

New Features Replication Agent™ 12.6

Document ID: DC00503-01-1260-02

Last revised: September 2006

Topic	Page
Command for migrating Replication Agent	2
ra_migrate	2
Commands for creating and removing replication definitions (Oracle only)	2
rs_create_repdef command	2
rs_drop_repdef command	4
pdb_auto_create_repdefs property	4
Configuration parameter for automatically marking tables (Oracle only)	5
pdb_automark_tables	6
Marking Oracle Sequences for replication	7
pdb_setrepseq command	7
Understanding Oracle Sequence replication	9
Specifying the owner of a table in a replication definition	11
Support for Microsoft SQL Server 2005 JDBC driver	12
Using the Microsoft SQL Server JDBC Driver	12
Using Windows authentication with the Microsoft SQL Server JDBC driver	14

Copyright 1998-2006 by Sybase, Inc. All rights reserved. Sybase, SYBASE (logo), ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Advantage Database Server, Afaria, Answers Anywhere, Applied Meta, Applied Metacomputing, App-Modeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, ASEP, Avaki, Avaki (Arrow Design), Avaki Data Grid, AvantGo, Backup Server, BayCam, Beyond Connected, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, ComponentPack, Connection Manager, Convo/D, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DataWindow .NET, DB-Library, dbQueue, DeJima, DeJima Direct, Developers Workbench, DirectConnect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRT, eFulfillment Accelerator, EII Plus, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, ExtendedAssist, Extended Systems, ExtendedView, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Legion, Logical Memory Manager, IriLite, M2M Anywhere, Mach Desktop, Mail Anywhere Studio, Mainframe Connect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Anywhere, M-Business Channel, M-Business Network, M-Business Suite, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, mFolio, Mirror Activator, MI Query, MobiCATS, MobileQ, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASIS, OASIS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniQ, OmniSQL Access Module, OmniSQL Toolkit, OneBridge, Open Biz, Open Business Interchange, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, Pharma Anywhere, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power++, Power Through Knowledge, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Pylon, Pylon Anywhere, Pylon Application Server, Pylon Conduit, Pylon PIM Server, Pylon Pro, QAnywhere, Rapport, Relational Beans, RemoteWare, RepConnector, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RFID Anywhere, RW-Display-Lib, RW-Library, SAFE, SAFE/PRO, Sales Anywhere, Search Anywhere, SDF, Search Anywhere, Secure SQL Server, Secure SQL Toolset, Security Guardian, ShareLink, SharePool, SKILLS, smart.partners, smart.parts, smart.script, SOA Anywhere Trademark, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQL Stage III Engineering, Startup.Com, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Unimul, Unisep, Unistrain, URK Runtime Kit for UniCode, Viafone, Viewer, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, XP Server, XTNDAccess and XTNDConnect are trademarks of Sybase, Inc. or its subsidiaries. 07/06

Command for migrating Replication Agent

You can use the Replication Agent™ `ra_migrate` command to migrate the installed Replication Agent version to the current version.

ra_migrate

Description	Performs migration tasks (as necessary) between releases of Replication Agent.
Syntax	<code>ra_migrate</code>
Usage	• When you apply an EBF of the Replication Agent release, you may be required to run this command to upgrade to the latest version of the Replication Agent. • When <code>ra_migrate</code> is invoked, it migrates the installed Replication Agent version to the current version. • The <code>ra_migrate</code> command updates the <i>XLog system</i> table with the latest build version if the Replication Agent was previously initialized. • This command does not support migration to an earlier version of Replication Agent. • This command must be run in ADMIN mode.

Commands for creating and removing replication definitions (Oracle only)

(This section and subsections apply to Oracle only.) Sybase has added a new feature for creating and removing replication definitions at Replication Server. This feature includes commands that allow replication definitions to be automatically created when tables or procedures are marked, and to be removed when they are unmarked.

***rs_create_repdef* command**

Description	Creates a replication definition at Replication Server® for a designated marked table or procedure, or for all marked tables and procedures.
-------------	--

Syntax	<code>rs_create_repdef [all <i>TABLE_NAME</i>]</code>
Parameters	<code>all</code> A replication definition is created for all tables and procedures that are marked for replication. <i>TABLE_NAME</i> A replication definition is created for that table or procedure.
Usage	- When <code>rs_create_repdef</code> is invoked and the parameter <code>all</code> or <code>ALL</code> is entered, a replication definition is created for all tables or procedures that are marked for replication. - When <code>rs_create_repdef</code> is invoked and the name of a table or procedure that is marked for replication is entered, a replication definition is created for that table or procedure. - For each table or procedure for which a replication definition of create is attempted, a result set is returned. The result set contains the replication definition name and status of the create. If the replication definition was created, the status will be created. If an error occurred, an error message from Replication Server or Replication Agent will be returned. - The <code>lcl_character_case</code> configuration parameter setting determines the character case of the object names in the replication definition. - Replication definition names for tables always begin with the prefix <i>ra\$</i>, followed by an alphanumeric value and the replicate table name. (See the following examples.) All non-alphanumeric characters except the underscore and period are removed from the replicate table name. The period (".") is converted to an underscore, and the resulting name is truncated, if necessary, to 24 characters (<i>repdef</i> names are limited to 30 characters.) The following table identifies examples of replicate names and their resultant repdef names.

Table 1: Replicate names and their resultant repdef names

Table name	Replication definition name
order\$req	ra\$0x794a_orderreq
orderreq	ra\$0x7952_orderreq
my.orderreq	ra\$0xa154my_orderreq

- The replication definition name for a procedure is the same name as the procedure.

***rs_drop_repdef* command**

Description	Drops a replication definition for a table or procedure at the Replication Server.
Syntax	<code>rs_drop_repdef [TABLE_NAME]</code>
Parameters	TABLE_NAME A replication definition is dropped for a table or procedure.
Usage	• When <code>rs_drop_repdef</code> is invoked, a replication definition for that table is dropped at the Replication Server. • For each table or procedure for which a replication definition is dropped, a result set is returned. The result set contains the table name or procedure name and the status of the drop. If the replication definition was dropped, the status will be “dropped.” If an error occurred, an error message from Replication Server or Replication Agent is returned. • The <code>ltl_character_case</code> configuration parameter setting determines the character case of the object names in the replication definition.

***pdb_auto_create_repdefs* property**

Description	Allows replication definitions to be automatically created and dropped at Replication Server for tables and procedures when they are marked and unmarked for replication.
Default	false
Values	true – Replication definitions are automatically created at the Replication Server when tables or procedures are marked. false – No replication definitions are created when tables or procedures are marked.
Comments	When this property is set to true: • The <code>pdb_setreptable</code> is invoked to mark a table or tables, and a replication definition is created at Replication Server for each table that is marked for replication. • The <code>pdb_setrepproc</code> is invoked to mark a procedure, and a replication definition is created at Replication Server for each procedure that is marked for replication. • The <code>pdb_setreptable</code> is invoked to unmark a table, and the replication definition is dropped at Replication Server for each table that is unmarked for replication.

- The `pdb_setrepproc` is invoked to unmark a procedure or procedures, and a replication definition is dropped at Replication Server for each procedure that is unmarked for replication.
- The `pdb_xlog` is initialized and table auto-marking is enabled, and a replication definition is created for each table that is marked for replication.
- Replication definition names for tables always begin with the prefix `ra$`, followed by an alphanumeric value and the replicate table name. (See the following examples.) All non-alphanumeric characters except the underscore and period are removed from the replicate table name. The period (".") is converted to an underscore, and the resulting name is truncated, if necessary, to 24 characters, (*repdef* names are limited to 30 characters.)

The following table identifies examples of replicate names and their resultant *repdef* names.

Table 2: Replicate names and their resultant repdef names

Table name	Replication definition name
order\$req	ra\$0x794a_orderreq
orderreq	ra\$0x7952_orderreq
my.orderreq	ra\$0xa154_my_orderreq

- A replication definition name for a procedure is the same name as the procedure.

Configuration parameter for automatically marking tables (Oracle only)

The configuration parameter `pdb_automark_tables` now determines whether or not the Replication Agent automatically marks user tables for replication. If the value is set to true, user tables are marked during initialization and DDL replication (when replication of DDL commands is enabled).

pdb_automark_tables

Determines if the Replication Agent automatically marks tables for replication during initialization or DDL replication (for Oracle only).

Default

false

Values

true – User tables are automatically marked during initialization or DDL replication.

false – User tables are not automatically marked during initialization or DDL replication. They must always be marked using the `pdb_setreptable` command (default).

Comments

- The default value for `pdb_automark_tables` is set to false when a Replication Agent instance is created. In this default setting, tables are never automatically marked for replication. To start or implement automatic marking of tables, changed the configuration parameter to true. When set to true, all user tables (those whose owners are not contained in the `owner_filter_list`) are marked for replication when the `pdb_xlog` command is executed with the `init` keyword. In addition, when replication of DDL commands is enabled (`pdb_setrepddl` setting is disabled by default), any create table command for a user table (those whose owners are not contained in the `owner_filter_list`) are automatically marked for replication.
- Automatic marking of new tables (those created in the primary database with the create table command) occurs only when replication of DDL commands is enabled (`pdb_setrepddl` is set to enable), and the table is a user table (those whose owners are not contained in the `owner_filter_list`), and `pdb_automark_tables` is set to true.
- Tables are automatically unmarked for replication when a drop table command is issued at the primary database and the operation is recorded in the *redo* log, regardless of the settings of `pdb_setrepddl` or `pdb_automark_tables`. This is because a dropped table cannot be replicated from.
- Automatic marking of user tables operates independently from manual marking of tables using the `pdb_setreptable` command. In other words, you can always mark or unmark individual tables or all tables for replication using the `pdb_setreptable` command, regardless of the setting of `pdb_automark_tables`.
- If the value is set to false, user tables are never automatically marked; they must be explicitly marked using the `pdb_setreptable` command.

Marking Oracle Sequences for replication

(This section and subsections apply only to Oracle.) Similar to the support for table and procedure replication, Oracle Sequences are individually marked for replication using the new Replication Agent `pdb_setrepseq` command.

pdb_setrepseq command

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

Syntax To return sequence replication marking status:

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

To unmark, enable, or disable all marked sequences:

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

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

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

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

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

Parameters *sequence_name*

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

```
"Sequence"
```

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

```
owner.sequence
```

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

repname

The replicated name of the sequence to be updated at the replicate site, if desired to be different than the sequence name at the primary site. The *repname* option can be delimited with quote characters to specify character case. (See the previous description of the *sequence_name* parameter for details.) By specifying a replicated name, sequence updates can be replicated to a sequence in the replicate database that has a different sequence name from the primary database.

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

`repowner.repname`

all

A keyword that refers to all user sequences in the primary database. By using the *all* keyword, you can unmark all user sequences, or apply an enable or disable operation to all marked sequences.

mark

A keyword that refers to marking user sequences for replication.

unmark

A keyword that refers to unmarking user sequences for replication.

force

A keyword that refers to the unmark operation. When the *force* keyword follows the *unmark* keyword, the `pdb_setrepseq` command immediately unmarks the specified sequence in the primary database, without first checking the enable status of the sequence. When the *force* keyword follows the *unmark* keyword and the *all* keyword, the `pdb_setrepseq` command immediately removes replication marking from all marked sequences in the primary database, regardless of their enable status.

enable

A keyword that refers to enabling replication for marked sequences.

disable

A keyword that refers to disabling replication for marked sequences.

Usage

- When `pdb_setrepseq` is invoked, its function is determined by the keywords and options you specify.

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

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

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

Understanding Oracle Sequence replication

The following section describes Oracle Sequence replication.

Logging of Oracle Sequence information

Individual sequence changes are not logged in the Oracle database *log* file; however, changes to Oracle Sequences do impact (update) the Oracle `sys.seq$` table. These changes do not occur with each new sequence value generated. Instead, the `sys.seq$` table is updated periodically, based on sequence caching refresh activity or other system changes. The value stored in the `sys.seq$` table for a sequence is the “next” value to be assigned “after” the existing cache of values has been exhausted.

For example, a newly created sequence starts with a value of 1, increments by 1, and has a cache value of 20. (These are all default values and can be customized.) The value stored in the `sys.seq$` record for this new sequence is 21. This indicates that the “next” value to be used by the sequence, after the existing cache of 20 numbers is used, is 21. The record in `sys.seq$` does not change until the sequence value hits 21. At that time, Oracle will cache the next 20 values for the sequence, and the `sys.seq$` record will be updated to 41. It is this value (41), recorded in change to the sequences `sys.seq$` record, that will be used for replication. The key point is to recognize that not every individual sequence update is recorded in the log and therefore is not available for replication.

Replicating sequence changes

When a sequence is marked for replication, changes to that sequence against `sys.seq$` are captured and sent to Replication Server in the form of parameters passed to a procedure. The procedure (`rs_update_sequence`) must be installed at the standby site as part of system setup, as well as a function replication definition for that procedure. At the standby site, an implementation of `rs_update_sequence` will increment a same-named sequence until its value is equal to the value at the primary site. Scripts are provided with installation to create the `rs_update_sequence` stored procedure and function replication definition and are located as follows:

```
$SYBASE/RAX-12_6/scripts/  
oracle_create_replicate_sequence_proc.sql  
$SYBASE/RAX-12_6/scripts/  
oracle_create_rs_sequence_repdef.sql
```

Performance considerations

Compared to the performance of incrementing a sequence at the primary database, particularly where sequence values are cached, the effort to increment the same sequence at the standby site may be less efficient. The stored procedure must dynamically determine the sequence to increment and must loop internally, incrementing the sequence until the primary value has been reached. The loop is required because there is no way to assign a specific value to a sequence.

Because the name of the sequence is passed as a parameter, Oracle cannot pre-compile the procedure for efficiency. With the addition of the looping activity required to properly increment the sequence, the performance of the solution may impact some environments where a large number of highly used sequences is the norm.

Sequence replication alternatives

If the performance of sequence replication is a concern, other alternatives to replication are available that support primary and standby use of the same sequence. These alternatives are currently suggested by Oracle and others interested in providing sequence coordination between multiple sites:

- Assuming the sequence is being used to generate primary key values, the sequence at each site can be concatenated with something unique to the site. For example, use a sequence number concatenated with the database name, site name, or something similar. This technique allows each site to maintain a unique “range” of sequence of numbers. If each site has a unique range, there is no value in sending (replicating) changes of one site’s range to another site.
- Similar to concatenating, each site can obtain a different range of numbers by having different starting points, or increment values, for the same sequence. For example, the sequence at one site can start at 1 and increment by 2 to generate odd numbers (1, 3, 5), while the other site starts at 2 and generates even numbers (2, 4, 6). Again, each site would have a unique range and would avoid any need for replication.
- A third option is available to standby solutions, where the standby site is for read-only and does not access the sequence value until failover. Rather than continually replicating a sequence’s value, the value of the sequence at the standby site can be updated as part of the failover tasks. After failover and before the standby allows connection to client applications, a script or procedure can query the last-used sequence value (based on the last table to use it for a primary key) and update or redefine the sequence once, based on that calculated value.

Specifying the owner of a table in a replication definition

How you specify the owner of the table in a replication definition has changed from that described in the *Reference Manual*. Now, you must always use the `owner` keyword if you want to enable the `SEND OWNER` mode. When marking, if you do not specify the replicate owner, it will default to the primary owner.

In the following examples, the leading letter indicates either primary (p) or replicate (r):

- Example 1:

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

The table in the replication definition will be:

```
powner.rtable
```

- Example 2:

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

The table in the replication definition will be:

```
rowner.rtable
```

- Example 3:

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

The table in the replication definition will not be owner-qualified:

```
rtable
```

Support for Microsoft SQL Server 2005 JDBC driver

(For Microsoft SQL Server only) Replication Agent now supports the Microsoft SQL Server 2005 JDBC driver version 1.1 (released June 2006) along with the DataDirect SQL Server JDBC driver that is distributed with Replication Agent.

Note To replicate from Microsoft SQL Server 2005 you must use the Microsoft SQL Server JDBC driver. To replicate from Microsoft SQL Server 2000, Sybase recommends that you also use the Microsoft SQL Server JDBC driver instead of the DataDirect SQL Server JDBC driver that is distributed with Replication Agent.

If you want to continue to use the DataDirect SQL server JDBC driver, you must set the Replication Agent property named `pds_connection_type` to "MSMERJDBC" (without the quotes). For additional information, see the Replication Agent *Administration Guide*.

Using the Microsoft SQL Server JDBC Driver

By default, after installing ESD #5 or later, the new Replication Agent for Microsoft SQL Server instances that you create are configured to use the Microsoft SQL Server JDBC driver. You can use this driver on both Windows and UNIX systems.

❖ To use the Microsoft SQL Server JDBC driver

- 1 Retrieve the Microsoft SQL Server JDBC driver at <http://msdn.microsoft.com/data/ref/jdbc/> (enter the URL all on one line), and install the JDBC driver on the same machine on which the Replication Agent is running or where Replication Agent can access it.
- 2 Before starting the Replication Agent, add the location of the JDBC driver to the CLASSPATH environment variable:

- On UNIX, enter:

```
setenv CLASSPATH /path_name/sqljdbc.jar:$CLASSPATH
```

where *path_name* is where you installed the Microsoft SQL Server JDBC driver.

- On Windows 2000 or 2003, go to Start | Settings | Control Panel | System | Environment, and add the following to the existing CLASSPATH environment variable, using the semicolon (;) as the path separator, or create the path in the User Variables panel by entering the following:

```
drive:\path_name\sqljdbc.jar
```

where:

- *drive* is the drive letter
- *path_name* is the name of the path where you installed the Microsoft SQL Server JDBC driver

Click Apply and then OK.

- 3 For an existing Replication Agent for Microsoft SQL Server instance to retrieve the new value for the CLASSPATH environment, do the following:
 - On UNIX, stop the Replication Agent instance, define the CLASSPATH as described in step #2, and then restart the instance.
 - On Windows, stop the Replication Agent, close the DOS window, define the CLASSPATH as described in step #2, open a new DOS window, and then restart the Replication Agent instance.
- 4 For an existing Replication Agent for Microsoft SQL Server instance, in ADMIN mode, configure the *pds_connection_type* to use the Microsoft SQL Server JDBC driver:

```
ra_config pds_connection_type, MSSQLJDBC
```

Then stop and restart the existing instance.

Note For new instances of Replication Agent for Microsoft SQL Server that you create after applying ESD #5 or later, MSSQLJDBC is the default value.

- 5 In Replication Agent, configure the following properties 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_password, <ra_pwd>
```

For more information, see the Replication Agent *Administration Guide*.

Using Windows authentication with the Microsoft SQL Server JDBC driver

When running Replication Agent for Microsoft SQL Server on a Windows platform, you have the option of configuring Replication Agent for Microsoft SQL Server to connect to Microsoft SQL Server to use Windows credentials to authenticate the user.

Note This feature is available only when Replication Agent for Microsoft SQL Server is running on Windows 2000 or Windows 2003 and requires that the primary database be either Microsoft SQL Server 2000 SP 3 or later, or Microsoft SQL Server 2005.

❖ To use Windows Authentication

- 1 In your primary Microsoft SQL Server, add the *<ra_user>* as a Windows-authenticated login, including the user's domain as appropriate. Be sure to also add the *<ra_user>* to the primary database and grant appropriate permissions. For details, refer to your 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, then add only `<ra_user>` to the Windows user account.
- 3 On the machine on which the Replication Agent for Microsoft SQL Server is running, 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 the following 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 Log in as the `<ra_user>` on to the machine on which Replication Agent for Microsoft SQL Server will be running and start the Replication Agent for Microsoft SQL Server instance.
- 5 Log in to Replication Agent and configure the connectivity properties described previously (in step #5); however, you can omit `pds_password`.
- 6 Configure the Replication Agent property, `pds_integrated_security` to "true" (default is "false"). For example:

```
ra_config pds_integrated_security, true
```
- 7 Continue with other Replication Agent configuration as described in the Replication Agent *Administration Guide*.

