xstream.RAOXSLogReader`' and '`ra_log_admin=com.sybase.ra.la.oracle.xstream.RAOXSLogAdmin`'.

Initializing a Replication Agent Instance to Use Oracle XStream APIs

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

Prerequisites

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

Task

1. Log in to the Replication Agent instance with the administrator login.
2. Run **ra_admin init**, where **init** is the keyword for creating Replication Agent system objects in the primary database.

   If the XStream administrator privilege is granted to the **pds_username**, an XStream outbound server is created during initialization.

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

If the initialization does not succeed, see *Replication Agent Administration Guide* for troubleshooting information.

Deinitializing a Replication Agent Instance

Deinitialize an instance of Replication Agent, if that instance is no longer required, and remove its objects from the primary database.

Prerequisites

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

Task

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

   ```
   ra_admin
   ```

   If there are no Replication Agent objects in the primary database, the **ra_admin** command returns no information, and you do not need to complete this task.

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

   ```
   pdb_setreptable all, disable
   ```
4. Disable replication for all marked procedures in the primary database:

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

   ```
   pdb_setreptable all, unmark
   ```

   ```
   pdb_setrepproc all, unmark
   ```
6. Remove the Replication Agent objects:

   ```
   ra_admin deinit
   ```

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

   If the deinitialization was successful, the script is stored in a file named `cleanup_pdb_for_replication_deinit.sql` in the `RAX-15_5\inst_name\scripts\cleanup` directory. Execute this script to remove Replication Agent instance objects.

Oracle 12c Release 1

All of the functionality that Replication Agent supports for Oracle Database 11g Release 2 is also supported by Replication Agent for Oracle Database 12c Release 1.

Replication Agent does not support the new features introduced by Oracle Database 12c Release 1.

Replication Agent Objects in the Oracle Primary Database

Replication Agent creates objects in the primary database to assist with replication tasks.

The Replication Agent objects are created by invoking the **ra_admin** command with the **init** keyword. When you invoke this command, Replication Agent generates a SQL script that contains the SQL statements for the objects created or modified in the primary database. This script is stored in the `partinit.sql` file in the `RAX-15_5\inst_name\scripts\xlog` directory. You must create these objects before marking any primary database objects for replication.

Note: The generated scripts are for informational purposes only. You cannot run them manually to initialize the primary database or Replication Agent. This is also true for the procedure marking and unmarking scripts that are generated when you use **pdb_setrepproc**. Scripts are no longer generated when marking and unmarking tables with **pdb_setreptable**.

See the *Replication Agent Administration Guide*.

Replication Agent Object Names

Replication Agent creates objects in the primary database to assist with replication tasks.

There are two variables in Replication Agent database object names:

- prefix – represents the one-to-three-character string value of the **ra_admin_instance_prefix** parameter (used for Replication Agent for Microsoft SQL Server or Replication Agent for UDB) or the **ra_admin_prefix** parameter.
- xxx – represents an alphanumeric counter, a string of characters that is (or may be) added to a database object name to make that name unique in the database.

The value of the **ra_admin_instance_prefix** parameter is the prefix string used in all Replication Agent object names.

The value of the **ra_admin_prefix_chars** parameter is a list of the nonalphanumeric characters allowed in the prefix string specified by **ra_admin_instance_prefix**. This list of allowed characters is database-specific. For example, in Oracle, the only nonalphanumeric characters allowed in a database object name are the $, #, and _ characters.

Use the **ra_admin** command to view the names of Replication Agent transaction log components in the primary database.

See the *Replication Agent Administration Guide* for details on setting up object names.

Finding the Names of the Objects Created

Find the names of Replication Agent objects created in the primary Oracle database.

Use the **ra_admin** command to return a list of all the Replication Agent objects created in the primary database.

Table Objects

Replication Agent creates table objects in the Oracle primary database.

These tables are considered Replication Agent objects.

Table 5. Replication Agent Tables

Table	Database Name
Procedure-active table	<*prefix*>PROCACTIVE
Multiple Replication Agents instances table	<*prefix*>AGENT
Multiple Replication Agents marked-tables table	<*prefix*>TABLE
Multiple Replication Agents marked-procedures table	<*prefix*>PROCEDURE

Procedure-Active Table

Maintains information about the actively running procedures during replication.

Table 6. Procedure-Active Table

Column Name	Type	Contents
callseq	NUMBER	The procedure call sequence.
sid	NUMBER	The ID of the session the procedure is executed on.
owner	VARCHAR2(256)	The procedure owner.
sproc	VARCHAR2(256)	The procedure name.
spid	NUMBER	The procedure object ID.
shadow	VARCHAR2(256)	The procedure shadow table name.
shid	NUMBER	The procedure shadow table object ID.
objtype	NUMBER	The procedure object type.

Column Name	Type	Contents
sqlerrm	VARCHAR2(256)	The text of the SQL error message for any error that occurred during procedure execution.

Multiple Replication Agents Instances Table

Maintains information about each Replication Agent instance in a Replication Agent group.

The multiple Replication Agents instances table is named *ra_admin_prefix*AGENT, where *ra_admin_prefix* is the prefix string used for Replication Agent system object names.

Table 7. Multiple Replication Agents Instances Table

Column Name	Type	Contents
prefix	VARCHAR2(3)	The instance prefix for this Replication Agent instance. This prefix is determined by the setting of the **ra_admin_instance_prefix** parameter.
locator	VARCHAR2(128)	The truncation point for this Replication Agent instance.
inittime	TIMESTAMP	The time at which this Replication Agent instance was initialized.
version	VARCHAR2(8)	The version of this Replication Agent instance.
rasd_export_file	VARCHAR(255)	The file to which RASD data is exported.
upg_from_ver	VARCHAR(16)	The version from which Replication Agent has been upgraded.
upg_fin_ts	TIMESTAMP	The time at which the upgrade finished.

Multiple Replication Agents Marked-Tables Table

Maintains information about tables marked for replication in a Replication Agent group.

The multiple Replication Agents marked-tables table is named *ra_admin_prefix*TABLE, where *ra_admin_prefix* is the prefix string used for Replication Agent system object names.

Table 8. Multiple Replication Agents Marked-Tables Table

Column Name	Type	Contents
prefix	VARCHAR2(3)	The instance prefix for this Replication Agent instance. This prefix is determined by the setting of the **ra_admin_instance_prefix** parameter.
tableid	NUMBER	The object ID for this marked table.

Column Name	Type	Contents
autocorrection	NUMBER(38)	Indicates whether or not autocorrection is enabled for the marked table. If autocorrection is enabled, this column has a value of 1. Otherwise, the column has a value of 0 (default).
ruletype	NUMBER(38)	Not used.
rulevalue	VARCHAR2(30)	Not used.

Multiple Replication Agents Marked-Procedures Table

Maintains information about procedures marked for replication in a Replication Agent group.

The multiple Replication Agents marked-procedures table is named *ra_admin_prefix*PROCEDURE, where *ra_admin_prefix* is the prefix string used for Replication Agent system object names.

Table 9. Multiple Replication Agents Marked-Procedures Table

Column Name	Type	Contents
prefix	VARCHAR2(3)	The instance prefix for this Replication Agent instance. This prefix is determined by the setting of the **ra_admin_instance_prefix** parameter.
procid	NUMBER	The object ID for this marked procedure.
shadowtable	VARCHAR2(30)	The object ID of the shadow table corresponding to this procedure.
ruletype	NUMBER(38)	Not used.
rulevalue	VARCHAR2(30)	Not used.

Marker Objects

Replication Agent creates marker objects in the primary database.

These Replication Agent objects are related to Replication Server markers. No permissions are granted when these objects are created.

Table 10. Replication Agent Marker Objects

Object	Name
Transaction log marker procedure	RS_MARKER
Dump marker procedure	RS_DUMP

Object	Name
Transaction log marker shadow table	*<prefix>*MARKERSH[*xxx*]
Dump marker shadow table	*<prefix>*DUMPSH[*xxx*]

Sequences

Replication Agent creates sequences in the Oracle primary database.

These Oracle sequences are considered Replication Agent objects.

Table 11. Replication Agent Sequences

Sequence	Database Name
Assign procedure call	*<prefix>*PCALL_[*xxx*]

Marked Procedures

Replication Agent creates objects for each primary procedure marked for replication in the Oracle primary database.

These Replication Agent objects are created for each primary procedure that is marked for replication. These objects are created only when a procedure is marked for replication.

Table 12. Replication Agent Objects for Each Marked Procedure

Object	Name
Shadow table	*<prefix><procedure_name>*SH

Shadow Table

Maintains information about the procedure call sequence and also maintains a column for every procedure argument where the column definition matches the definition of the argument.

Table 13. Shadow Table

Column Name	Type	Contents
callseq	NUMBER	The procedure call sequence.

Transaction Log Truncation

Replication Agent supports both automatic and manual log truncation.

Replication Agent provides two options for automatic transaction log truncation:

- Periodic truncation, based on a time interval you specify

- Automatic truncation whenever Replication Agent receives a new LTM locator value from the primary Replication Server. You also have the option to switch off automatic log truncation. By default, automatic log truncation is enabled and is set to truncate the log whenever Replication Agent receives a new LTM locator value from the primary Replication Server.

To configure Replication Agent log truncation, observe these guidelines:

- When **pdb_include_archives** is set to true, the default, and **pdb_archive_remove** is set false, the Replication Agent does not perform any online or archived transaction log truncation. When **pdb_include_archives** is set to true, the default, and **pdb_archive_remove** is set to true, Replication Agent deletes from the **pdb_archive_path** location the archive redo logs that have already been processed. The Replication Agent is not responsible for archiving online transaction logs.

  Note: It is recommended that you configure the Replication Agent to remove archive log files only if an additional archive log directory is used.

- When the configuration parameter **pdb_include_archives** is set to false, Replication Agent performs online redo log truncation (either scheduled or manual) by issuing the **alter system** command with the archive log sequence keywords. The command uses the log sequence number of the redo log file whose contents have been processed by the Replication Agent and are ready to be archived.

  Note: The **alter system** command syntax in Oracle allows redo log files to be archived in addition to the single log sequence specified in the command. To avoid unintentional archiving, Replication Agent issues this command only when it is processing the redo log file whose status is current.

- You can specify the automatic truncation option you want (including none) by using **ra_config** to set the value of the **truncation_type** configuration parameter.

  To truncate the transaction log automatically based on a time interval, use **ra_config** to set the value of the **truncation_interval** configuration parameter.
- You can truncate the Replication Agent transaction log manually, at any time, by invoking **pdb_truncate_xlog** at the Replication Agent administration port.

For more information on these properties, see the *Replication Agent Reference Manual*. For a more detailed description of truncating, see "Administering Replication Agent" in the *Replication Agent Administration Guide*.

Replication Agent for Microsoft SQL Server

Review the features of Replication Agent that are unique to Replication Agent for Microsoft SQL Server.

The term "Replication Agent for Microsoft SQL Server" refers to an instance of Replication Agent software that is installed and configured for a primary database that resides in a Microsoft SQL Server data server.

Note: For information on the basic features and operation of Replication Agent, see the *Replication Agent Administration Guide* and *Replication Agent Reference Manual*.

Microsoft SQL Server-Specific Considerations

These general issues and considerations are specific to using Replication Agent with the Microsoft SQL Server data server.

Replication Agent for Microsoft SQL Server reads the Microsoft SQL Server primary database log. To read the database log, Replication Agent must be installed where it can directly access the log files. Because the machine on which Replication Agent is installed must be of the same hardware and operating system as the machine on which the primary database resides, Replication Agent for Microsoft SQL Server is available only on the Windows platform. The term “Windows” refers to all supported Microsoft Windows platforms. For a complete list of supported platforms, see the *Replication Agent Installation Guide*.

Microsoft SQL Server Requirements

Observe these requirements for Microsoft SQL Server.

- You cannot simultaneously use Microsoft replication and Replication Agent on the same Microsoft SQL Server database. Disable Microsoft replication before using Replication Agent for Microsoft SQL Server.
- You must disable Change Data Capture for Microsoft SQL Server (all versions).
- The Microsoft SQL Server TCP/IP protocol must be enabled.

Microsoft SQL Server Restrictions

Microsoft SQL Server imposes these restrictions as the primary database with Replication Agent.

- Use of the **TRUNCATE TABLE** command on a table that has been marked for replication is forbidden.

- Dropping a stored procedure that has been marked for replication is forbidden.

Unsupported Software Features

Features that are not supported for Microsoft SQL Server data replication.

- Microsoft SQL Server virtual computed columns
- Replication Server warm standby (for non-Adaptive Server Enterprise databases)
- Replication Server **rs_subcomp** utility (for non-Adaptive Server Enterprise databases)
- Database level materialization

  Note: Direct load type of automatic materialization and bulk materialization (database level/manual) are supported.
- Some Microsoft SQL Server 2008 features

Replication Agent does not support these Microsoft SQL Server 2008 features.

- Column sets
- **MERGE SQL** statements
- Procedures with table-valued parameters
- Sparse columns
- Transparent data encryption (TDE)

A table or stored procedure that uses these features cannot be marked, even by using **pdb_setreptable** with the **force** keyword.

Replication of Deferred Updates on Primary Keys

Updates to the unique column index in a table is not supported by traditional replication, and the Replication Server reports errors.

The replication of updates to the unique column index in a table is not supported, and Replication Server reports errors. For example, table `t` has a unique index on column `c`, with values 1, 2, 3, 4 and 5. A single **update** statement is applied to the table:

```
update t set c = c+1
```

Using traditional replication, this statement results in:

```
update t set c = 2 where c = 1
update t set c = 3 where c = 2
update t set c = 4 where c = 3
update t set c = 5 where c = 4
update t set c = 6 where c = 5
```

The first update attempts to insert a value of c=2 into the table. However, this value already exists in the table. Replication Server displays error 2601—an attempt to insert a duplicate key.

The Replication Server DSI stops working if you attempt to replicate updates to a non-SAP table that has a unique column index. To work around this problem, broaden the unique index definition.

Unsupported Datatypes

Datatypes that are not supported for Microsoft SQL Server data replication.

- `cursor`
- `table`
- `xml`

Replication Agent does not support these Microsoft SQL Server 2008 datatypes.

- `datetimeoffset`
- `filestream`
- `geography`
- `geometry`
- `hierarchyid`
- Large user-defined datatypes

Tables containing columns of unsupported types can be marked by using **pdb_setreptable** with the **force** keyword, but replication might fail at runtime time when parsing log of such tables.

Applying Microsoft SQL Server Patches

Apply Microsoft SQL Server patches on a database that is being replicated.

1. Be sure that all data has been replicated to the replicate site.

   Note: All activities must stop before this step, and all users except the **pds_username** must log off from the primary database.

   For each existing Replication Agent for Microsoft SQL Server instance, verify that it is in *Replicating* state and allow replication to finish. To verify that replication has finished, quiesce the Replication Agent instance by issuing the **quiesce** command.

   Note: It may take a while for the command to return because Replication Agent reads all data from the log file and sends it to the Replication Server.

2. Disable the Replication Agent triggers.

   If the **pdb_automark_tables** configuration parameter is set to true, log on to the primary database and disable the automark trigger by issuing:

   ```
   DISABLE TRIGGER ra_createtable_trig_ ON DATABASE
   ```

   where `ra_createtable_trig_` is the name of the automark trigger created by Replication Agent.

   Note: The trigger name is based on the prefix and suffix setting for database object name.

3. Apply the service patch using the instructions in the Microsoft documentation.

4. Regenerate the objects in the Microsoft SQL Server system resource database.
   - Restart Microsoft SQL Server in single-user mode by opening a new command window and executing:

     ```
     "C:\Program Files\Microsoft SQL
     Server\MSSQL.1\MSSQL\Binn\sqlservr.exe" -m
     -sserverName\instanceName
     ```

     where *instanceName* is the name of the Microsoft SQL Server instance.
   - Log in to the Replication Agent instance:

     ```
     isql -U username -P password -S instanceName
     ```
   - Reinitialize Microsoft SQL Server :

     ```
     server_admin remove, force
     go
     server_admin init
     go
     ```
   - Restart Microsoft SQL Server in multiuser mode.
5. If the **pdb_automark_tables** configuration parameter is set to true before applying the patch:
   - Log in to the primary database, and enable the automark trigger by issuing:

     ```
     ENABLE TRIGGER ra_createtable_trig_ ON DATABASE
     ```

     where **ra_createtable_trig_** is the name of the automark trigger created by Replication Agent.
   - Log on to the primary database, and enable the DDL trigger by issuing:

     ```
     ENABLE TRIGGER ra_ddl_trig_ ON DATABASE
     ```

     where **ra_ddl_trig_** is the name of the DDL trigger created by Replication Agent.
6. Zero the LTM locator, and move the truncation point to the end of the log:
   - Zero the LTM locator by logging in to RSSD and issuing:

     ```
     rs_zeroltm ra_instance, pdb_name
     ```
   - Move the truncation point to the end of log by logging in to Replication Agent and issuing:

     ```
     ra_locator move_truncpt
     ```
7. Resume replication or other operations in Replication Agent.

DDL Replication

Replication of data definition language (DDL) commands is supported, but only to to Microsoft SQL Server databases.

Note: No translation or adjustment of DDL commands is provided by Replication Agent. DDL commands should therefore be replicated only to other Microsoft SQL Server databases.

Replication of DDL commands is enabled or disabled in Replication Agent using the **pdb_setrepddl** command. Replication Server uses the **ddl_username** parameter to execute DDL commands in the replicate database as the same user who executed the DDL commands in the primary database.

See *Replication Agent Reference Manual > Command Reference > pdb_setrepddl* and *Replication Agent Reference Manual > Configuration Parameters > ddl_username* for details on using **pdb_setrepddl** and **ddl_username**.

DDL parameters

To replicate DDL in Microsoft SQL Server, in addition to setting the value of **pdb_setrepddl** to enable, set the Replication Agent **ddl_username** and **ddl_password** parameters.

The **ddl_username** parameter is the replicate database user name included in LTL for replicating DDL commands to the replicate or target database.

Note: You do not need to set the **ddl_username** and **ddl_password** parameters if the Replication Server parameter **dsi_replication_ddl** is **on**.

Permissions

In addition to the permission to execute all replicated DDL commands at the replicate database, the **ddl_username** must also have the `impersonate` permission granted for all users whose DDL commands may be replicated to the replicate database. This `impersonate` permission is necessary to switch session context in the replicate database when executing a DDL command. This user switches context to apply the DDL command using the same privileges and default schema settings as the user who executed the DDL command at the primary database. To provide this context switch, the **ddl_username** user must have permission to execute the **execute as user** Microsoft SQL Server command for any user who might execute DDL commands to be replicated from the primary database.

For example, `user1` with a default schema of `schema1` executes this DDL command at the primary database:

```
create table tab1 (id int)
```

This results in the creation of a table named `schema1.tab1` at the primary database. At the replicate database, `user2` with a default schema of `schema2`, cannot immediately execute this DDL because it generates a table named `schema2.tab1`. Therefore, `user2`, whose name is specified by the **ddl_username** configuration parameter, must impersonate `user1` by issuing this command at the replicate database:

```
execute as user = 'user1'
```

The DDL can then be executed with the correct schema by `user2` at the replicate database, generating a table named `schema1.tab1`.

See the *Replication Agent Reference Manual*.

Impersonate Permission

There are two ways to grant `impersonate` permission to the **ddl_username** user:

- You can grant database owner permission to the to the **ddl_username** user. In doing this, you implicitly grant `impersonate` permission.
- Alternately, you can grant `impersonate` permission explicitly:

  ```
  GRANT IMPERSONATE ON USER::user1 TO ddl_user
  ```

  where *user1* is a user whose DDL is expected to be replicated to the replicate database, and *ddl_user* is the **ddl_username** user.

Note: This grant command must be executed in the replicate database, where the user defined to **ddl_username** executes the DDL commands.

When you replicate DDL in Microsoft SQL Server, use Microsoft SQL Server as the replicate database. You cannot replicate DDL commands from Microsoft SQL Server to non-Microsoft SQL Server replicate databases.

Note: To replicate DDL, Replication Server must have a database-level replication definition with **replicate DDL** set in the definition. See the *Replication Server Reference Manual*.

DDL Commands and Objects Filtered from Replication

These database-scope DDL commands are not replicated.

- **ALTER_APPLICATION_ROLE**
- **ALTER_ASSEMBLY**
- **ALTER_AUTHORIZATION_DATABASE**
- **ALTER_CERTIFICATE**
- **CREATE_APPLICATION_ROLE**
- **CREATE_ASSEMBLY**
- **CREATE_CERTIFICATE**
- **CREATE_EVENT_NOTIFICATION**
- **DROP_EVENT_NOTIFICATION**

These server-scope DDL commands are not replicated:

- **ALTER_AUTHORIZATION_SERVER**
- **ALTER_DATABASE**
- **ALTER_LOGIN**
- **CREATE_DATABASE**
- **CREATE_ENDPOINT**
- **CREATE_LOGIN**
- **DENY_SERVER**
- **DROP_DATABASE**

- **DROP_ENDPOINT**
- **DROP_LOGIN**
- **GRANT_SERVER**
- **REVOKE_SERVER**

Any object owned by users defined in the list of nonreplicated users is not replicated. You can modify this list using the **pdb_ownerfilter** command. In addition, SAP has provided a default list of owners whose objects are not replicated. Use the **pdb_ownerfilter** command to return, add, or remove the list of owners whose objects are not replicated. See the *Replication Agent Reference Manual*.

Replication Agent Connectivity

Replication Agent for Microsoft SQL Server uses JDBC™ for communications with all replication system components.

The Microsoft SQL Server JDBC driver must be installed on the Replication Agent host machine, and the JAR file path must be set in the CLASSPATH environment variable.

See the *Replication Agent Installation Guide* for the specific JDBC driver and version to install.

Replication Agent Permissions and Roles

Replication Agent for Microsoft SQL Server must create database objects to assist with replication tasks in the primary database.

These permissions must be granted to the user specified by the **pds_username** parameter:

- **create table** – required to create tables in the primary database.
- **create trigger** – required to create DDL triggers in the primary database.
- **create procedure** – required to create procedures in the primary database.

The user specified by the **pds_username** parameter must be added to these roles in the primary database:

- **db_owner** – required to allow Replication Agent to execute **sp_repltrans** and **sp_repldone** in the primary database. This role is also required for primary database initialization.
- **sysadmin** – required for Microsoft SQL Server data server initialization and deinitialization (using **ra_admin init** and **ra_admin deinit**, respectively).

The sybfilter Driver

Replication Agent must be able to read the Microsoft SQL Server log files. However, the Microsoft SQL Server process opens these log files with exclusive read permission, and the file cannot be read by any other processes, including Replication Agent. Before Replication Agent can replicate data, you must use the sybfilter driver to make the log files readable.

See also

- *sybfilter Driver Reference* on page 163

The sp_SybTruncateTable Stored Procedure

The **sp_SybTruncateTable** stored procedure allows you to truncate tables marked for replication from SAP enterprise resource planning (ERP) or any applications by overriding the Microsoft SQL Server limitation for truncation of tables.

The SAP ERP system that uses Replication Agent for Microsoft SQL Server for replication requires the latest patch of the SAP Kernel that contains the **sp_SybTruncateTable** stored procedure. The SAP ERP applications require the **truncate table** operations to replicate the tables.

See *SAP Note 1972365 – Retry on error 4711*: *https://css.wdf.sap.corp/sap/support/notes/1972365* for the SAP Kernel download and installation instructions.

When you issue **sp_SybTruncateTable**, the stored procedure turns off the replication flag for the marked tables temporarily, issues the **truncate table** command, and then turns on the replication flag.

You must run the **sp_SybTruncateTable** stored procedure for any applications, which requires **truncate table** operations against tables replicated by Replication Agent for Microsoft SQL Server. For example:

```
sys.sp_SybTruncateTable @tabname sysname(280)
```

where *tabname* refers to the table name that is truncated.

See *SAP Business Suite Database as a Primary Data Server* in the *SAP Replication Server Heterogeneous Replication Guide*.

Initialization of the Primary Data Server and Replication Agent

For Microsoft SQL Server initialization, Replication Agent for Microsoft SQL Server installs objects at both the data server and database level.

Data-server-level modifications are only required once. However, to make the server-level modifications, additional permissions are required, the **pds_dac_port_number** parameter is used, and the primary database must be in standalone mode. Subsequent executions of **ra_admin init** do not modify the server again and do not require the additional permission or configurations.

First-Time Initialization

You must initialize the primary Microsoft SQL Server so that Replication Agent can open the supplemental log of a table or procedure that is marked for replication. Do this only once for each primary data server.

When initializing the primary data server and Replication Agent for the first time:

1. Stop the Microsoft SQL Server Analysis Service. In **Control Panel** > **Administrative Tools** > **Services**, find the service named SQL Server Analysis Services. Stop this service.
2. Make sure Microsoft SQL Server allows a remote dedicated administrative connection (DAC):

   ```
   sp_configure 'remote admin connections', 1
   GO
   RECONFIGURE
   GO
   ```

   To execute **sp_configure** with both parameters to change a configuration option or to run the **RECONFIGURE** statement, you must be granted the **ALTER SETTINGS** server-level permission. This permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.
3. Determine the primary Microsoft SQL Server DAC port number.
   a) Open the `ERRORLOG` file in a text editor. This file is located in the log directory of your Microsoft SQL Server. For example,

      ```
      C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG
      \ERRORLOG
      ```

   b) Search for the string “Dedicated admin” to find an entry similar to:

      ```
      2007-11-09 13:40:02.40 Server Dedicated admin
      connection support was established for listening
      locally on port 1348.
      ```

   c) Record the port number specified in this entry.
4. Update your `sql.ini` file with the instance name and port number of your Replication Agent instance.
5. Log in to your Replication Agent, and set the **pds_dac_port_number** configuration parameter:

   ```
   ra_config pds_dac_port_number, port
   ```

   where *port* is the DAC port number you recorded.
6. Also configure these Replication Agent connectivity parameters for the Microsoft SQL Server primary database:
   - **pds_server_name**
   - **pds_database_name**
   - **pds_username**
   - **pds_password**
   - **pds_port_number**

   For information about these configuration parameters, see the *Replication Agent Installation Guide* and *Replication Agent Reference Manual*.
7. Stop the Microsoft SQL Server service.

a) In **Control Panel** > **Administrative Tools** > **Services**, find the service named SQL Server (*SERVER*), where *SERVER* is the name of your Microsoft SQL Server data server.
b) Stop this service.

8. Open a command window, and restart Microsoft SQL Server in single-user mode.

   For example:

   ```
   "C:\Program Files\Microsoft SQL
   Server\MSSQL.1\MSSQL\Binn\sqlservr.exe" -m -s
   instanceName
   ```

   Note: The directory path may vary depending on the version of Microsoft SQL Server.

   where *instanceName* is the name of the Microsoft SQL Server instance.

9. Make sure that there are no other connections to the primary database, and verify that Replication Agent can connect to the primary database.
   a) Log in to the Replication Agent instance:

      ```
      isql -U username -P password -S instanceName
      ```

      where *username*, *password*, and *instanceName* are your user ID, password, and Replication Agent instance name.
   b) Issue:

      ```
      test_connection PDS
      ```

10. Initialize the Microsoft SQL Server data server and Replication Agent:

    ```
    server_admin init
    ra_admin init
    ```

    In the primary database, Replication Agent creates tables, procedures, and triggers. The **sp_SybSetLogforReplTable**, **sp_SybSetLogforReplProc**, **sp_SybSetLogforLOBCol**, and **sp_SybTruncateTable** procedures are created in the `mssqlsystemresource` database with execute permission granted to Public.

11. Stop the Microsoft SQL Server service again by using **CTRL+C** at the command prompt. Alternatively, at the command prompt, run the **SHUTDOWN** command.
12. Restart Microsoft SQL Server in multiuser mode (normal start).
    a) In **Control Panel** > **Administrative Tools** > **Services**, find the service named SQL Server (*SERVER*), where *SERVER* is the name of your Microsoft SQL Server data server.
    b) Start this service.

       Start other Microsoft SQL Server services, such as Microsoft SQL Server Agent service or the Microsoft SQL Server Analysis Service.

See also

- *Replication Agent Objects in the Microsoft SQL Server Primary Database* on page 112

Subsequent Initialization

After you have initialized Replication Agent for the first time and have subsequently deinitialized Replication Agent using **ra_admin deinit**, you may want to reinitialize this Replication Agent instance or another Replication Agent instance for a different database in the same primary data server.

When initializing a Replication Agent instance subsequent to the first-time initialization:

1. Determine the primary Microsoft SQL Server DAC port number, and make sure Microsoft SQL Server allows a remote DAC:

   ```
   sp_configure 'remote admin connections', 1
   GO
   RECONFIGURE
   GO
   ```

   To execute **sp_configure** with both parameters to change a configuration option or to run the **RECONFIGURE** statement, you must be granted the **ALTER SETTINGS** server-level permission. The **ALTER SETTINGS** permission is implicitly held by the **sysadmin** and **serveradmin** fixed server roles.
2. Log in to your Replication Agent, and set the **pds_dac_port_number** configuration parameter.
3. Configure these Replication Agent connectivity parameters for the Microsoft SQL Server primary database:

   - **pds_server_name**
   - **pds_database_name**
   - **pds_username**
   - **pds_password**

   For information about these configuration parameters, see the *Replication Agent Installation Guide* and *Replication Agent Reference Manual*.
4. Verify that Replication Agent can connect to the primary database:

   ```
   test_connection PDS
   ```
5. Initialize the Microsoft SQL Server data server and Replication Agent:

   ```
   ra_admin init
   ```

Final Cleanup

After you have removed all Replication Agent objects from all the databases on a given primary data server by issuing **ra_admin deinit** in each database in which you had issued **ra_admin init**, you may want to remove all the remnants of Replication Agent and completely clean the primary data server.

To clean up all Replication Agent remnants from the primary data server:

1. Stop the Microsoft SQL Server service.
   a) In **Control Panel** > **Administrative Tools** > **Services**, find the service named SQL Server (*SERVER*), where *SERVER* is the name of your Microsoft SQL Server data server.
   b) Stop this service.
2. Open a command window, and restart Microsoft SQL Server in single-user mode:

   ```
   "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn
   \sqlservr.exe" -m -s instanceName
   ```

   where *instanceName* is the name of the Microsoft SQL Server instance.
3. Make sure the Microsoft SQL Server SQL Browser service is running, and connect to the data server using the **sqlcmd** utility with **-A** option or using the Management Studio. Specify the server name as `Admin:` *servername*, where *servername* is the name of your data server.
4. Remove the `pds_username` user if it has been created for Replication Agent:

   ```
   drop user pds_username
   ```
5. Remove the special marking and truncating procedures from the `mssqlsystemresource` database:

   ```
   drop procedure sp_SybSetLogforReplTable;
   drop procedure sp_SybSetLogforReplProc;
   drop procedure sp_SybSetLogforLOBCol;
   drop procedure sp_SybTruncateTable;
   ```
6. Stop Microsoft SQL Server in single-user mode by shutting down the Windows service or by issuing the **shutdown** command with the **sqlcmd** utility.
7. To undo the effects of the sybfilter driver on each of the log devices, remove the log path entry by editing the configuration file or by using the sybfilter manager console.
8. Restart Microsoft SQL Server in multiuser mode (normal start).
   a) In **Control Panel** > **Administrative Tools** > **Services**, find the service named SQL Server (*SERVER*), where *SERVER* is the name of your Microsoft SQL Server data server.
   b) Start this service.

See also

- *sybfilter Driver Reference* on page 163

Character Case of Database Object Names

Database object names must be delivered to the primary Replication Server in the same format as specified in replication definitions; otherwise, replication fails. For example, if a replication definition specifies a table name in all lowercase, then that table name must appear in all lowercase when it is sent to the primary Replication Server by the Replication Agent.

To control the way Replication Agent treats the character case of database object names sent to the primary Replication Server, set the **ltl_character_case** configuration parameter to one of these values:

- **asis** – (the default) database object names are passed to Replication Server in the same format as stored in the primary data server.
- **lower** – database object names are passed to Replication Server in all lowercase, regardless of how they are stored in the primary data server.
- **upper** – database object names are passed to Replication Server in all uppercase, regardless of how they are stored in the primary data server.

In Microsoft SQL Server, database object names are stored in the same case as entered (uppercase or lowercase).

Format of Origin Queue ID

Each record in the transaction log is identified by an origin queue ID that consists of 64 hexadecimal characters (32 bytes). The format of the origin queue ID is determined by the Replication Agent instance and varies according to the primary database type.

Table 14. Replication Agent for Microsoft SQL Server Origin Queue ID Format

Character	Bytes	Description
0–3	2	Database generation ID
4–11	4	Virtual file sequence number
12–19	4	Page start offset
20–23	2	Operation number
24–31	4	Available for specifying uniqueness
32–39	4	Oldest active transaction: virtual file sequence number
40–47	4	Oldest active transaction: page start offset
48–51	2	Oldest active transaction: operation number
52–59	4	Latest committed transaction: page start offset
60–63	2	Latest committed transaction: operation number

Microsoft SQL Server Datatype Compatibility

Replication Agent processes Microsoft SQL Server transactions and passes transaction information to the primary Replication Server. The primary Replication Server uses the

datatype formats specified in the replication definition to receive the data from Replication Agent.

Table 15. Microsoft SQL Server to SAP Replication Server Default Datatype Mapping

Microsoft SQL Server Datatype	Microsoft SQL Server Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`bigint`	-2^{63} to $2^{63} - 1$	`rs_msss_bigint`	-2^{63} to $2^{63} - 1$	
`binary`	Fixed-length up to 8000 bytes	`binary`	32K	
`bit`	Integer with value of 0 or 1	`bit`	Integer with value of 0 or 1	
`char`	Fixed-length up to 8000 characters	`char`	32K	
`date`	01/01/0001 to 12/31/9999	`date`	01/01/0001 to 12/31/9999	
`datetime`	Date and time from 01/01/1753 to 12/31/9999	`rs_msss_datetime`	Date and time from 01/01/1753 to 12/31/9999	
`datetime2`	Date and time from 01/01/1753 to 12/31/9999	`rs_msss_datetime2`	Date and time from 01/01/1753 to 12/31/9999	
`datetimeoffset`	01/01/0001 to 12/31/9999	`rs_msss_datetimeoffset`	01/01/0001 to 12/31/9999	
`decimal`	Decimal type with precision between 1 and 38 and scale from 1 to 38	`decimal`	Numeric from -10^{38} to $10^{38} - 1$	

Microsoft SQL Server Datatype	Microsoft SQL Server Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`float`	Floating point number -1.79E + 308 through -2.23E - 308, 0 and 2.23E + 308 through 1.79E + 308.	`float`	Floating precision from -1.79E + 308 to 1.79E + 308	Results are machine dependent.
`image`	Variable-length binary data up to 2^{31} - 1 bytes	`image`	2GB	
`int`	-2^{31} to 2^{31} - 1	`rs_msss_bigint`	-2^{31} to 2^{31} - 1	
`money`	Monetary from -2^{63} to 2^{63} - 1	`money`	Monetary from -2^{63} to 2^{63} - 1	
`nchar`	Fixed-length Unicode up to 4000 characters	`unichar` or `char`	32K	Actual maximum length is @@ncharsize * number of characters.
`ntext`	Variable-length Unicode up to 2^{30} - 1 characters	`unitext` or `image`	2GB	For Replication Server 15.0 and later versions, `ntext` maps to `unitext`. For earlier versions of Replication Server, `ntext` maps to `image`.

Microsoft SQL Server Datatype	Microsoft SQL Server Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`nvarchar`	Variable-length Unicode up to 4000 characters	`unichar` or `varchar`	32K	Actual maximum length is @@ncharsize * number of characters.
`nvarchar(max)`	Variable-length Unicode up to 2^{30} - 1 characters	`rs_msss_nvarchar_max`	2GB	The `nvarchar(max)` datatype cannot be replicated to data servers other than Microsoft SQL Server. For Replication Server 15.0 and later versions, `nvarchar(max)` maps to `unitext`. For earlier versions of Replication Server, `nvarchar(max)` maps to `image`.
`numeric`	Synonym for `decimal` datatype	`numeric`	Synonym for `decimal` datatype	
`real`	Floating point number from -3.40E + 38 to 3.40E + 38	`real`	Floating precision from -3.40E + 38 to 3.40E + 38	Results are machine dependent.

Microsoft SQL Server Datatype	Microsoft SQL Server Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`smalldatetime`	Date and time from 01/01/1900 to 06/06/2079	`rs_msss_datetime`	Date and time from 01/01/1900 to 06/06/2079	
`smallint`	Integer with value from -2^{15} to $2^{15} - 1$	`smallint`	Integer with value from -2^{15} to $2^{15} - 1$	
`smallmoney`	Monetary from -214,748.3648 to 214,748.3647	`smallmoney`	Monetary from -214,748.3648 to 214,748.3647	
`text`	Variable-length up to $2^{31} - 1$ characters	`text`	2GB	
`time`	00:00:00.0000000 to 23:59:59.9999999	`rs_msss_time`	00:00:00.0000000 to 23:59:59.9999999	
`timestamp`	Database-wide unique number	`timestamp` or `varbinary(8)`	Database-wide unique number	For replication to Replication Server 15.0 and earlier versions, the SAP datatype should be `varbinary(8)`. For replication to Replication Server 15.1 or later, the SAP datatype should be `timestamp`.

Microsoft SQL Server Datatype	Microsoft SQL Server Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`tinyint`	Integer with value from 0 to 255	`tinyint`	Integer with value from 0 to 255	
`uniqueidentifier`	Globally unique identifier	`char(36)`	Globally unique identifier	No SAP equivalent. Map to `char(38)`.
`varbinary`	Variable-length up to 8000 bytes	`varbinary`	32K	
`varbinary(max)`	Variable-length up to 2^{31} - 1 bytes	`rs_msss_varbinary_max`	2GB	The `varbinary(max)` datatype cannot be replicated to data servers other than Microsoft SQL Server.
`varchar`	Variable-length up to 8000 characters	`varchar`	32K	
`varchar(max)`	Variable-length up to 2^{31} - 1 characters	`rs_msss_varchar_max`	2GB	The `varchar(max)` datatype cannot be replicated to data servers other than Microsoft SQL Server.

Replication Server 15.0 Unsigned Datatype Mapping

For Replication Server 15.0 and later, unsigned datatypes are supported and can be specified in the replication definitions.

Table 16. Unsigned Integer Replication Definition Datatype Mapping

RepServer 15.0 Unsigned Datatypes	Replication Definition Datatypes
`unsigned bigint`	`numeric (20)`
`unsigned int`	`numeric (10)`
`unsigned smallint`	`int`
`unsigned tinyint`	`tinyint`

ntext Datatype Replication

Microsoft SQL Server stores double-byte `ntext` datatype values in little-endian byte order. By default, the byte order of `ntext` data is converted during replication to big-endian so that the data may be transmitted over networks using big-endian, which is the common network byte order.

To support replicating `ntext` data to a Microsoft SQL Server (or other replicate server that does not provide the necessary conversion), you may force the byte order to be sent using the **lr_ntext_byte_order** property by specifying a value of **big** (for big-endian) or **little** (for little-endian) as desired to meet the expectations of your replicate database.

The **lr_ntext_byte_order** parameter is available for Microsoft SQL Server, and Oracle and is especially important for replication between two different database types and between databases that reside on different platforms. For example, for replication between two Microsoft SQL Server databases, both the primary and replicate database store data in little-endian byte order because Microsoft SQL Server runs only on Windows. Therefore, the **lr_ntext_byte_order** parameter should be set to **little**. However, if the replicate database is not a Microsoft SQL Server, determine its byte order and set the **lr_ntext_byte_order** parameter accordingly.

Note: The default behavior of Replication Agent for Microsoft SQL Server is to force any Unicode data to big-endian order as defined by the **ltl_big_endian_unitext** configuration property. To allow the **lr_ntext_byte_order** configuration property to successfully override the Microsoft SQL Server byte order, you must also set **ltl_big_endian_unitext** configuration property to false whenever the **lr_ntext_byte_order** property is used.

The **ltl_big_endian_unitext** parameter specifies whether `unitext` data should be converted from little-endian to big-endian before sending LTL to Replication Server. Valid values are true and false. When setting this parameter, you must know how the **lr_ntext_byte_order** parameter is set. If the **lr_ntext_byte_order** parameter is set to send the correct byte order for

the replicate database, the **ltl_big_endian_unitext** parameter must be set to false so that the byte order is not changed.

The **ltl_big_endian_unitext** and **lr_ntext_byte_order** configuration properties have important differences. By default, the **ltl_big_endian_unitext** property is true. When the **ltl_big_endian_unitext** property is true, Replication Agent for Microsoft SQL Server ensures all Unicode data is sent in big-endian order. When the **ltl_big_endian_unitext** property is false, Replication Agent for Microsoft SQL Server allows Unicode data to be sent in whatever byte order is used when the data is stored in the transaction log file. The **lr_ntext_byte_order** property 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.

Alteration of Replication Definitions from the Primary Data Server

You can alter replication definitions from the primary data server.

To avoid having to quiesce the replication system before altering a replication definition, you can issue the Replication Server **alter replication definition** command from the primary data server and make schema changes to primary database objects at the same time. The propagation of changes to a replication definition can be automatically coordinated with data replication without having to stop the replication process.

To issue the Replication Server **alter replication definition** command from the primary data server, create a stored procedure named **rs_send_repserver_cmd** in the primary Microsoft SQL Server database. The SQL for creating this procedure is contained in the appropriate connection profile on Replication Server. For a list of connection profiles, use the Replication Server **admin show_connection_profiles** command.

For a full description of **rs_send_repserver_cmd** and the **alter replication definition** Replication Server command, see the *Replication Server Reference Manual*.

Security Considerations

When the **rs_send_repserver_cmd** procedure is invoked at the primary data server, Replication Agent passes corresponding Replication Command Language (RCL) directly to Replication Server. You should therefore consider carefully to whom execution privileges are assigned for the **rs_send_repserver_cmd** procedure, and assign privileges as appropriate for your environment and security policy.

Limitations

You cannot use the **rs_send_repserver_cmd** procedure to alter replication definitions for tables that contain columns of certain datatypes.

These types are:

- `nvarchar(max)`
- `varbinary(max)`
- `varchar(max)`

Note: If you manually change a table-level replication definition in Replication Server, you must then suspend and resume replication in the Replication Agent to ensure that the Replication Agent clears and refreshes its cache.

Replication Server set autocorrection Command

The Replication Server **set autocorrection** command prevents failures that would otherwise be caused by missing or duplicate rows in a replicated table.

The **set autocorrection** command corrects discrepancies that may occur during materialization by converting each **update** or **insert** operation into a **delete** followed by an **insert**.

In version 15.7.1 SP200, by default, Replication Agent autocorrects tables marked for replication when the **ra_autocorrect_on_mark** configuration parameter value is set to true. In earlier versions, you can set autocorrection from Replication Agent for one or all marked tables in the primary database by using the **ra_set_autocorrection** command as described in the *Replication Agent Reference Manual*. To set autocorrection from Replication Server, use the **set autocorrection** command in a replication definition. You must do this from Replication Server because Replication Agent cannot alter the autocorrection setting on a replication definition. See the *Replication Server Administration Guide*.

Computed Columns

Tables containing computed columns that are physically stored in a table—columns marked as PERSISTED in Microsoft SQL Server—can be marked for replication, and these columns will be replicated.

Tables containing virtual computed columns—computed columns that are not physically stored in a table—can be marked for replication, but these columns are not replicated. To maintain consistency between the primary and replicate databases for a marked table containing a virtual computed column, make sure that the expression defining the virtual computed column is the same in both the primary and replicate databases.

Replicate Microsoft SQL Server Large Object Datatypes into an SAP HANA Database

Replication Agent for Microsoft SQL Server allows to replicate of `varchar(max)`, `nvarchar(max)`, and `varbinary(max)` large object (LOB) datatypes into an SAP HANA® replicate database.

In Replication Agent for Microsoft SQL Server 15.7.1 SP100 and earlier versions, `varchar(max)`, `nvarchar(max)`, and `varbinary(max)` LOB datatypes were considered as a partial update, even for a full LOB update. Replication of partial updates to LOB datatypes was supported only in the Microsoft SQL Server to Microsoft SQL Server (homogeneous) replication environment.

With Replication Agent for Microsoft SQL Server 15.7.1 SP120, you can use the **lr_partial_updates_allowed** configuration parameter to determine whether replicate databases support partial updates to LOB datatypes.

By default, **lr_partial_updates_allowed** is true.

When the **lr_partial_updates_allowed** is true, Replication Agent sends changes to LOB datatype columns as partial updates to Replication Server. Replication Server sends partial updates to the replicate databases that have native support for partial LOB updates, such as Microsoft SQL Server.

If the replicate database do not support partial updates to LOB datatypes, set the **lr_partial_updates_allowed** to false.

When the **lr_partial_updates_allowed** is false, Replication Agent converts partial LOB updates to a full update, which is consumed in heterogeneous replication environments. If partial LOB updates are detected, an error exception occurs.

To replicate LOB datatypes from the Microsoft SQL Server to SAP HANA replicate database, set the **lr_partial_updates_allowed** parameter to false.

For Microsoft SQL Server LOB datatype mappings and limitations, see the *Replication Server Administration Guide: Volume1*.

Replication Agent Objects in the Microsoft SQL Server Primary Database

Replication Agent creates objects in the primary database to assist with replication tasks.

The Replication Agent objects are created by invoking the **ra_admin** command with the **init** keyword. When you invoke this command, Replication Agent generates a SQL script that contains the SQL statements for the objects created or modified in the primary database. This script is stored in the `partinit.sql` file in the `RAX-15_5\inst_name\scripts\xlog` directory. You must create these objects before marking any primary database objects for replication.

Note: The generated scripts are for informational purposes only. You cannot run them manually to initialize the primary database or Replication Agent.

See the *Replication Agent Administration Guide*.

Replication Agent Object Names

Replication Agent creates objects in the primary database to assist with replication tasks.

There are two variables in Replication Agent database object names:

- prefix – represents the one-to-three-character string value of the **ra_admin_instance_prefix** parameter (used for Replication Agent for Microsoft SQL Server or Replication Agent for UDB) or the **ra_admin_prefix** parameter.
- xxx – represents an alphanumeric counter, a string of characters that is (or may be) added to a database object name to make that name unique in the database.

The value of the **ra_admin_instance_prefix** parameter is the prefix string used in all Replication Agent object names.

The value of the **ra_admin_prefix_chars** parameter is a list of the nonalphanumeric characters allowed in the prefix string specified by **ra_admin_instance_prefix**. This list of allowed characters is database-specific. For example, in Oracle, the only nonalphanumeric characters allowed in a database object name are the $, #, and _ characters.

Use the **ra_admin** command to view the names of Replication Agent transaction log components in the primary database.

See the *Replication Agent Administration Guide* for details on setting up object names.

Table Objects

Replication Agent creates table objects in the Microsoft SQL Server primary database.

These tables are considered Replication Agent objects. Insert and delete permissions are granted to public only on the DDL shadow table. No permissions are granted on the other tables.

Table 17. Replication Agent Table Objects

Object	Name
DDL shadow table	*prefix*ddl_trig_*xxx*
Instance table	*prefix*instance_*xxx*
Object marking table	*prefix*markObject_*xxx*
Object verifying table	*prefix*checkObject_*xxx*

Microsoft SQL Server System Tables

At initialization, Replication Agent creates some system tables in the primary database using the Microsoft SQL Server **sp_replicationdboption** stored procedure.

These tables are also removed with **sp_replicationdboption** when the Replication Agent **ra_admin deinit** command is used. Do not modify these tables directly. For more information about **sp_replicationdboption**, see the Microsoft SQL Server documentation.

Procedure Objects

Replication Agent creates procedure objects in the Microsoft SQL Server primary database.

The table below lists the procedure objects that are considered Replication Agent objects. The **sp_SybSetLogforReplTable**, **sp_SybSetLogforReplProc**, and **sp_SybSetLogforLOBCol**

procedures are created in the Microsoft SQL Server `mssqlsystemresource` system database. Although execute permission on these procedures is granted to Public, only the Replication Agent `pds_username` user can successfully execute the procedures because only the `pds_username` user is granted **select** permission on the `sys.sysschobjs` table. No permissions are granted on the other procedures when they are created.

Note: The stored procedures listed here have no effect when executed outside the context of replication.

Table 18. Replication Agent Procedure Objects

Object	Name
Marks/unmarks an object	*prefix*mark_*xxx*
Verifies an object	*prefix*check_*xxx*
Retrieves the ID of the last committed transaction	*prefix*lct_sql_*xxx*
Marks/unmarks a table	sp_SybSetLogforReplTable
Marks/unmarks a procedure	sp_SybSetLogforReplProc
Marks/unmarks LOB column	sp_SybSetLogforLOBCol

Marker Objects

Replication Agent creates marker objects in the primary database.

These marker procedures and marker shadow tables are considered Replication Agent objects. No permissions are granted when these objects are created.

Table 19. Replication Agent Marker Objects

Object	Name
Transaction log marker procedure	rs_marker_*xxx*
Dump marker procedure	rs_dump_*xxx*
Transaction log marker shadow table	*prefix*markersh_*xxx*
Dump marker shadow table	*prefix*dumpsh_*xxx*

Trigger Objects

Replication Agent creates trigger objects in the Microsoft SQL Server primary database.

Table 20. Replication Agent Trigger Objects

Object	Name
Captures DDL commands	*prefix*ddl_trig_*xxx*
Captures **create_table** DDL commands	*prefix*createtable_trig_*xxx*

Transaction Log Administration

The only transaction log administration tasks required are backing up and truncating the transaction log.

Transaction Log Backup and Restoration

Replication Agent does not support backing up and restoring the transaction log automatically.

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

Transaction Log Truncation

Replication Agent supports both automatic and manual log truncation.

Replication Agent provides two options for automatic transaction log truncation:

- Periodic truncation, based on a time interval you specify
- Automatic truncation whenever Replication Agent receives a new LTM locator value from the primary Replication Server. You also have the option to switch off automatic log truncation. By default, automatic log truncation is based on LTM locator value from the primary Replication Server.

To configure Replication Agent log truncation, observe these guidelines:

- You can specify the automatic truncation option you want (including none) by using **ra_config** to set the value of the **truncation_type** configuration parameter.

  The valid values are:

  - **command** – Replication Agent truncates the transaction log only when the **pdb_truncate_xlog** command is invoked.
  - **locator_update** (default) – Replication Agent automatically truncates the transaction log whenever it receives a new LTM Locator value from the primary Replication Server.
  - **interval** – Replication Agent automatically truncates the transaction log when determined by a configurable interval of time. Interval is set in minutes (0 (never) to 720) in the **truncation_interval** parameter.

To truncate the transaction log automatically based on a time interval, use **ra_config** to set the value of the **truncation_interval** configuration parameter.

- You can truncate the Replication Agent transaction log manually, at any time, by invoking **pdb_truncate_xlog** at the Replication Agent administration port.

  To truncate the transaction log at a specific time, use a scheduler utility to execute the **pdb_truncate_xlog** command automatically.
- Replication Agent for Microsoft SQL Server truncates the primary database log in units of transactions. After Replication Agent for Microsoft SQL Server receives the LTM locator from Replication Server, Replication Agent for Microsoft SQL Server queries the primary database to obtain the transaction ID of the newest transaction that can be truncated. Replication Agent for Microsoft SQL Server then marks as reusable the transaction log space before the newest transaction. Microsoft SQL Server can then write log records into the reusable space.
- The **sp_repltrans** and **sp_repldone** Microsoft SQL Server commands are issued by Replication Agent to control log truncation within Microsoft SQL Server. These commands require that the Replication Agent user have the `db_owner` role permission.

Note: Microsoft SQL Server allows only one session to control log truncation using the **sp_repltrans** and **sp_repldone** commands. You should not use these commands while Replication Agent is controlling the log truncation processing.

For more information on these properties, see the *Replication Agent Reference Manual*. For a more detailed description of truncating, see "Administering Replication Agent" in the *Replication Agent Administration Guide*.

Using Windows Authentication with Microsoft SQL Server

When running Replication Agent for Microsoft SQL Server on a Windows platform, you have the option of configuring it to connect to Microsoft SQL Server using Windows credentials to authenticate the user.

To configure Replication Agent to use Windows authentication:

1. In your primary Microsoft SQL Server, add the user who will be starting Replication Agent, <*ra_user*>, as a Windows-authenticated user, including the user domain as appropriate. Add the <*ra_user*> to the primary database and grant the appropriate permissions. For additional information, see the Microsoft SQL Server documentation.
2. On the machine on which the Replication Agent for Microsoft SQL Server is running, add <*domain*>\<*ra_user*> to the Windows user account. If no domain exists, add only the <*ra_user*> to the Windows user account.
3. On the same machine, copy the `sqljdbc_auth.dll` file from the Microsoft SQL Server JDBC driver location to a directory on the Windows system path. When you installed the Microsoft SQL Server JDBC driver, the `sqljdbc_auth.dll` files were installed in this location:

```
<install_dir>\sqljdbc_<version>\<language>\auth\
```

Note: On a 32-bit processor, use the `sqljdbc_auth.dll` file in the x86 folder. On a 64-bit processor, use the `sqljdbc_auth.dll` file in the x64 folder.

4. On the same machine, login as the <*ra_user*> and start the Replication Agent for Microsoft SQL Server instance.
5. Log in to Replication Agent and configure these parameters using values appropriate for the primary Microsoft SQL Server:

```
ra_config pds_server_name, <server>
ra_config pds_port_number, <port>
ra_config pds_database_name, <database>
ra_config pds_username, <ra_user>
ra_config pds_integrated_security, true
```

6. Continue configuring and using Replication Agent as described in Replication Agent documentation.

Setting Up Replication Agent and Microsoft SQL Server on Different Machines

Run Replication Agent and the primary data server on different machines.

1. Install the sybfilter driver on the same machine as the primary Microsoft SQL Server, and use this driver to make the transaction logs readable for Replication Agent.
2. On the machine on which the primary Microsoft SQL Server is running, share the drive or drives containing the transaction log files so that the drives can be mounted on the machine on which Replication Agent is to be installed.
3. Install Replication Agent on a machine of the same hardware and operating system as the machine on which the primary data server is running.
4. Install the JDBC driver on the same machine as Replication Agent.
5. On the Replication Agent machine, map network drives that contain the primary Microsoft SQL Server database transaction log files. Use the **ra_devicepath** command to point Replication Agent to the log files.

Setting Up Replication Agent for Microsoft SQL Server on Windows Server Failover Clustering

Set up Replication Agent for Microsoft SQL Server on a shared disk and add it as a generic application on the Windows Server Failover Clustering (WFSC) cluster.

Prerequisites

- Before installing Replication Agent for Microsoft SQL Server, setup and configure the WSFC cluster and Microsoft SQL Server.

  See the *Windows Server Failover Clustering with SQL Server* documentation available at *http://technet.microsoft.com/en-us/library/hh270278.aspx* .
- Make sure at least one shared disk is available in the Microsoft SQL Server Cluster service for sharing data between Microsoft SQL Server and the Replication Agent instance.

  See the *Create a New SQL Server Failover Cluster* documentation available at *http://technet.microsoft.com/en-us/library/ms179530.aspx* .

Software Requirements

- Windows Server Failover Clustering (WSFC) on Windows Server 2008 R2
- Microsoft SQL Server 2008 R2
- Replication Agent 15.7.1 SP110 or later

Supported Configurations

Replication Agent 15.7.1 supports only the Instance-level High Availability cluster configuration of Microsoft SQL Server. Replication Agent does not support the Database-level High Availability (an availability group) cluster configuration.

Task

1. Install the Replication Agent on the primary cluster node.

   See the *Replication Agent Installation Guide* for the installation information.
2. Verify that the `%SYBASE%` installation directory is on a shared disk that is part of the Microsoft SQL Server Cluster service.
3. Create the sybfilter configuration file on a persistent system disk.
4. During the instance initialization, set **verify_sybfilter** to no in the resource file.
5. Complete these steps on the remaining cluster nodes:
   a) Set the `%SYBASE%` directory to the existing installation location on the shared disk manually.
   b) Install the SQL Server JDBC Driver and configure the `%CLASSPATH%`.

See *Installing the Microsoft SQL Server JDBC Driver* in the *Replication Server Options Quick Start Guide*.

c) Install and set up the sybfilter driver.

For the detailed instructions, see *Installing and Setting Up the sybfilter Driver* on page 163.

6. Add the Replication Agent instance as a 'Generic Application' to the Microsoft SQL Server Cluster service with this command:

```
%SYBASE%\RAX-15_5\bin\ra.bat

-i <instance_name> -replicate
```

where:

- `%SYBASE%` – is the Replication Agent installation directory.
- *instance_name* – is the name of your Replication Agent instance.
- *replicate* – is the name of your replicate Replication Server.

After running this command successfully, a generic application is added in the Microsoft SQL Server Cluster service.

7. To add a dependency to the 'SQL Server' for a generic application:

a) In the Microsoft SQL Server Cluster, right-click on the Generic Application and select **Properties**.

b) In the Properties dialog, select **Dependencies** tab, and click the **Insert** button.

c) Select **SQL Server** from the drop-down list, and click **OK**.

See also

- *Installing and Setting Up the sybfilter Driver* on page 163

Replication Agent for UDB

Review the features of Replication Agent that are unique to Replication Agent for UDB.

The term "Replication Agent for UDB" refers to an instance of Replication Agent software that is installed and configured for a primary database that resides in an IBM DB2 for Linux, Unix, and Windows server.

Note: For information on the basic features and operation of Replication Agent, see the *Replication Agent Administration Guide* and *Replication Agent Reference Manual*.

IBM DB2-Specific Considerations

These general issues and considerations are specific to using Replication Agent with the IBM DB2 for Linux, Unix, and Windows server.

Unsupported Software Features

Features that are not supported for IBM DB2 data replication.

- DB2 stored procedures
- Replication Server parallel DSI
- Replication Server **rs_init** utility
- Replication Server **rs_subcomp** utility
- Replication Server when replicating in an environment where other vendors are replicating

Replication of Deferred Updates on Primary Keys

Updates to the unique column index in a table is not supported by traditional replication, and the Replication Server reports errors.

The replication of updates to the unique column index in a table is not supported, and Replication Server reports errors. For example, table `t` has a unique index on column `c`, with values 1, 2, 3, 4 and 5. A single **update** statement is applied to the table:

```
update t set c = c+1
```

Using traditional replication, this statement results in:

```
update t set c = 2 where c = 1
update t set c = 3 where c = 2
update t set c = 4 where c = 3
update t set c = 5 where c = 4
update t set c = 6 where c = 5
```

The first update attempts to insert a value of c=2 into the table. However, this value already exists in the table. Replication Server displays error 2601—an attempt to insert a duplicate key.

The Replication Server DSI stops working if you attempt to replicate updates to a non-SAP table that has a unique column index. To work around this problem, broaden the unique index definition.

Unsupported Datatypes

Datatypes that are not supported for IBM DB2 data replication.

- `ROWID`
- `XML`
- User-defined datatypes

These datatypes are not supported when the replicate database is IBM DB2:

- `BLOB`
- `CLOB`
- `DBCLOB`
- `LONG VARCHAR`
- `LONG VARGRAPHIC`

Note:

If graphic datatypes are used in a non-unicode database, Replication Agent does not replicate the values in these columns with the correct charset encoding (except for UTF-16BE encoding).

Feature Differences in Replication Agent for UDB

These Replication Agent features have unique behavior in Replication Agent for UDB.

Initializing Replication Agent

The Replication Agent for UDB provides the same features for initializing Replication Agent and creating its objects in the primary database as other implementations of the Replication Agent. Replication Agent for UDB creates only a few tables in the primary database to store its system information.The Replication Agent for UDB creates four procedures in the primary IBM DB2 UDB database for the archive log related feature. For example:

- RA_GET_LOG_NAME_
- RA_GET_VERSION_STR_
- RA_TRUNC_LOG_FILES_
- RA_GET_TRUNC_VER_STR_

Because the Replication Agent for UDB requires access to the UDB transaction log, the user ID that the Replication Agent uses to access the primary database must have either **SYSADM**

or **DBADM** authority in the database; otherwise, the **ra_admin init** command returns an error. This user ID is stored in the Replication Agent **pds_username** configuration parameter.

Marking a Table for Replication

The Replication Agent for UDB provides the same features for marking and unmarking tables for replication as other implementations of the Replication Agent. However, the Replication Agent for UDB does not create any stored procedures or triggers in the primary database.

When marking a table for replication, Replication Agent for UDB alters the table to set the UDB **DATA CAPTURE** attribute to **DATA CAPTURE CHANGES**. When the table is unmarked, the table is altered to return to its original **DATA CAPTURE** attribute.

Note: Do not manually change the **DATA CAPTURE** attribute of a table that has been marked for replication by Replication Agent for UDB. Doing so may adversely effect replication results.

Unavailable Features

These Replication Agent features are not available with Replication Agent for UDB:

- Stored procedure replication–the replication of stored procedures is not available with the Replication Agent for UDB. Therefore, the **pdb_setrepproc** command is not supported.
- Altering replication definitions from the primary data server–this involves stored procedure replication, which is not supported.

Note: When you invoke Replication Agent commands related to these features, you receive an error.

See also

- *Replication Agent Objects in the DB2 Primary Database* on page 137

IBM DB2 Requirements

Observe these requirements for IBM DB2 for Linux, Unix, and Windows.

- The database must be version 9.1, 9.5, 9.7, 10.1, or 10.5.
- If you have a UDB client instance and a UDB server instance on different machines, the client and server must be of the same UDB version.
- The database must have a valid JDK path configured. The JDK_PATH configuration parameter must contain the full path to the directory above the `bin` directory, which contains the `java` executable. To determine the database manager JDK_PATH setting, use this DB2 command:

  ```
  get dbm cfg
  ```

  Note: A 64-bit IBM DB2 instance requires a 64-bit JDK, and a 32-bit DB2 instance requires a 32-bit JDK.

- If Replication Agent is installed on a Linux or UNIX host, you must configure a client or server 64-bit DB2 instance.
- The database LOGARCHMETH1 configuration parameter must be set to LOGRETAIN or DISK:*<path>*, where *<path>* is a directory to which logs are archived. This enables archive logging in place of circular logging. To determine the LOGARCHMETH1 setting, use this DB2 command:

  ```
  get db cfg for <db-alias>
  ```
- On a Windows system, the DB2 connectivity **autocommit** parameter must be turned on (**automcommit**=*1*). The **autocommit** parameter is specified in the DB2 call level interface (CLI) configuration file for the primary database. If the **autocommit** parameter is not turned on, a deadlock problem can occur. The path to the CLI configuration file is:

  %DB2DIR% `\sqllib\db2cli.ini`

  where *%DB2DIR%* is the path to the DB2 client installation.

  Alternatively, to turn on autocommit, open the DB2 administrative command line console and run:

  ```
  db2set DB2OPTIONS=+c
  ```
- To initialize Replication Agent without error, the database must have a tablespace created with these characteristics:
  - The tablespace should be a user temporary tablespace. By default, user temporary tablespaces are not created when a database is created.
  - The tablespace must be a system-managed space (SMS).
  - The **PAGESIZE** parameter must be set to 8192 (8 kilobytes) or greater.
- The user ID you specify as the **pds_username** user must have either SYSADM or DBADM authority to access the primary database transaction log.
- All the DB2 environment variables must be set before you start the Replication Agent. Replication Agent uses the DB2 CLI driver to connect to the primary DB2 database. For UNIX, the driver is contained in `libdb2.so`, `libdb2.sl`, or `libdb2.a`, depending on the operating system. For Windows, the DB2 driver is contained in `db2cli.dll`. Replication Agent also uses DB2 API libraries to read the transaction log. The library path environment variable must therefore be set for Replication Agent to load the correct driver and API libraries at runtime.

  For UNIX and Linux, the 64-bit versions of the libraries are located in the `$HOME/sqllib/lib64` directory, where `$HOME` is the home directory of the DB2 instance owner. If Replication Agent is installed on Linux or UNIX, the library path environment variable must point to the 64-bit libraries. For Windows, the library path environment variable must point to the 32-bit libraries.

  The exact name of the library path environment variable depends on the operating system. For Linux, the library path variable is named LD_LIBRARY_PATH. For Windows, the library path variable is named PATH.

  On Windows, the DB2 server or client installation sets all necessary environment variables. On UNIX or Linux, you must source the DB2 `db2cshrc` (for C-shell) or the `db2profile` (for Bourne and Korn shells) script before starting the Replication Agent.

These scripts are located at `$HOME/sqllib`, where `$HOME` is the home directory of the DB2 instance owner (for a DB2 client or server instance).

Java Heap Size

Make sure the size of the JVM heap is large enough for your primary DB2 UDB data server.

Set the DB2 UDB **java_heap_sz** configuration parameter to 2048 or a larger value.

Replication Agent and a DB2 Server on Different Machines

If the Replication Agent for UDB software is installed on a different host machine from the DB2 server, you must install the DB2 Administration Client on the same host machine as the Replication Agent.

If the Replication Agent for UDB software is installed on the same host machine as the DB2 server, a separate DB2 Administration Client is not required.

If the Replication Agent for UDB software is installed on Linux or UNIX, a 64-bit DB2 client instance must be configured. On Windows, a 32-bit DB2 client instance must be configured.

DB2 Connectivity

On a Windows system, you must configure a DB2 Universal Database JDBC data source in the DB2 Administration Client, then use the database name and database alias specified for that DB2 Universal Database JDBC data source when you configure Replication Agent for UDB connectivity.

On a UNIX system, instead of using ODBC, simply catalog the node and the primary database in DB2. Set the Replication Agent **pds_datasource_name** parameter to the database alias. Also set the **pds_host_name** and **pds_host_number**.

Cataloging the Remote TCP/IP Node from the DB2 Client

Catalog the remote DB2 client node.

1. Log in as the DB2 instance owner.

   Logging in sets up your DB2 environment variables by executing the environment scripts. You can also execute these scripts manually as follows.

   In Korn shell, source the `db2profile` file:

   ```
   . $HOME/sqllib/db2profile
   ```

   In C shell, source the `db2cshrc` file:

   ```
   source $HOME/sqllib/db2cshrc
   ```

   where *$HOME* is the home directory of the DB2 instance owner.
2. Start the DB2 command-line processor by typing the **db2** command.
3. Catalog the remote TCP/IP node using this command at the DB2 prompt:

   ```
   catalog tcpip node MYNODE remote MYHOST server XXXX
   ```

where *MYNODE* is the node name, *MYHOST* is the host name or IP address of the data server, and *XXXX* is the data server port number.

4. Verify the catalog entry:

```
list node directory
```

DB2 should return something similar to:

```
Node 1 entry:
      Node name            = MYNODE
      Comment              =
      Directory entry type = LOCAL
      Protocol             = TCPIP
      Hostname             = MYHOST
      Service name         = XXXX
```

Cataloging the Primary Database from the DB2 Client

Catalog the primary database.

1. Catalog the primary database using this command at the DB2 prompt:

```
catalog database MYDB as MYDB_ALIAS at node MYNODE
```

where *MYDB* is the database name, *MYDB_ALIAS* is an alias for the database, and *MYNODE* is the node name used in the **catalog tcpip node** command.

2. Verify the catalog entry:

```
list database directory
```

DB2 should return something similar to:

```
System Database Directory

Number of entries in the directory = 1

Database 1 entry:

Database alias         = MYDB_ALIAS
Database name          = MYDB
Node name              = MYNODE
Database release level = b.00
Comment                =
Directory entry type   = Remote
```

Configuring pds_datasource_name

Set the Replication Agent **pds_datasource_name** parameter.

1. In Replication Agent, set **pds_datasource_name** to the database alias:

```
ra_config pds_datasource_name, MYDB_ALIAS
```

where *MYDB_ALIAS* is the database alias that was used when cataloging the primary database.

2. Also set these Replication Agent parameters:

   - **pds_database_name**
   - **pds_username**
   - **pds_password**
   - **pds_host_name**
   - **pds_port_number**

   See the *Replication Agent Reference Manual*.

Replication Agent for UDB Connectivity Parameters

These Replication Agent configuration parameters are required to configure a connection between the Replication Agent for UDB and a DB2 server.

- **pds_username** – must have **DBADM** authority, for example, repuser.
- **pds_password** – for user ID specified in **pds_username,** for example, repuser_pwd.
- **pds_database_name** – DB2 database name, for example, TEST_DB1.
- **pds_datasource_name** – DB2 data source name, for example, TEST_DB1_DS.
- **pds_host_name** – name of the host on which the primary DB2 data server resides.
- **pds_port_number** – port number of the primary DB2 data server.

Repositioning in the Log

The Replication Agent uses the value of the LTM locator received from the primary Replication Server to determine where to begin looking in the DB2 transaction log for transactions to be sent to the Replication Server.

The Replication Agent for UDB uses the LTM locator value as follows:

- When the value of the LTM locator received from Replication Server and the LTM locator stored by Replication Agent are both zero (0), the Replication Agent positions the Log Reader component at the end of the DB2 transaction log.

  Warning! In the event that both LTM locator values are zero, two specific conditions may cause data loss:

  - When the Replication Agent Log Reader component goes to the *Replicating* state, it does so asynchronously. When you receive a prompt after invoking the **resume** command, the Log Reader component may not be finished getting into the *Replicating* state and positioning itself at the end of the log. If you mark a table immediately after the prompt returns from the **resume** command, the record containing the mark information may be written to the log before the Log Reader component has positioned itself. In that case, the Log Reader component misses that record and does not replicate any subsequent data for that table. To avoid this problem, wait a short time after invoking the **resume** command before you mark a table for replication.
  - If you mark a table for replication, insert data into the table, and then resume replication, the data is not replicated if the LTM locators for Replication Agent and Replication Server are zero (as they would be at the beginning of replication). This

problem occurs because when both LTM locators are zero, resuming replication repositions the Log Reader component at the end of the log, skipping over any previous transactions. To avoid this problem when the LTM locators for Replication Agent and Replication Server are zero, mark the table for replication after you have issued the resume command.

- When both the value of the LTM locator received from Replication Server and the LTM locator stored by Replication Agent are not zero, Replication Agent uses the LTM locator value it received from Replication Server to determine the starting position of the oldest open transaction and positions the Log Reader component at that location in the DB2 transaction log.
- When the value of the LTM locator received from Replication Server is 0 (zero) and the value of the LTM locator stored by Replication Agent is not zero, Replication Agent uses the LTM locator value it has stored to determine the starting position of the oldest open transaction and positions the Log Reader component at that location in the DB2 transaction log.

Replication Agent for UDB Behavior

These Replication Agent issues are unique to Replication Agent for UDB.

Table Marking Immediately After Resume When LTM Locator Is Zero

When the Replication Agent instance goes to *Replicating* state, the Log Reader component reads the primary database transaction log and uses the value of the origin queue ID to determine the position in the log to start reading. When the value of the LTM locator is 0 (zero), the Log Reader starts reading at the end of the log.

Because the Log Reader operation is asynchronous, the Replication Agent instance can return to the operating system prompt after the **resume** command but before the Log Reader has completed its start-up process. If you immediately invoke the **pdb_setreptable** command to mark a table for replication after the **resume** command returns, the mark object entry can be placed in the transaction log before the Log Reader finds the end of the log. In that event, the Log Reader misses the mark table entry, and table marking fails.

To avoid this problem, wait 5 to 10 seconds after invoking **resume** before invoking **pdb_setreptable** to mark a table.

*DB2 **FORCE APPLICATION** Command*

The DB2 **FORCE APPLICATION** command causes the data server to drop its connections with an application. **FORCE APPLICATION ALL** causes the data server to drop its connections with all applications.

If you invoke **FORCE APPLICATION** and specify either the Replication Agent application handle or the **ALL** keyword, the data server drops its connections with the Replication Agent instance. In that event, the Replication Agent receives DB2 error code -30081 and cannot recover, so the Replication Agent instance shuts itself down.

To avoid this situation, invoke the Replication Agent **shutdown** command before using **FORCE APPLICATION**.

Read Buffer Size

The Replication Agent for UDB Log Reader component uses the value of the **lr_read_buffer_size** parameter to determine the maximum number of bytes to be read from the transaction log during each scan. Because the Log Reader reads bytes, it requires a buffer to store the bytes read.

It is difficult to identify a minimum buffer size that always works. The value range of **lr_read_buffer_size** is 10000 to 2147483647. It is recommended that you set value of **lr_read_buffer_size** to the default value (64000).

If the read buffer size is too small to read one operation, Replication Agent goes into the *Admin* state, and the Log Reader component shuts down and reports the DB2 -2650 error. Unfortunately, this error message covers general communication errors, not just an insufficient buffer size.

LOB Replication

When replication is enabled for a LOB column, Replication Agent makes an entry in the *prefix*`vblob_columns_` table to support replication for that column.

When Replication Agent processes a transaction that affects a LOB column, the LOB data may not be stored in the transaction log because of its possible size. Instead, the Replication Agent Log Reader component reads the LOB data directly from the primary database at the time it processes the transaction.

Note: If you do not specify the primary key for a table that contains columns with approximate numeric datatypes, the LOB and long field columns may be replicated as NULL values.

For instructions on enabling and disabling replication for LOB columns, see the *Replication Agent Administration Guide*.

Transaction Integrity and LOB Data

Because of the way Replication Agent processes the LOB column data when replicating transactions, transaction integrity may be compromised. For example, if two transactions change the data in a LOB column and the Log Reader does not process the first transaction until after the second transaction has been committed, when the LOB data is read from the primary database, the value of that data is the result of the second transaction. In this event, the value of the LOB data in the first transaction is never sent to the replicate database. After the second transaction is processed by the Log Reader, the primary and replicate databases are synchronized again, but for a period of time between processing the first and second transactions, the replicate database contains data that does not match the originating transaction.

This problem occurs only when a LOB column is changed more than once by a sequence of transactions. The period of time over which the problem exists may be significant if the

replication system throughput is slow or if a replication system component fails. As soon as the last transaction that changes the LOB column is processed at the replicate site, the problem is corrected.

Character Case of Database Object Names

Database object names must be delivered to the primary Replication Server in the same format as specified in replication definitions; otherwise, replication fails. For example, if a replication definition specifies a table name in all lowercase, then that table name must appear in all lowercase when it is sent to the primary Replication Server by the Replication Agent.

To control the way Replication Agent treats the character case of database object names sent to the primary Replication Server, set the **ltl_character_case** configuration parameter to one of these values:

- **asis** – (the default) database object names are passed to Replication Server in the same format as stored in the primary data server.
- **lower** – database object names are passed to Replication Server in all lowercase, regardless of how they are stored in the primary data server.
- **upper** – database object names are passed to Replication Server in all uppercase, regardless of how they are stored in the primary data server.

In the DB2 server, database object names are stored in all uppercase.

Format of Origin Queue ID

Each record in the transaction log is identified by an origin queue ID that consists of 64 hexadecimal characters (32 bytes). The format of the origin queue ID is determined by the Replication Agent instance and varies according to the primary database type.

Table 21. Replication Agent for DB2 Origin Queue ID Format

Character	Bytes	Description
0–3	2	Database generation ID
4–19	8	Operation sequence number
20–35	8	Transaction ID
36–51	8	First operation sequence number of oldest active transaction
52–55	2	Operation type (begin = 0, data/LOB = 1, commit/rollback = 7FFF)
56–63	4	LOB sequence ID

DB2 Datatype Compatibility

Replication Agent for UDB processes transactions and passes data to the primary Replication Server. The primary Replication Server uses the datatype formats specified in the replication definition to receive the data from Replication Agent for UDB.

This table describes the default conversion of DB2 datatypes to SAP datatypes. For each datatype in this table, lengths in the second column are described as:

- Character datatypes – maximum number of bytes.
- Graphic datatypes – maximum number of characters.
- Numeric datatypes – range from smallest to largest values.
- Temporal datatypes – range from earliest time to latest time.

Table 22. DB2 to SAP Replication Server Default Datatype Mapping

DB2 Datatype	DB2 Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
`BIGINT`	-9,223,372,036, 854,775,808 to 9,223,372,036, 854,775,807	`bigint`	10^{-38} to 10^{38}, 38 significant digits	
`BLOB`	Variable-length, 2GB, binary data	`rs_udb_varchar_for_bit`	2GB	
`CHAR`	254 bytes	`char`	32K	
`CHAR FOR BIT DATA`	254 bytes, binary data	`rs_udb_char_for_bit` (recommended) `binary`	32K	
`CLOB`	Variable-length, 2GB, character data	`text`	2GB	

DB2 Datatype	DB2 Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
DATE	0001-01-01 to 9999-12-31	char, date, or datetime	32K (char)	Use the Replication Server heterogeneous datatype support (HDS) feature for datatype conversion and translation.
DBCLOB	Variable-length, 2GB, double-byte character data	unitext or image	2GB	For Replication Server 15.0 and later versions, DBCLOB maps to unitext. For earlier versions of Replication Server, DBCLOB maps to image.
DECFLOAT(16)	8 bytes, -9.9999999999999999 x 10^{384} to -1.0 x 10^{-383} and1.0 x 10^{-383} to 9.9999999999999999 x 10^{384}	rs_udb_decfloat	Precision and range corresponds to a C double datatype, approximately 16 significant digits	Some loss of precision may result.

DB2 Datatype	DB2 Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
DECFLOAT(34)	16 bytes, -9.999999999999999999999999999999999 x 10^{6144} to -1.0 x 10^{-6143} and 1.0 x 10^{-6143} to 9.999999999999999999999999999999999 x 10^{6144}	float	Precision and range corresponds to a C double datatype, approximately 16 significant digits	Some loss of precision may result.
DECIMAL	-10^{31}+1 to 10^{31}-1, 31 digits of precision	decimal	10^{-38} to 10^{38}, 38 significant digits	
DOUBLE				See FLOAT.
FLOAT	8 bytes, -1.79769^{308} to 1.79769^{308}	float	Precision and range corresponds to a C double datatype, approximately 16 significant digits	Extremely small values are truncated to 16 digits to the right of the decimal. Extremely large values retain their precision.
GRAPHIC	127 characters, double-byte character data	unichar	32K	
INTEGER	-2,147,483,648 to 2,147,483,647	rs_udb_bigint	-2,147,483,648 to 2,147,483,647	

DB2 Datatype	DB2 Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
LONG VARCHAR	Variable-length, 32,700 bytes, character data	text	2GB	
LONG VARCHAR FOR BIT DATA	32,700 bytes, binary data	image	2GB	
LONG VARGRAPHIC	16,350 characters, double-byte character data	unitext or image	2GB	For Replication Server 15.0 and later versions, LONG VARGRAPHIC maps to unitext. For earlier versions of Replication Server, LONG VARGRAPHIC maps to image.
NUMERIC (synonym for DECIMAL)				See DECIMAL.
REAL	-3.402^{38} to 3.402^{38}	decimal	10^{-38} to 10^{38}, 38 significant digits	
SMALLINT	-32,768 to 32,767	smallint	-32,768 to 32,767	
TIME	00:00:00 to 24:00:00	char, time, or datetime	32K (char)	

DB2 Datatype	DB2 Length/ Range	SAP Replication Server Datatype	SAP Replication Server Length/ Range	Notes
TIMESTAMP	0001-01-01-00.00.00.000000 to 9999-12-31-24.00.00.000000	char or datetime	32K (char)	Use the Replication Server heterogeneous datatype support (HDS) feature for datatype conversion and translation.
VARCHAR	32,672 bytes	varchar	32K	
VARCHAR FOR BIT DATA	32,672 bytes, binary data	varbinary	32K	
VARGRAPHIC	16,336 characters, double-byte character data	univarchar	32K	

Replication Server 15.0 Unsigned Datatype Mapping

For Replication Server 15.0 and later, unsigned datatypes are supported and can be specified in the replication definitions.

Table 23. Unsigned Integer Replication Definition Datatype Mapping

RepServer 15.0 Unsigned Datatypes	Replication Definition Datatypes
unsigned bigint	numeric (20)
unsigned int	numeric (10)
unsigned smallint	int
unsigned tinyint	tinyint

DECFLOAT Datatype Compatibility

Replication Agent for UDB supports the replication of the DECFLOAT datatype. Both 16 and 34 digits of precision are supported.

When Replication Agent for UDB replicates DECFLOAT columns from the DB2 primary to replicate databases that do not support the DECFLOAT type or an equivalent type, Replication

Agent maps the DECFLOAT type to the FLOAT type. Consequently, some loss of precision may result.

In addition to a number value, a DECFLOAT column may contain special values that are not supported by Replication Agent, such as positive and negative INFINITY, NAN, and SNAN. Replication of these values is not supported. In these cases, Replication Agent replicates special values to NULL, if the column is nullable, or to '0.0' if the column is not nullable.

XML Datatype Compatibility

Replication of the XML datatype is not supported by Replication Agent for UDB.

If you attempt to mark a table that has an XML column, Replication Agent reports an error. You can mark a table containing an XML column using the **force** option of the **pdb_setreptable** command, but this column is not replicated.

Replication Server set autocorrection Command

The Replication Server **set autocorrection** command prevents failures that would otherwise be caused by missing or duplicate rows in a replicated table.

The **set autocorrection** command corrects discrepancies that may occur during materialization by converting each **update** or **insert** operation into a **delete** followed by an **insert**.

In version 15.7.1 SP200, by default, Replication Agent autocorrects tables marked for replication when the **ra_autocorrect_on_mark** configuration parameter value is set to true. In earlier versions, you can set autocorrection from Replication Agent for one or all marked tables in the primary database by using the **ra_set_autocorrection** command as described in the *Replication Agent Reference Manual*. To set autocorrection from Replication Server, use the **set autocorrection** command in a replication definition. You must do this from Replication Server because Replication Agent cannot alter the autocorrection setting on a replication definition. See the *Replication Server Administration Guide*.

Large Identifiers

Replication Agent for UDB supports UDB 9.5 large identifiers—authorization ID, column, and schema names up to 128 bytes long.

To support UDB 9.5 large identifiers, you must migrate Replication Agent instances from versions 15.0, 15.1, and 15.2 to 15.5 or later to accommodate the modified Replication Agent system tables. You must also migrate Replication Agent instances when DB2 is upgraded from an earlier version to 9.5 to replicate tables with large identifiers.

See also

- *Upgrade and Migration Procedures for Replication Agent for UDB* on page 155

Compression

Replication Agent for UDB supports value compression—tables created with the **VALUE COMPRESSION** clause—and row compression.

Replication Agent Objects in the DB2 Primary Database

Replication Agent creates objects in the primary database to assist with replication tasks. Replication Agent also uses the native database transaction log maintained by the DB2 server to capture transactions in the primary database for replication.

The Replication Agent objects are created by invoking the **ra_admin** command with the **init** keyword. When you invoke this command, Replication Agent generates a SQL script that contains the SQL statements for the objects created or modified in the primary database. This script is stored in the `create.sql` file in the `RAX-15_5\inst_name\scripts\xlog\installed` directory. You must create these objects before marking any primary database objects for replication.

Note: The JAR files are installed when the **ra_admin init** command is executed. The **ra_admin deinit** command uninstalls the JAR files from the primary database. You must issue **ra_admin deinit** command before reinitializing Replication Agent.

See the *Replication Agent Administration Guide*.

See also

- *Java Procedure Objects* on page 138

Replication Agent Object Names

Replication Agent creates objects in the primary database to assist with replication tasks.

There are two variables in Replication Agent database object names:

- *prefix* – represents the one- to three-character string value of the **ra_admin_instance_prefix** parameter (the default is **ra_**).
- *xxx* – represents an alphanumeric counter, a string of characters that is (or may be) added to a table name to make that name unique in the database.

The value of **ra_admin_instance_prefix** is the prefix string used in all Replication Agent system object names.

If this value conflicts with the names of existing database objects in your primary database, you can change the value of **ra_admin_instance_prefix** by using the **ra_config** command.

Note: Replication Agent uses the value of **ra_admin_instance_prefix** to find its objects in the primary database. If you change the value of **ra_admin_instance_prefix** after you create the

Replication Agent objects, the Replication Agent instance cannot find the objects that use the old prefix.

Use the **ra_admin** command to view the names of Replication Agent objects in the primary database.

See the *Replication Agent Administration Guide* for details on setting up replication object names.

Table Objects

Replication Agent creates table objects in the DB2 primary database.

These tables are considered Replication Agent objects. No permissions are granted on these tables when they are created. All of these tables contain at least one index, and some contain more than one index.

Table 24. Replication Agent Tables

Table	Database name
Articles table	*prefix*varticles_*xxx*
LOB columns table	*prefix*vblob_columns_*xxx*
`rs_dump` shadow table	*prefix*dumpsh_*xxx*
Force record table	*prefix*force_record_*xxx*
Marked objects table	*prefix*vmarked_objs_*xxx*
`rs_marker` shadow table	*prefix*markersh_*xxx*
Proc active table	*prefix*procactive_*xxx*
Log Admin work table	*prefix*rawork_*xxx*
System table	*prefix*xlog_system_*xxx*

Java Procedure Objects

Replication Agent creates Java procedure objects in the DB2 primary database.

Replication Agent for UDB installs `SYBRAUJAR.jar` and `SYBTRUNCJAR.jar` into these directories.

- On Windows, the files are installed in `$DB2DIR/SQLLIB/FUNCTION/jar/`*pds_username*, where `$DB2DIR` is the path to the DB2 installation, and *pds_username* is the value of **pds_username**.

- On UNIX, the files are installed in `$HOME/sqllib/function/jar/` *pds_username*, where `$HOME` is the home directory of the DB2 instance owner, and *pds_username* is the value of **pds_username**.

Note: If more than one Replication Agent instance is configured for a DB2 server—one Replication Agent instance for each database—a unique primary database user name must be specified in the **pds_username** configuration parameter for each Replication Agent instance. This is required to install and uninstall these JAR files.

These JAR files implement several Java procedures in the UDB primary database and are created and used in log truncation.

Table 25. Java Procedures for Truncation

Procedure	Database name
Retrieves the name of the log file that contains the current LSN	*prefix*get_log_name_
Retrieves the version of the **get_log_name** Java class	*prefix*get_version_str_
Truncates the database log file or files from the archive log directory	*prefix*trunc_log_files_
Retrieves the version of the **trunc_log_files** Java class	*prefix*get_trunc_ver_str_

Finding the Names of Replication Agent Objects

The Replication Agent instance generates the names of its database objects. To find out the actual names of these objects, use the **ra_admin** command.

To obtain the generated names of Replication Agent objects in the primary DB2 database:

At the Replication Agent administration port, invoke the **ra_admin** command with no keywords:

```
ra_admin
```

The **ra_admin** command returns a list of objects in the primary database.

Marked Objects Table

One of the Replication Agent objects is the *marked objects table*. The marked objects table contains an entry for each marked table in the primary database.

Each marked table entry contains the:

- PRIMARY_NAME: Name of the marked primary object (table)
- REPL_NAME: Primary object replicated name

- OBJ_TYPE: Type of the primary object (table only, in Replication Agent for UDB)
- ENABLED: “Replication enabled” flag for the primary object
- SHADOW_NAME: Deprecated
- PROC_NAME: Deprecated
- OWNER: Owner of the primary object
- REPDEFMODE: “Send owner” flag
- TABLSP_ID: Tablespace ID of the primary object
- TABLE_ID: Table ID of the primary object
- CONVERT_DT: “Convert datetime” flag
- DATA_CAPTURE: Original value of the table **DATA CAPTURE** attribute
- AUTO_CORRECTION: Autocorrection flag
- VERSION: LSN of the end of log when the table is marked
- MARKED: "mark" flag

Transaction Log Truncation

Replication Agent supports both automatic and manual log truncation.

Replication Agent provides two options for automatic transaction log truncation:

- Periodic truncation, based on a time interval you specify
- Automatic truncation whenever Replication Agent receives a new LTM locator value from the primary Replication Server

To configure Replication Agent log truncation, observe these guidelines:

- All DB2 transaction logs are maintained through the data server. You can configure Replication Agent for UDB to truncate transaction logs from either the active or the archive log directory. When you have enabled DB2 archiving with **LOGARCHMETH1**, you can also configure a second archive location by setting the **LOGARCHMETH2** DB2 configuration parameter. DB2 then archives logs into the two directories. You can then configure Replication Agent to automatically truncate the processed archives from one of these directories.

  Set **pdb_archive_path** to point to the location specified by either **LOGARCHMETH1** or **LOGARCHMETH2**.

  Warning! If you enable truncation without also setting **pdb_archive_path**, Replication Agent deletes the primary database log files it no longer needs from the active log directory using the DB2 **prune** command. Because the active directory is used for DB2 recovery, it is recommended that you do not set **pdb_archive_path** to point to your active directory and also that you do not enable truncation without first setting **pdb_archive_path**.
- Set **pdb_archive_remove** to true if you want Replication Agent to delete archives that are no longer necessary.

Note: By default, **pdb_archive_remove** is set to false. You must configure **pdb_archive_path** before setting **pdb_archive_remove** to true.

- To enable automatic truncation, set **truncation_type** to interval, and set **truncation_interval** to a value greater than 0 (zero), which deletes log files at the designated interval. Alternately, set **truncation_type** to locator_update, which causes truncation to occur each time Replication Agent receives a new LTM locator value from the primary Replication Server.
- You can truncate the Replication Agent transaction log manually, at any time, by invoking **pdb_truncate_xlog** at the Replication Agent administration port.
- When DB2 truncate runs, the oldest LSN for which Replication Agent has not processed a commit/rollback (oldest active LSN) is obtained and the archive log file that contains the LSN is determined. All archive log files up to but not including the file with the oldest active LSN are deleted.

For more information on these properties, see the *Replication Agent Reference Manual*. For a more detailed description of truncating, see "Administering Replication Agent" in the *Replication Agent Administration Guide*.

See also

- *Java Procedure Objects* on page 138

Upgrading and Downgrading Replication Agent

Review the procedures for upgrading and downgrading Replication Agent.

Warning! Replication Agent cannot be downgraded after you use any new features. Before you use any new features, test the new version of Replication Agent at the same functional level as your previous version of Replication Agent. Only when you are satisfied with the new version of Replication Agent at this existing functional level should you finalize the upgrade by using new features. You should also make a backup copy of an instance of the Replication Agent version to which you have upgraded before attempting to use any new feature.

Upgrade and Migration Procedures for Replication Agent for Oracle

Replication Agent for Oracle 15.7.1 SP200 does not have to be installed on the same machine as the primary Oracle data server.

Using any of the upgrade procedures described here, the new Replication Agent for Oracle 15.7.1 SP200 instances will have the same configuration as previously existing instances, including instance names, administrative user IDs and passwords, and administrative port numbers.

Replication Agent for Oracle 15.7.1 SP200 does not have to be installed on the same machine as the primary Oracle data server. However:

- Replication Agent for Oracle must be installed on a host that has access to Oracle LogMiner.
- If Replication Agent for Oracle is configured to automatically truncate Oracle logs, it must be installed on a machine that has direct access to the Oracle logs.

If you install Replication Agent for Oracle 15.7.1 SP200 on the same host on which the primary Oracle server is running, you must:

1. Obtain a local copy of the Oracle timezone file, so Replication Agent can correctly process the Oracle timestamp with *timezone* datatype.
2. Configure the **pdb_timezone_file** parameter.

Note: For upgrades within a common release level, as in the case of an ESD applied to a particular version of Replication Agent, use the **ra_admin -u** option applied to a particular instance of Replication Agent or to all instances of Replication Agent. See the *Replication Agent Administration Guide*.

Upgrading Replication Agent for Oracle to 15.7.1 SP200

Upgrade Replication Agent for Oracle to version 15.7.1 SP200.

1. Set the SYBASE environment variables by changing to the `SYBASE` directory in which Replication Agent 15.7.1 SP200 is installed and sourcing the `SYBASE` script:
   - For C Shell: `source SYBASE.csh`
   - For Bourne or Korn shell: `. SYBASE.sh`
   - For Windows, run: `SYBASE.bat`
2. Change to the Replication Agent `bin` directory:
   - On UNIX:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
   - On Windows:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```

   Note: The Replication Agent directory name for release 15.5 and later is RAX-15_5.
3. Upgrade the Replication Agent instances.

Upgrading instances within the same product installation	• To upgrade a specific Replication Agent instance in the Replication Agent installation directory, at the command prompt, run:
	`./ra_admin.sh -u instance=<instance_name>`
	• To upgrade all Replication Agent instances in the Replication Agent installation directory, at the command prompt, run:
	`./ra_admin.sh -u all`

Upgrading instances in a separate product installation	• To upgrade a specific Replication Agent instance in the specified source instance directory from the current product installation directory, run:
	`./ra_admin.sh -u <source_instance_directory>`
	• To upgrade all Replication Agent instances in the specified source installation directory from the current product installation directory, at the command prompt, run:
	`./ra_admin.sh -u <source_installation_directory>`

The configuration files are backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back. The **all** option requires relatively less space because upgrades are performed directly on instances within the current product installation directory, not to copies. However, reversing an upgrade is more difficult for the same reason.

4. Start the Replication Agent instance:

   ```
   $SYBASE/RAX-15_5/<instance>/RUN_<instance>
   ```

5. Log in to Replication Agent instance and migrate the Replication Agent metadata by running:

   ```
   ra_migrate
   ```

   Note: You can use the **ra_finalize_upgrade** to manually force upgrade finalization of an instance from a previous version and prevent downgrade to the previous version. You must finalize the upgrade to enable any new functionality. See the *Replication Agent Reference Manual* for details on using the **ra_finalize_upgrade** command.

6. Resume replication.

   ```
   resume
   ```

Migrating Replication Agent for Oracle 15.7.1 SP200 When Upgrading Oracle 10g to 11g

Migrate Replication Agent 15.7.1 SP200 when you are also upgrading Oracle 10g to 11g.

Replication Agent for Oracle migration to support upgrading Oracle 10g to Oracle 11g is similar to upgrading Replication Agent for Oracle 15.1 or 15.2 to Replication Agent for Oracle 15.7.1 ESD #2.

Note: Quiesce the Replication Agent before upgrading Oracle 10g to Oracle 11g. The replication environment must have completed processing of all transactions before upgrading Oracle because the Replication Agent moves the truncation point to the end of the log during Replication Agent migration.

1. Follow the steps that Oracle provides in their documentation for upgrading from Oracle 10g to Oracle 11g.
2. After upgrading Oracle, restart the Replication Agent, and issue the **ra_migrate** command.
3. As with the log-based Replication Agent upgrade process, you may need to reconfigure the Replication Agent for Oracle instance to read archive logs depending on the configuration in Oracle. This may change following the Oracle upgrade.

If you are upgrading from log-based Replication Agent and upgrading Oracle 10g to Oracle 11g at the same time, migrate Replication Agent 15.7.1 ESD #2 only once.

Upgrading the Replication Agent for Oracle from LogMiner to XStream

To upgrade the Replication Agent for Oracle Log Reader component from LogMiner to XStream APIs, use the **ra_admin** command with the **-u** option.

Prerequisites

- Make sure the primary database version is Oracle11.2.0.3 or later.

- Initialize the Replication Agent.
- Set the Oracle library path and the XStream JDBC driver in the CLASSPATH environment variable for your platform. See *Setting Up the Oracle XStream JDBC Driver* on page 77.
- Shut down the Replication Agent instance and make sure no database operations are running to avoid risking data loss due to missed replication.

Note: You cannot downgrade to LogMiner once you have upgraded the Replication Agent Log Reader to use XStream APIs.

Task

1. To use maximum-strength password encryption, install Java Cryptography Extension (JCE) policy files and reset the system administrator account.

   See *Installing Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7* in the *Replication Agent Administration Guide*.
2. Set the SYBASE environment variable by changing the `SYBASE` directory in which Replication Agent is installed.
   - For C Shell: `source SYBASE.csh`
   - For Bourne or Korn shell: `. SYBASE.sh`
   - For Windows, run: `SYBASE.bat`
3. Change to the Replication Agent `bin` directory using the command line prompt:
   - On UNIX:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
   - On Windows:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```
4. Upgrade the Replication Agent instances to use Oracle XStream APIs.

Upgrading instances within the same product installation	• To upgrade a specific Replication Agent instance in the Replication Agent installation directory, at the command prompt, run:
	On UNIX:
	`./ra_admin.sh -u instance=<instance_name> -t oraclexs`
	On Windows:
	`ra_admin.bat -u instance=<instance_name> -t oraclexs`
	• To upgrade all Replication Agent instances in the Replication Agent installation directory, at the command prompt, run:
	`./ra_admin.sh -u all -t oraclexs`
	`ra_admin.bat -u all -t oraclexs`
	Note: Because the current version upgrade is not supported, you cannot upgrade the Replication Agent for Oracle 15.7.1 SP200 Log Reader component from LogMiner to use XStream APIs.

Upgrading instances in a separate product installation	• To upgrade a specific Replication Agent instance in the specified source instance directory from the current product installation directory, run:
	On UNIX:
	`./ra_admin.sh -u <source_instance_directory> -t oraclexs`
	On Windows:
	`ra_admin.bat -u <source_instance_directory> -t oraclexs`
	• To upgrade all Replication Agent instances in the specified source installation directory from the current product installation directory, at the command prompt, run:
	`./ra_admin.sh -u <source_installation_directory>`
	`ra_admin.bat -u <source_installation_directory>`
	If the Oracle primary database version is earlier than 11.2.0.3, the upgrade process stops and shows: "`The Replication Agent upgrade is not supported on <instance_name> for database version <xx.xx.xx.xx>. The minimum database version supported to upgrade Replication Agent from LogMiner to XStream APIs is:11.2.0.3.`"

In the commands above:

- *all* – upgrades all instances of Replication Agent within the current product installation directory except the Replication Agent 15.7.1 SP200 instance with the LogMiner

component. The configuration files are backed up before the upgrade. If an error occurs, the upgrade is rolled back.

- *instance_name* – is the name of the Replication Agent instance to be upgraded. When you enter *instance_name*, the upgrade is performed directly on the specified instance. The configuration file is backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back.
- *source_installation_directory* – is the Replication Agent source installation directory. When you enter the *source_installation_directory*, all instances of Replication Agent in the specified source installation directory are upgraded from the current product installation directory provided the earlier version of the Replication Agent instance uses LogMiner component to read from the transaction logs. The instance is first copied to and then updated within the current product installation directory
- *oraclexs* – is the Oracle XStream database type.
  When you enter -t *oraclexs*, the Log Reader component is upgraded from LogMiner to Oracle XStream APIs.

5. Start the Replication Agent instance.

   On UNIX:

   ```
   $SYBASE/RAX-15_5/<instance>/RUN_<instance>
   ```

   On Windows:

   ```
   %SYBASE%\RAX-15_5\<instance>\RUN_<instance>
   ```

   where *instance* is the Replication Agent instance name.
6. Log in to the Replication Agent instance and migrate the metadata by running:

   ```
   ra_migrate
   ```

   This command initiates the upgrade process, finalizes the upgrade of an instance from an earlier version, and prevents a subsequent downgrade to the earlier version.

   During upgrade, Replication Agent creates an XStream outbound server with the current start system change number (SCN) to read from transactions logs.
7. Once the Replication Agent is upgraded, set the Oracle environment variables for XStream APIs.
8. Execute the Replication Server System Database (RSSD) **rs_zeroltm** command at the primary database connection to reset the LTM locator value to zero.

   Run **rs_zeroltm** before resuming replication, to prevent Replication Server from requesting a log starting point that occurs earlier in the log than the location established by the **ra_migrate** option. The Replication Server LTM locator value for the primary connection must be zeroed.

   For example:

   ```
   rs_zeroltm data_server, database
   ```
9. Resume replication.

   ```
   resume
   ```

Before resuming replication, wait for the upgrade process to complete. Otherwise, data inserted during upgrade is not replicated.

Upgrading the Replication Agent for Oracle from XStream to XStream

To upgrade the Replication Agent for Oracle from 15.7.1 SP120 to version15.7.1 SP200 with XStream APIs Log Reader, use the **ra_admin** command with the **-u** option.

Prerequisites

- Make sure the primary database version is Oracle11.2.0.3 or later.
- Initialize the Replication Agent.
- Set the Oracle library path and the XStream JDBC driver in the CLASSPATH environment variable for your platform. See *Setting Up the Oracle XStream JDBC Driver* on page 77.
- Shut down the Replication Agent instance and make sure no database operations are running to avoid risking data loss due to missed replication.

Task

1. To use maximum-strength password encryption, install Java Cryptography Extension (JCE) policy files and reset the system administrator account.

   See *Installing Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7* in the *Replication Agent Administration Guide*.
2. Set the SYBASE environment variable by changing the `SYBASE` directory in which Replication Agent is installed.
   - For C Shell: `source SYBASE.csh`
   - For Bourne or Korn shell: `. SYBASE.sh`
   - For Windows, run: `SYBASE.bat`
3. Change to the Replication Agent `bin` directory using the command line prompt:
   - On UNIX:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
   - On Windows:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```
4. Upgrade the Replication Agent instances to use Oracle XStream APIs.

Upgrading instances within the same product installation	• To upgrade a specific Replication Agent instance in the Replication Agent installation directory, at the command prompt, run:
	On UNIX:
	`./ra_admin.sh -u instance=<instance_name>`
	On Windows:
	`ra_admin.bat -u instance=<instance_name>`
	• To upgrade all Replication Agent instances in the Replication Agent installation directory, at the command prompt, run:
	`./ra_admin.sh -u all`
	`ra_admin.bat -u all`
	Note: Because the current version upgrade is not supported, you cannot upgrade the Replication Agent for Oracle 15.7.1 SP200 Log Reader component to use XStream APIs.

Upgrading instances in a separate product installation	• To upgrade a specific Replication Agent instance in the specified source instance directory from the current product installation directory, run:
	On UNIX:
	`./ra_admin.sh -u <source_instance_directory>`
	On Windows:
	`ra_admin.bat -u <source_instance_directory>`
	• To upgrade all Replication Agent instances in the specified source installation directory from the current product installation directory, at the command prompt, run:
	`./ra_admin.sh -u <source_installation_directory>`
	`ra_admin.bat -u <source_installation_directory>`

In the commands above:

- *all* – upgrades all instances of Replication Agent within the current product installation directory except the Replication Agent 15.7.1 SP200 instance with the XStream component. The configuration files are backed up before the upgrade. If an error occurs, the upgrade is rolled back.
- *instance_name* – is the name of the Replication Agent instance to be upgraded. When you enter *instance_name*, the upgrade is performed directly on the specified instance. The configuration file is backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back.
- *source_installation_directory* – is the Replication Agent source installation directory. When you enter the *source_installation_directory*, all instances of Replication Agent in the specified source installation directory are upgraded from the current product installation directory provided the earlier version of the Replication Agent instance

uses XStream component to read from the transaction logs. The instance is first copied to and then updated within the current product installation directory

5. Start the Replication Agent instance.

   On UNIX:

   ```
   $SYBASE/RAX-15_5/<instance>/RUN_<instance>
   ```

   On Windows:

   ```
   %SYBASE%\RAX-15_5\<instance>\RUN_<instance>
   ```

   where *instance* is the Replication Agent instance name.

6. Log in to the Replication Agent instance and migrate the metadata by running:

   ```
   ra_migrate
   ```

   This command initiates the upgrade process from an earlier version instance.

7. (Optional) Execute **ra_finalize_upgrade** to finalize the upgrade from an earlier version instance.

8. Resume replication.

   ```
   resume
   ```

Note: Before resuming replication, wait for the upgrade process to complete. Otherwise, data inserted during upgrade is not replicated.

Upgrade Procedures for Replication Agent for Microsoft SQL Server

Replication Agent for Microsoft SQL Server must be installed on the same Windows host on which the primary Microsoft SQL Server is running, and Replication Agent for Microsoft SQL Server cannot be installed on a UNIX or Linux host. Before upgrading, consider where the existing instance of the earlier version of Replication Agent is installed and the current version of the primary data server.

When you use any of the upgrade procedures described in this section, the new Replication Agent for Microsoft SQL Server instances will have the same configuration as previously existing instances, including instance names, administrative user IDs and passwords, and administrative port numbers.

Upgrading Replication Agent for Microsoft SQL Server to 15.7.1 SP200

Upgrade Replication Agent for Microsoft SQL Server to version 15.7.1 SP200.

Note: Replication Agent 15.7.1 SP200 must be installed on the same host on which the primary Microsoft SQL Server is running.

1. Stop the primary Microsoft SQL Server database service.
   a) From **Windows Control Panel**, go to **Administrative Tools > Services**, and find the service named SQL Server (SERVER).

      where SERVER is the name of your Microsoft SQL Server data server.
   b) Stop this service.
2. Restart Microsoft SQL Server in a single-user mode.

   For example:

   ```
   "C:\Program Files\Microsoft SQL
   Server\MSSQL.1\MSSQL\Binn\sqlservr.exe" -m -s
   instanceName
   ```

   Note: The directory path may vary depending on the version of Microsoft SQL Server.

   where *instanceName* is the name of the Microsoft SQL Server instance.
3. Open a command window.
4. Set the SYBASE environment variables by changing to the `SYBASE` directory in which Replication Agent 15.7.1 is installed and executing the `SYBASE.bat` script.
5. Change to:

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

   Note: The Replication Agent directory name for release 15.5 and later is RAX-15_5.
6. Upgrade the Replication Agent instances.

Upgrading instances within the same product installation	• To upgrade a specific Replication Agent instance in the Replication Agent installation directory, at the command prompt, run:
	`ra_admin.bat -u instance=<instance_name>`
	• To upgrade all Replication Agent instances in the Replication Agent installation directory, at the command prompt, run:
	`ra_admin.bat -u all`

Upgrading instances in a separate product installation	• To upgrade a specific Replication Agent instance in the specified source instance directory from the current product installation directory, run:
	`ra_admin.bat -u <source_instance_directory>`
	• To upgrade all Replication Agent instances in the specified source installation directory from the current product installation directory, at the command prompt, run:
	`ra_admin.bat -u <source_installation_directory>`

   The configuration files are backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back. The **all** option requires relatively less space because upgrades are performed directly on instances within the current product

installation directory, not to copies. However, reversing an upgrade is more difficult for the same reason.

7. Start and log in to each of the Replication Agent for Microsoft SQL Server 15.7.1 SP200 instances and:
   a) Set the **rs_charset** configuration parameter to match the SAP Replication Server character set, as described in the *Replication Agent Reference Manual*.
   b) In the command line prompt, to ensure that Replication Agent can connect to both Microsoft SQL Server and SAP Replication Server, enter:

      ```
      test connection
      ```

   c) To remove the existing stored procedures installed, enter:

      ```
      server_admin remove
      ```

   d) To reinstall the **sp_SybSetLogforReplTable**, **sp_SybSetLogforReplProc**, and **sp_SybSetLogforLOBCol** stored procedures, and install the **sp_SybTruncateTable** stored procedure, enter:

      ```
      server_admin init
      ```

   e) To initialize the Replication Agent instance and migrate the Replication Agent instance metadata, enter:

      ```
      ra_migrate
      ```

      When this command executes in the first Replication Agent 15.7.1 instance, it will also initialize the Microsoft SQL Server. In subsequent Replication Agent 15.7.1 instances, it will only initialize the instance and migrate the instance metadata.

      Note: You can use the **ra_finalize_upgrade** to manually force upgrade finalization of an instance from a previous version and prevent downgrade to the previous version. You must finalize the upgrade to enable any new functionality. See **ra_finalize_upgrade**in the *Replication Agent Reference Manual*.

8. In the command prompt, stop the primary Microsoft SQL Server database using CTRL+C or enter:

   ```
   shutdown
   ```

9. Restart the primary Microsoft SQL Server database in multiuser mode.
   a) From the **Windows Control Panel**, go to **Administrative Tools > Services**, and find the service named SQL Server (SERVER).

      where SERVER is the name of your Microsoft SQL Server data server.
   b) Start this service.
10. Restart the Replication Agent for Microsoft SQL Server instance.
11. Resume replication.

    ```
    resume
    ```

Migrating Replication Agent for Microsoft SQL Server When Upgrading SQL Server Database

Migrate Replication Agent for Microsoft SQL Server when you upgrade SQL Server database from 2008 to 2008R2 version.

1. To prevent loss of any replicated data, deny users—other than existing Replication Agent **pds_username** users—access to the primary database.

   See the Microsoft SQL Server documentation.
2. Log in to the Replication Agent for Microsoft SQL Server 15.7.1 SP200 instance, verify that it is in the *Replicating* state, and allow replication to finish.

   To verify that replication has completed:

   a) Periodically issue the **ra_statistics** command, until all of these statistics are 0 (zero):

      - `Input queue size`
      - `Output queue size`

   b) Note the `Last QID Sent` from the last set of statistics.

   c) Issue the **ra_locator update** command so that Replication Agent retrieves the truncation point from Replication Server.

   d) Issue the **ra_locator** command again and compare the displayed locator with that of the `Last QID Sent`. If they are different, wait and repeat this step.

   e) Quiesce the Replication Agent instance by issuing the **quiesce** command.

   f) Shut down the Replication instance by issuing the **shutdown** command.
3. Upgrade Microsoft SQL Server database from 2008 to 2008R2 version, following the steps in the Microsoft SQL Server documentation.
4. Start the primary database in single-user mode:

   ```
   "C:\Program Files\Microsoft SQL Server\MSSQL<version>\MSSQL\Binn
   \sqlservr.exe" -m -s instanceName
   ```

   where:

   - *version* – is the version number of the database.
   - *instanceName* – is the name of the Microsoft SQL Server instance.
5. Start the Replication Agent instance in one of these ways:

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

   The RUN script and the **ra** utilities are batch files on Microsoft Windows.

   See *Replication Agent Start-Up* in the *Replication Server Administration Guide* to start the instance on your platform.

6. Initialize Replication Agent system database objects by issuing **server_admin init**.
7. Restart the primary Microsoft SQL Server database in multiuser mode:
   a) From Windows Control Panel, go to **Administrative Tools > Services**, and find the service named SQL Server (SERVER).

      where *SERVER* is the name of your Microsoft SQL Server data server.
   b) Start this service.
8. Migrate the Replication Agent metadata by issuing **ra_migrate**.
9. In the Replication Agent instance, resume replication by issuing **resume**.
10. Allow all users to access the primary database.

Upgrade and Migration Procedures for Replication Agent for UDB

Replication Agent for UDB 15.7.1 SP200 provides automatic upgrade of Replication Agent for UDB version 15.0 and later instances.

You can migrate a Replication Agent for UDB instance when you upgrade from these IBM DB2 versions:

- 8.2 or 9.1 to 9.5 or 9.7
- 9.5 to 9.7
- 9.7 to 10.1 or 10.5

When you use any of the upgrade procedures described in this section, the new Replication Agent for UDB 15.7.1 instances will have the same configuration as previously existing instances, including instance names, administrative user IDs and passwords, and administrative port numbers.

Replication Agent for UDB 15.7.1 does not support:

- Upgrading Replication Agent for UDB version 12.6 or earlier to version 15.0 or later.
- Migrating Replication Agent for UDB 12.6 when UDB is upgraded from version 6 or 7 to version 8 or 9.

Upgrading Replication Agent for UDB to 15.7.1 SP200

Upgrade Replication Agent for UDB versions 15.5 and later to version 15.7.1 SP200.

1. Set the SYBASE environment variables by changing to the `SYBASE` directory in which Replication Agent 15.7.1 SP200 is installed and sourcing the `SYBASE` script:
   - For C Shell: `source SYBASE.csh`
   - For Bourne or Korn shell: `. SYBASE.sh`

- For Windows, run: `SYBASE.bat`

2. Change to the Replication Agent `bin` directory:

   - On UNIX:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
   - On Windows:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```

Note: The Replication Agent directory name for release 15.5 and later is RAX-15_5.

3. Upgrade the Replication Agent instances.

Upgrading instances within the same product installation	• To upgrade a specific Replication Agent instance in the Replication Agent installation directory, at the command prompt, run:
	`./ra_admin.sh -u instance=<instance_name>`
	• To upgrade all Replication Agent instances in the Replication Agent installation directory, at the command prompt, run:
	`./ra_admin.sh -u all`

Upgrading instances in a separate product installation	• To upgrade a specific Replication Agent instance in the specified source instance directory from the current product installation directory, run:
	`./ra_admin.sh -u <source_instance_directory>`
	• To upgrade all Replication Agent instances in the specified source installation directory from the current product installation directory, at the command prompt, run:
	`./ra_admin.sh -u <source_installation_directory>`

   The configuration files are backed up before the upgrade for use in error recovery, if required. If an error occurs, the upgrade is rolled back. The **all** option requires relatively less space because upgrades are performed directly on instances within the current product installation directory, not to copies. However, reversing an upgrade is more difficult for the same reason.

4. Start the Replication Agent instance:
   ```
   $SYBASE/RAX-15_5/<instance>/RUN_<instance>
   ```
5. Log in to Replication Agent instance and migrate the Replication Agent metadata by running:
   ```
   ra_migrate
   ```

Note: You can use the **ra_finalize_upgrade** to manually force upgrade finalization of an instance from a previous version and prevent downgrade to the previous version. You must

finalize the upgrade to enable any new functionality. See the *Replication Agent Reference Manual* for details on using the **ra_finalize_upgrade** command.

6. Resume replication.

```
resume
```

Migrating Replication Agent for UDB When Upgrading DB2

Migrate Replication Agent for UDB when you are also upgrading IBM DB2 database version.

You can migrate a Replication Agent for UDB instance when you upgrade from these IBM DB2 versions:

- 8.2 or 9.1 to 9.5 or 9.7
- 9.5 to 9.7
- 9.7 to 10.1 or 10.5

1. To prevent loss of any replicated data, deny users—other than the previously existing Replication Agent **pds_username** users—any further access to the primary database.
2. Log in to the Replication Agent 15.7.1 SP200 instance, verify that it is in *Replicating* state, and allow replication to finish. To verify that replication has completed:
   a) Periodically issue the **ra_statistics** command, watching until all of these statistics are 0 (zero):
      - `Input queue size`
      - `Output queue size`
   b) When all of these values are zero, note the `Last QID Sent` from the last set of statistics.
   c) Issue the **ra_locator update** command so that Replication Agent retrieves the truncation point from Replication Server.
   d) Wait, then issue the **ra_locator** command again and compare the displayed locator with that of the `Last QID Sent`. If they are different, wait and repeat this step.
   e) Quiesce the Replication Agent instance by issuing the **quiesce** command.
   f) Shut down the Replication instance by issuing the **shutdown** command.
3. Follow the steps in the DB2 documentation for upgrading DB2.
4. Verify that all the primary database requirements are met.

   Note: If the **use_rssd** configuration parameter was set to true before migration, skip this step.
5. Start the Replication Agent instance, and set the **use_rssd** configuration parameter to true:

```
ra_config use_rssd, true
```

   Replication Agent for UDB uses this configuration to connect to the RSSD and to reset the locator to zero.

6. Migrate the Replication Agent metadata by issuing the **ra_migrate** command.

   Note: If the **use_rssd** configuration parameter was set to true before migration, skip this step.

7. In the Replication Agent 15.7.1 instance, resume replication by issuing the **resume** command.
8. Allow all users to access the primary database.

Note: If you are upgrading Replication Agent and upgrading UDB from version 8.2 or 9.1 to version 9.5 or 9.7 at the same time, you need to migrate Replication Agent only once.

See also

- *IBM DB2 Requirements* on page 123

Downgrading Replication Agent for Oracle

You can downgrade Replication Agent for Oracle from version 15.7.1 SP200 to version 15.7.1 ESD #2 or later.

You may need to downgrade Replication Agent if the upgrade process fails or if replication fails after an upgrade. Replication may fail when new features fail to function as expected or if there are changes to:

- DDL and how it is handled by Replication Agent
- The content or structure of the Replication Agent System Database (RASD)

If you are using any new features from Replication Agent 15.7.1 SP200, you cannot downgrade. For a list of the new features for Replication Agent 15.7.1 SP200, see the *New Features in Replication Agent* in the *SAP Replication Server Options New Features Bulletin*.

Replication Agent 15.7.1 SP200 must be installed on the same platform on which the primary Oracle server is running.

1. Change to the Replication Agent 15.7.1 SP200 `bin` directory:

   - On UNIX:

     ```
     cd $SYBASE/RAX-15_5/bin
     ```

   - On Windows:

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

   The Replication Agent directory name for release 15.5 and later is RAX-15_5.

2. Run the **ra_downgrade** command at the Replication Agent instance from which you are downgrading (the current version):

   ```
   ra_downgrade
   ```

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. 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 absolute path to this file is returned if **ra_downgrade** executes successfully.

Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands are deprecated as of Replication Agent 15.7.1.

3. Complete the downgrade by running the **ra_migrate** command at the Replication Agent instance to which you are downgrading (the earlier version):

```
ra_migrate
```

   If the **ra_migrate** command executes successfully, Replication Agent shuts down.

4. Start the Replication Agent instance to which you have downgraded (the earlier version), and resume replication:

```
resume purge
```

   The **purge** keyword is needed here to purge data from the Replication Server inbound queue for the connection to which this Replication Agent is connected. Purging prevents any duplicate records from being created in Replication Server as a result of the change in OQID formats between the earlier and later versions of Replication Agent.

Downgrading Replication Agent for Microsoft SQL Server

You can downgrade Replication Agent for Microsoft SQL Server from version 15.7.1 SP200 to version 15.7.1 ESD #2 or later.

You may need to downgrade Replication Agent if the upgrade process fails or if replication fails after an upgrade. Replication may fail when new features fail to function as expected or if there are changes to:

- DDL and how it is handled by Replication Agent
- The format of the origin queue ID (OQID)
- The content or structure of the Replication Agent System Database (RASD)
- Replication Agent system objects in the primary database

If you are using any new features from Replication Agent 15.7.1 SP200, you cannot downgrade. For a list of the new features for Replication Agent 15.7.1 SP200, see the *New Features in Replication Agent* in the *SAP Replication Server Options New Features Bulletin*.

1. Change to the Replication Agent 15.7.1 SP200 `bin` directory:

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

Note: The Replication Agent directory name for release 15.5 and later is RAX-15_5.

2. Run the **ra_downgrade** command at the Replication Agent instance from which you are downgrading (the current version):

   ```
   ra_downgrade
   ```

   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. 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 absolute path to this file is returned if **ra_downgrade** executes successfully.

   Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands are deprecated as of Replication Agent 15.7.1.

3. Complete the downgrade by running the **ra_migrate** command at the Replication Agent instance to which you are downgrading (the earlier version):

   ```
   ra_migrate timestamp.export
   ```

   where *timestamp*.`export` is the file to which the **ra_downgrade** command extracted RASD contents.

   If the **ra_migrate** command executes successfully, Replication Agent shuts down.

4. Start the Replication Agent instance to which you have downgraded (the earlier version), and resume replication:

   ```
   resume purge
   ```

   The **purge** keyword is needed here to purge data from the Replication Server inbound queue for the connection to which this Replication Agent is connected. Purging prevents any duplicate records from being created in Replication Server as a result of the change in OQID formats between the earlier and later versions of Replication Agent.

Downgrading Replication Agent for UDB

You can downgrade Replication Agent for UDB from version 15.7.1 SP200 to version 15.7.1 ESD #2 or later.

You may need to downgrade Replication Agent if the upgrade process fails or if replication fails after an upgrade.

If you are using any new features from Replication Agent 15.7.1 SP200, you cannot downgrade. For a list of the new features for Replication Agent 15.7.1 SP200, see the *New Features in Replication Agent* in the *SAP Replication Server Options New Features Bulletin*.

1. Change to the Replication Agent 15.7.1 SP200 `bin` directory:
   - On UNIX:
     ```
     cd $SYBASE/RAX-15_5/bin
     ```
   - On Windows:
     ```
     cd %SYBASE%\RAX-15_5\bin
     ```

   The Replication Agent directory name for release 15.5 and later is RAX-15_5.
2. Log in to the Replication Agent for UDB 15.7 instance. Run the **ra_downgrade** command at the Replication Agent instance from which you are downgrading (the current version):
   ```
   ra_downgrade
   ```

   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. 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 absolute path to this file is returned if **ra_downgrade** executes successfully.

   Note: The **ra_downgrade_prepare** and **ra_downgrade_accept** commands are deprecated as of Replication Agent 15.7.1.
3. Shut down the Replication instance by issuing the **shutdown** command.
4. Start and log in to the Replication Agent instance to which you are downgrading (the earlier version), and run **ra_migrate**:
   ```
   ra_migrate
   ```
5. Resume replication.
   ```
   resume
   ```

sybfilter Driver Reference

Learn how to install, configure, use, and troubleshoot the sybfilter driver.

Replication Agent must be able to directly read Microsoft SQL Server log files. However, the Microsoft SQL Server process opens these log files with exclusive read permission, and the files cannot be read by any other processes, including Replication Agent. Before Replication Agent can replicate data, you must use the sybfilter driver to make the log files readable.

Determining the Microsoft Filter Manager Library Version

For the sybfilter driver to work properly, the Microsoft Filter Manager Library must be version 5.1.2600.2978 or later.

To determine the version of the library:

1. In Windows Explorer, right-click `c:\windows\system32\fltlib.dll`.
2. Select **Properties**, and click the **Version** tab in the **Properties** dialog.
3. If the version is earlier than 5.1.2600.2978, go to the Microsoft Web site at *http://windowsupdate.microsoft.com*, and update your Windows system.

Installing and Setting Up the sybfilter Driver

Install and set up the sybfilter driver.

Note: On Windows Vista, you must be logged in as an Administrator to install, set up, and run the sybfilter driver.

1. In Windows Explorer, navigate to the sybfilter driver installation directory. On Windows, this directory is located at `%SYBASE%\RAX-15_5\system\` *<platform>*.

   where *<platform>* is one of:

   - winx86–if your operating system is 32-bit version of Windows Server 2003, Windows Server 2008, Windows Vista, or Windows XP.
   - winx64–if your operating system is 64-bit version of Windows Server 2003 or Windows XP.
   - winvistax64–if your operating system is 64-bit version of Windows Server 2008 or Windows Vista.
2. Right-click `sybfilter.inf` to install the sybfilter driver.

   Note: There can be only one sybfilter driver on a Windows machine. Once the driver is installed, it works for all Replication Agent for Microsoft SQL Server instances running

on the same machine. The sybfilter driver must be installed on the same machine as the primary Microsoft SQL Server.

3. In any directory, create a configuration file to store all log file paths for primary databases. The configuration file must have a `.cfg` suffix.

   For example, under the directory `%SYBASE%\RAX-15_5\system\<platform>`, create a file named `LogPath.cfg`.

4. Add a system environment variable named *RACFGFilePath*, and set its value to the path of the configuration file.
   a) From the **Control Panel**, open **System > Advanced > Environment Variables**.
   b) Click **New** to add a new system variable.
   c) Name the variable *RACFGFilePath*, and set its value to the location of the your configuration file.
5. In Windows Explorer, navigate to `%SYBASE%\RAX-15_5\bin`, and double-click `sybfiltermgr.exe` to start the sybfilter driver management console.
6. To start the sybfilter driver, enter `start` at the management console.
7. Add the log file path to the sybfilter driver with the user manager or by modifying the configuration file. Use directory and drive names that are recognizable to the primary Microsoft SQL Server.

   - User manager – use the **add** command in the management console. The syntax for this command is:

     ```
     add serverName dbName logFilePath
     ```

     For example, to add the log file named `pdb2_log.ldf` at `D:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\` to the *dbName* database on the *serverName* data server:

     ```
     add myseverName dbName D:\Program Files\Microsoft SQL Server
     \MSSQL.1\MSSQL\Data\pdb2_log.ldf
     ```

     Note: If you add the log file path with the user manager, the user manager automatically refreshes all log paths to the sybfilter driver after adding the log path into the configuration file.

   - Configuration file – to add the log file path directly to the configuration file, open and manually edit the configuration file. This is an example of log file path entries:

     ```
     [myserver, pdb1]

     log_file_path=D:\Program Files\Microsoft SQL Server\MSSQL.
     1\MSSQL\Data\pdb11_log.ldf

     log_file_path=D:\Program Files\Microsoft SQL Server\MSSQL.
     1\MSSQL\Data\pdb12_log.ldf

     [myserver, pdb2]

     log_file_path=D:\Program Files\Microsoft SQL Server\MSSQL.
     1\MSSQL\Data\pdb2_log.ldf
     ```

Note: Once you have added the log file paths to the configuration file, use the **refresh** command in the management console.

8. If you added a log file for your primary database before adding the log file path to the sybfilter driver, restart Microsoft SQL Server to make the log file readable.
9. At the management console, enter `check` to verify that log files are readable.

   If some log files are unreadable, make sure the files have been created and that Microsoft SQL Server has been restarted, if necessary.

Troubleshooting

Consider these issues when troubleshooting the sybfilter driver.

Table 26. Known issues for the sybfilter driver

Problem description
System environment variable is not set. **Problem**: The management console reports an error similar to: `ERROR: System environment variable RACFGFilePath has not been set. Please set its value before starting this manager. Fatal error occurs. Please press any key to quit.` **Workaround**: Set the *RACFGFilePath* environment variable.
Configuration file does not exist. **Problem**: In response to the **list** command, the management console reports: `ERROR: Cannot open config file.` **Workaround**: Create a configuration file.
Configuration file is not writeable. **Problem**: In response to the **add** command, the management console reports: `ERROR: Cannot open config file.` **Workaround**: Add write permission for the configuration file.
Microsoft SQL Server log files are locked. **Problem**: After restarting the machine on which Replication Agent for Microsoft SQL Server resides, you cannot open the Microsoft SQL Server log files because they are locked. **Workaround**: Restart the sybfilter management console. Issue the **stop** command followed by the **start** command to restart the sybfilter driver. Restart the primary Microsoft SQL Server data server.

Using the Trace Log

Use sybfilter trace log information to diagnose and troubleshoot problems.

1. Turn on tracing from the sybfilter management console with the **trace** command and the appropriate trace flag. For example, to find out why a Microsoft SQL Server log file is unreadable after a restart, turn on tracing with the **T3** flag before restarting Microsoft SQL Server:

   ```
   trace T3
   ```

2. Open the sybfilter trace log file `sybfilter.trc` to view logged messages.
3. Turn off tracing from the sybfilter management console:

   ```
   trace off
   ```

sybfilter Command Reference

These commands are available in the sybfilter management console. For a list and description of commands, enter the **help** command at the sybfilter management console.

add

Add a log file path to the sybfilter driver and configuration file.

Syntax

```
add serverName dbName logFilePath
```

Parameters

- **serverName –** the name of the Microsoft SQL Server.
- **dbName –** the name of the database to be replicated.
- **logFilePath –** the path of the database log.

check

Check whether the sybfilter driver is running. Check for differences between path names in the configuration file and the sybfilter driver. Check whether configuration files for sybfilter are readable, and list any files that are not readable.

Syntax

```
check
```

exit

Exit from the sybfilter management console.

Syntax

```
exit
```

help

Print help information for all sybfilter commands.

Syntax

```
help
```

list

List all configured database names and the corresponding log file paths in the configuration file.

Syntax

```
list
```

refresh

Refresh the content in the sybfilter configuration file.

Syntax

```
refresh
```

remove

Remove a log file path from the sybfilter driver and configuration file.

Syntax

```
remove logFilePath
```

Parameters

- **logFilePath –** path of the database log.

start

Start the sybfilter driver.

Syntax

```
start
```

stop

Stop the sybfilter driver.

Syntax

```
stop
```

trace

Trace sybfilter driver execution.

Syntax

```
trace [T1] [T2] [T3] [T4] | all | off
```

Parameters

- **T1 –** log routine trace messages.
- **T2 –** log operation status informational messages.
- **T3 –** log normal messages.
- **T4 –** log error messages.
- **all –** log all messages for the T1, T2, T3, and T4 flags.
- **off –** turn tracing off.

Glossary

This glossary describes SAP® Replication Server® Options terms.

- **SAP Adaptive Server –** the brand name for SAP relational database management system (RDBMS) software products.
  - SAP® Adaptive Server® Enterprise manages multiple, large relational databases for high-volume online transaction processing (OLTP) systems and client applications.
  - SAP® IQ manages multiple, large relational databases with special indexing algorithms to support high-speed, high-volume business intelligence, decision support, and reporting client applications.
  - SAP® 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 to a replicate database through the network in a single atomic operation, using a select operation with a holdlock. No changes to primary data are allowed until data transfer is complete. See also *bulk materialization* and *nonatomic materialization*.
- **BCP utility –** a feature that improves SAP Replication Server performance when replicating large batches of insert statements on the same table in SAP Adaptive Server Enterprise supported versions. SAP Replication Server implements bulk copy-in in Data Server Interface (DSI), the SAP Replication Server module responsible for sending transactions to replicate databases, using the SAP Open Client Open Server™ Bulk-Library. 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. For example, data may be transferred from a primary database using media such as magnetic tape, diskette, CDROM, or optical storage disk. Bulk materialization involves a series of commands, starting with define subscription. 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 SAP Replication Server or Replication Agent to manage the database and send or receive transactions from or to a database. Each database in a replication system can have only one database connection in SAP Replication Server and one primary database connection in Replication Agent. See also *SAP 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 –** 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*.
- **Direct Load Materialization –** an automatic materialization method to consume data between different primary and replicate databases. Data is loaded directly from a primary table into a replicate table, no materialization queue is used. See also *materialization*, *atomic materialization*, *bulk materialization*, and *nonatomic materialization*.
- **ERSSD –** Embedded Replication Server System Database

  The SAP SQL Anywhere database that stores Replication Server system tables. You can choose whether to store SAP Replication Server system tables on the ERSSD or the SAP Adaptive Server Enterprise RSSD. See also *SAP 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. SAP Replication Server distributes operations to replicate databases as functions. See also *stored procedure*.
- **function string –** a string that SAP Replication Server uses to map a function and its parameters to a data server API. Function strings allow SAP 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 SAP Replication Server to spool messages received from a Replication Agent. See also *outbound queue* and *stable queue*.
- **interfaces file –** a file containing entries that define network access information for server programs in an SAP client/server architecture. Server programs may include SAP Adaptive Servers, gateways, Replication Servers, and Replication Agents. The interfaces file entries enable clients and servers to connect to each other in a network. See also *Open Client* and *Open Server*.
- **isql –** an Interactive SQL client application that can connect and communicate with any SAP® Open Server™ application, including SAP Adaptive Server Enterprise, Replication Agent, and SAP Replication Server. See also *Open Client* and *Open Server*.
- **Java –** an object-oriented programming language developed by Oracle (formerly Sun Microsystems). A platform-independent, “write once, run anywhere” programming language.
- **JVM –** Java Virtual Machine

  the JVM is the part of the Java Runtime Environment (JRE) that is responsible for interpreting Java byte codes. See also *Java* and *JRE*.
- **JDBC –** Java Database Connectivity

  JDBC is the standard communication protocol for connectivity between Java clients and data servers. See also *data server* and *Java*.
- **JRE –** Java Runtime Environment

  the JRE consists of the Java Virtual Machine (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 –** 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 –** 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 *LTM*.
- **LTI –** Log Transfer Interface

  an internal component of Replication Agent that interacts with SAP Replication Server to forward transactions for distribution to Replication Server. See also *Log Reader* and *LTM*.
- **LTL –** Log Transfer Language

  a subset of the Replication Command Language (RCL). A Replication Agent such as RepAgent uses LTL commands to submit to SAP Replication Server the information it retrieves from primary database transaction logs. See also *Log Reader* and *LTI*.
- **LTM –** 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 *LTI*.
- **maintenance user –** a special user login name in the replicate database that SAP Replication Server uses to apply replicated transactions to the database. See also *replicate database* and *SAP 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*.
- **MPR –** Multi-Path Replication™

  SAP Replication Server feature that improves performance by enabling parallel paths of data from the source database to the target database. You can configure multi-path replication in warm standby and multisite availability (MSA) environments. These multiple paths process data independently of each other and are applicable when sets of data can be processed in parallel without transactions consistency requirements between

them while still maintaining data consistency within a path, but not adhering to the commit order across different paths.

- **nonatomic materialization –** a materialization method that copies subscription data from a primary to a replicate database through the network in a single operation, without a holdlock. Changes to the primary table are allowed during data transfer, which may cause temporary inconsistencies between replicate and primary databases. Data is applied in increments of ten rows per transaction, which ensures that the replicate database transaction log does not fill. Nonatomic materialization is an optional method for the **create subscription** command. Contrast with *atomic materialization*. See also *bulk materialization*.
- **ODBC –** Open Database Connectivity

  an industry-standard communication protocol for clients connecting to data servers. See also *client*, *data server*, and *JDBC*.
- **Open Client –** an SAP product that provides customer applications, third-party products, and other SAP products with the interfaces needed to communicate with Open Server applications. See also *Open Server*.
- **Open Client application –** An application that uses SAP Open Client libraries to implement Open Client communication protocols. See also *Open Client* and *Open Server*.
- **SAP Open Server –** an SAP 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 SAP Open Server libraries to implement Open Server communication protocols. See also *Open Client* and *Open Server*.
- **outbound queue –** a stable queue managed by SAP 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 –** a state in which log scanning has stopped and all scanned records have been propagated to their destinations in a replication system. Some Replication Agent and SAP 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 –** 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 –** Replication Command Language

  the command language used to manage SAP Replication Server. See also *SAP Replication Server*.
- **RDBMS –** 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 an SAP Replication Server for distribution to a replicate database. See also *primary database* and *SAP 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 SAP Replication Server, includes information about the columns to be replicated and the

location of the primary table or stored procedure. See also *SAP Replication Server* and *subscription*.

- **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 SAP Replication Server to a replicate SAP Replication Server. Routes carry data-changing commands (including those for RSSDs) and replicated functions (database procedures) between separate SAP Replication Servers. See also *SAP Replication Server*.
- **RSSD –** Replication Server System Database

  the SAP Adaptive Server Enterprise (SAP ASE) database containing an SAP Replication Server system tables. The user can choose whether to store Replication Server system tables on SAP ASE or embedded in an SAP SQL Anywhere database hosted by SAP Replication Server. See also *SAP Replication Server*.
- **SAP Replication Server –** an SAP software product that provides the infrastructure for a transaction replication system. See also *Replication Agent*.
- **SQL –** 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 SAP Replication Server. Messages written into the stable queue remain there until they can be delivered to the appropriate process or replicate database. SAP Replication Server provides a stable queue for both incoming messages (the inbound queue) and outgoing messages (the outbound queue). See also *database connection*, *SAP 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 SAP 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 *SAP 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 *SAP 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 –** wide-area network

  a system of local-area networks (LANs) connected together with data communication lines. Contrast with *LAN*.

Index

O

P

R

S

T

U

W