Synchronized Database Service (SDS) Administration Guide

VERSION 7.1.08

jade™

Copyright©2016 Jade Software Corporation Limited. All rights reserved.

Jade Software Corporation Limited cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages, or loss of profits. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Jade Software Corporation Limited.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Copyright © 2016 Jade Software Corporation Limited.

All rights reserved.

JADE is a trademark of Jade Software Corporation Limited. All trade names referenced are the service mark, trademark, or registered trademark of the respective manufacturer.

For details about other licensing agreements for third-party products, you must read the JADE **ReadMe.txt** file.

Contents

Contents iii

Before You Begin vii

Who Should Read this Guide vii
What's Included in this Guide vii
Related Documentation vii
Conventions viii

Chapter 1 Administering a JADE Synchronized Database Service (SDS) Environment 9

Overview 9
Implementing a Database Recovery Strategy 10
Disaster Recovery 10
Offline Inquiry Processing 11
Delta Databases 11
SDS and RPS Delta Database Operational Characteristics 12
Database Tracking and RPS Replication 12
Administrative Restrictions 12
SDS_ReasonDeltaModeEntered Tracking Stopped Reason Code 13
SDS and Recovery Considerations when Using Partitioned Database Files 13
RPS Considerations 14
Synchronized Database Service Functionality 14
Database Roles in a Synchronized Database Environment 15
Primary Database Role 15
Secondary Database Role 16
Subroles 16
Synchronizing with a Primary 16
Selecting the Journal Synchronization Mode for a Secondary Database 16
Journal Block Write Mode 16
Journal Switch Mode 17
Synchronization Mode and Disaster Recovery 17
Disaster Recovery in Journal Block Write Mode 18
Disaster Recovery in Journal Switch Mode 18
Restart Recovery 18
Delayed Replay 18
Deferring SDS Transactions 19
Version Information 19
Security and Authentication 19
SDS Takeover Operations 20
Negotiated Takeovers 20
Hostile Takeovers 21
Considerations before Performing a Takeover 21
Preparing for a Takeover Operation 21
Performing a Negotiated Takeover 22
Performing a Hostile Takeover 22
Backing Up and Restoring Secondary Databases 22
Quiesced Database Backup 22
Online Database Backup 22
Offline Database Backup 23
Creating a Secondary Database from an Online Backup of a Primary Database 23
Reorganizing an SDS Database 23
Configuring the Synchronized Database Service 24
[ConnectionParams] Section 24
[SyncDbService] Section 24
JADE Initialization File Configuration Examples 26
Example 1 26
Example 2 27
Creating your First SDS Environment 29
Recommendations 29

Assumptions 29
Creating an SDS Environment 29
Using the SDS Administration Application 30
Administering and Monitoring a Primary Database 32
Administering and Monitoring a Secondary Database 34
Using SDS Administration Application Menus 36
Using the Admin Menu 36
Taking Over the Primary Database Role 37
Making a Database with an Undefined Role the Primary Database 38
Refreshing the SDS Admin Database View 38
Closing the SDS Administration Application 38
Using the Primary Menu 38
Stopping and Starting a Synchronized Database Service 39
Stopping Audit Tracking 39
Closing the Current Journal 39
Using the Secondary Menu 39
Enabling or Disabling Read Access of Persistent Objects in the Database 40
Enabling or Disabling Tracking 40
Taking Over of the Primary Database Role from a Secondary Database 41
Resuming Journal Replay 42
Replaying the Next Journal 43
Disabling a Connection to the Primary Database 43
Clearing a Disrupted Network Connection 43
Reconnecting a Secondary Database 43
Using the Options Menu 44
Automatically Refreshing the SDS Admin Database View 44
Displaying the Legend for the Database State 44
Changing the Font Displayed in the SDS Admin Database View 45
Specifying the Number of Seconds at which the Database View Refreshes 45
Saving Settings when Exiting from the SDS Admin Application 46
Using the Help Menu 46
Accessing the Help Index 46
Accessing Information about the SDS Administration Utility Release 46

Chapter 2 Relational Population Service (RPS) Support 48

Overview 48
RPS Node Creation Example 49
RPS Mapping 50
Primary Keys 51
Default Mapping Values 51
Entity Name Mapping 51
Class Mappings 51
Attribute Mapping 51
Method Mapping 51
OID Columns 51
Edition 52
Primary Keys 52
Embedded References 52
Special Columns 52
Operation Column 52
Timestamp Column 52
TranID Column 52
Type Column 52
Special Tables 52
Exception Logging Table 53
Transaction Table 53
Control Table 53
JADE_TRAN_INFO Table 53
JADE to RDBMS Type Mapping 53

Contents v

JADE to SQL Server Type Mapping 53
Mapping Time, TimeStamp, and Date Primitive Types to the SQL Server DATETIME Type 56
RPS Node 57
RPS Participant in an SDE 57
RPS Data Store 57
Full Database Replica 58
Mapped Extent 58
Working Set 58
RPS Datapump Application 59
Implementing User-Defined Output Control 61
Column-Mapping Methods 62
Handling Exceptions in Column-Mapping Methods 62
Table Creation and Alter Scripts 63
Replicating Data 63
Controlling Replication from the Primary 64
Filtering Transactions from the Primary System 64
Schema Instantiation 64
Alter Scripts 65
Datapump Application Execution 66
Restarting the Datapump Application 67
Automatically Initiating Scripts 67
Automatically Initiating Drop/Reload Scripts 67
Manual Table Redefinition 68
Table Consistency Checking 69
Point-in-Time Snapshots 69
RPS Audited SQL Script Execution 69
Error Handling Policy 69
Handling Exceptions 70
Fault Handling when the Datapump Application Terminates Abnormally 70
Automatic Restart on Connection or Timeout Errors 71
Disabling Datapump Application Automatic Process Dumps 71
Exception Logging 72
Restart Handling 72
Managing an RPS Node 72
Using the RPS Manager Application 72
Selecting Your RPS Mapping 72
Using the File Menu 73
Configuring your RPS Node 74
Resetting the Relational Database Identifier 78
Checking the Consistency of the JADE and RDBMS Databases 79
Executing an SQL Script 79
Printing Text Displayed in the Jade RPS Manager Window 79
Selecting the Jade RPS Manager Window Display Font 79
Clearing Text from the Jade RPS Manager Window 80
Exiting from the RPS Manager Application 80
Using the RPS Menu 80
Creating the RPS Database 80
Creating an RPS Database from a Backup of the Primary 82
Starting the Data Pump 82
Stopping the Data Pump 83
Setting Up RDBMS 83
Creating Tables 83
Extracting and Loading Data 83
Backing Up the RPS Database 84
Using the Extracts Menu 85
Extracting Data from the JADE Database 85
Selecting Tables 88
Maintaining Extract File Names 90
Extracting Data and Loading it into the Relational Database 91
Loading Extracted Data into the Relational Database 91

Contents vi

Generating Table Creation SQL Scripts 92
Generating Table Alter SQL Scripts 93
Using the Help Menu 95
Accessing the Help Index 95
Accessing Information about the RPS Manager Application Release 95
Calling the RPS Manager from a Web Service Consumer 95
Site-Specific RPS Mapping Customization 99
Deploying Systems with Excluded Elements 99
Using Methods to Exclude RPS Mapping Elements 101
Relational Population Service (RPS) Restrictions 101
RootSchema-Defined Types and Features 102
Using Methods Defined in RootSchema Classes 102

Before You Begin

The *JADE Synchronized Database Service (SDS) Administration Guide* is intended as the main source of information when you are administering a JADE Synchronized Database System (SDS) environment, including Relational Population Services (RPS).

Who Should Read this Guide

The main audience for the *JADE Synchronized Database Service (SDS) Administration Guide* is expected to be system administrators.

What's Included in this Guide

The *JADE Synchronized Database Service Administration Guide* has two chapters.

Chapter 1	Covers monitoring and administering a JADE Synchronized Database Service (SDS) environment
Chapter 2	Covers considerations when using the Relational Population Service (RPS) to replicate a production JADE database to one or more Relational Database Management System (RDBMS) target databases

Related Documentation

Other documents that are referred to in this guide, or that may be helpful, are listed in the following table, with an indication of the JADE operation or tasks to which they relate.

Title	Related to…
JADE Database Administration Guide	Administering a JADE database
JADE Developer's Reference	Developing or maintaining JADE applications
JADE Development Environment Administration Guide	Administering the JADE development environment
JADE Development Environment User's Guide	Using the JADE development environment
JADE Encyclopaedia of Classes	System classes (Volumes 1 and 2), Window classes (Volume 3)
JADE Encyclopaedia of Primitive Types	Primitive types and global constants
JADE Initialization File Reference	Maintaining JADE initialization file parameter values
JADE Installation and Configuration Guide	Installing and configuring JADE
JADE Object Manager Guide	JADE Object Manager administration
JADE Thin Client Guide	Administering JADE thin client environments

Conventions

The *JADE Synchronized Database Service (SDS) Administration Guide* uses consistent typographic conventions throughout.

Convention	Description
Arrow bullet (❯❯)	Step-by-step procedures. You can complete procedural instructions by using either the mouse or the keyboard.
Bold	Items that must be typed exactly as shown. For example, if instructed to type **foreach**, type all the bold characters exactly as they are printed. File, class, primitive type, method, and property names, menu commands, and dialog controls are also shown in bold type, as well as literal values stored, tested for, and sent by JADE instructions.
Italic	Parameter values or placeholders for information that must be provided; for example, if instructed to enter *class-name*, type the actual name of the class instead of the word or words shown in italic type. Italic type also signals a new term. An explanation accompanies the italicized type. Document titles and status and error messages are also shown in italic type.
Blue text	Enables you to click anywhere on the cross-reference text (the cursor symbol changes from an open hand to a hand with the index finger extended) to take you straight to that topic. For example, click on the "Administering and Monitoring a Secondary Database" cross-reference to display that topic.
Bracket symbols ([ ])	Indicate optional items.
Vertical bar (|)	Separates alternative items.
`Monospaced font`	Syntax, code examples, and error and status message text.
ALL CAPITALS	Directory names, commands, and acronyms.
SMALL CAPITALS	Keyboard keys.

Key combinations and key sequences appear as follows.

Convention	Description
KEY1+KEY2	Press and hold down the first key and then press the second key. For example, "press SHIFT+F2" means to press and hold down the SHIFT key and press the F2 key. Then release both keys.
KEY1,KEY2	Press and release the first key, then press and release the second key. For example, "press ALT+F,X" means to hold down the ALT key, press the F key, and then release both keys before pressing and releasing the X key.

In this document, the term Microsoft *Windows* refers to Windows 10, Windows 8, Windows 7, Windows Server 2012, Windows Server 2008, Windows Vista, or Windows Mobile. When there are differences between the versions of Microsoft Windows, the specific version of Microsoft Windows is stated.

With the exception of the **jade.exe** program, when referring to program executables in this document, the **.exe** file suffix is omitted; for example, **jadclient** refers to **jadclient.exe**. Similarly, the **.dll** (Dynamic Link Library) suffix is omitted. For example, **jomos** refers to **jomos.dll**.

Chapter 1 Administering a JADE Synchronized Database Service (SDS) Environment

This chapter covers the following topics.

- Overview
  - Implementing a Database Recovery Strategy
  - Disaster Recovery
  - Offline Inquiry Processing
  - Delta Databases
  - SDS and Recovery Considerations when Using Partitioned Database Files
- Synchronized Database Service Functionality
- Configuring the Synchronized Database Service
  - [ConnectionParams] Section
  - [SyncDbService] Section
  - JADE Initialization File Configuration Examples
- Creating your First SDS Environment
  - Recommendations
  - Assumptions
  - Creating an SDS Environment
- Using the SDS Administration Application
  - Administering and Monitoring a Primary Database
  - Administering and Monitoring a Secondary Database
  - Using SDS Administration Application Menus

For details about developing application using SDS, see Chapter 10, "Synchronized Database Service (SDS) Development Considerations", in the *JADE Developer's Reference*.

See also the *Synchronized Database Service (SDS)* white paper on the JADE Web site at https://www.jadeworld.com/developer-center/resource-library/white-papers, for details about sample usage scenarios, technology comparison, disaster recovery concepts, terminology, and so on.

Overview

The JADE Synchronized Database Service (SDS) environment enables you to:

- Keep secondary copies of a database synchronized with a primary updating copy.
- Support optional read-only access to secondary copies of the database.

An SDS environment is made up of one primary database server and one or more secondary database servers, which are usually hosted on different machines that may be in different geographic locations.

Any JADE database server node is capable of operating as the primary database server in a hot standby configuration.

Subject to licensing restrictions, a primary database server offers a passive communication port to accept connections from secondary servers and can accept multiple connections.

A secondary database server establishes a connection to its specified primary server to receive control and database tracking information. A secondary database:

- Cannot be directly updated by application programs
- Is modified only by the replay of audit records generated on the primary database server
- Can assume the role of the primary database if the primary database becomes unavailable

Notes You require a licence to run a Synchronized Database Environment (SDE).

JADE licenses are not transferred automatically between databases in an SDE. It is your responsibility to apply new licenses to any existing databases in an SDE. In addition, to ensure proper operation, you must apply the primary license to every secondary.

In an SDS environment, the secondary database server has a node stub that represents the primary server node. This node stub does not have the full functionality of a normal client; in particular, some statistical functions cannot be carried out on the node stub (for example, getting a node's cache statistics). Any attempt to do so now raises exception 1265 (*Environmental operation out of scope for process*).

You can use the value returned by the **Node** class **nodeRole** method to distinguish the node stub from standard client nodes. For the node stub, the returned value is the **Node** class **Role_Replay** constant (as opposed to the **Role_Standard** constant).

For details about the database, environment, and server identities, see "Database Identities", in Chapter 3 of the *JADE Database Administration Guide*.

Implementing a Database Recovery Strategy

You can use an SDS secondary database as a 'hot standby' server located on-site, to provide fast recovery in the event of local hardware failure.

Recovery using a standby server involves applying any outstanding transactions from active journals on the primary databases, if these remain accessible, and then switching processing to the secondary server while the primary hardware is repaired. (This is almost certainly quicker than the traditional methods of restoring from backup and then performing a roll-forward recovery.)

Disaster Recovery

Disaster recovery strategies require a means to quickly relocate processing to a site in a different geographic location to the primary processing center.

A hot standby server is an ideal solution for remote database mirroring, or Remote Database Backup (RDB), as it is sometimes called.

Without SDS, you can achieve a remote database backup by using 'volume level' hardware or software mirroring. These solutions are typically expensive to implement, in particular, to achieve acceptable performance very high-speed networks are essential.

SDS provides a cost-effective alternative that also supports read-only access on secondary databases. (Read-only access to a volume-replicated database is not viable using third-party mirroring solutions because of cache and disk image consistency issues.)

Note When a primary database or an offline copy of a primary database is opened, the current write journal (that is, the write journal that was open when the database was last closed) is mandatory.

Offline Inquiry Processing

A read-only secondary database enables you to offload inquiry-type processing (for example, JADE Report Writer-generated reports), which can have a significant impact on update transactions.

This allows the primary update database to operate with the contention for resources incurred by the inquiry workload removed.

Delta Databases

A delta database allows you to make non-permanent (or tentative) changes to an SDS primary database, an SDS secondary database, or an RPS database in read-only mode.

When you activate delta database mode, a new empty database called a *delta database* is created and the original database, which referred to as the *root database*, is converted to a read-only state.

When a database is in delta mode:

- All create, update, and delete operations on non-environmental persistent objects are re-directed to the delta database instead of being applied to the root database.
- When the root database is an SDS primary database, updates applied to the delta database are not replicated to any secondary databases.
- Because changes to meta data objects stored in **_user*.dat** files are not permitted, schema changes cannot be applied.
- When applications access objects are created or updated, they are fetched from the delta database.
- If objects that have been deleted in delta mode are accessed, exception 4 *(Object not found)* is raised.

The delta database is located in the **deltadb** subdirectory of the root database path.

When you deactivate delta database mode, the delta database is removed and all updates made in delta mode are discarded.

The delta database can be recovered; that is, if the server node is restarted while the delta database is active, any updates are still present when the node is available again.

To start using delta database functionality, server nodes must be specified to be delta database-capable, by specifying the **DeltaDatabaseCapable** command with a value of **true** in the [JadeServer] section of the JADE initialization file on the server node.

Set the **ResetDeltaModeOnRestart** parameter in the [JadeServer] section to **true** if on restart, you want the database server node to take the root database out of delta mode and restart in normal mode (that is, to *not* use the delta database). This effectively results in the contents of the delta database being lost on server restart, as it is always recreated whenever activated. By default, if the database server node is terminated while in delta mode, it attempts to activate the delta database when it restarts.

To activate or deactivate a delta database dynamically from your application logic:

- Call the **System** class **activateDeltaDatabase** method with the **activate** parameter set to **true** or **false**, respectively.
- Execute the respective **ActivateDeltaDb** or **DeactivateDeltaDb** command in the online JADE Database Administration utility (that is, **jdbadmin**) from a command script.

SDS and RPS Delta Database Operational Characteristics

This section describes delta database operational characteristics of SDS and RPS.

Database Tracking and RPS Replication

When an SDS native or SDS secondary database enters delta mode, database tracking is stopped. In an RPS node, the data pump is left active in an idle state.

When a secondary database exits delta mode, database tracking is restarted (if it is not disabled) and on an RPS node, the **Datapump** application is restarted if it is not running and the **AutoStartDataPump** parameter in the [JadeRps] section of the JADE initialization file is set to **true**.

Administrative Restrictions

When an SDS secondary or primary database is in delta mode, specific administrative operations invoked by calling SDS-related methods in the **JadeDatabaseAdmin** class are not permitted; attempts to execute a disallowed operation that is processed synchronously fail and an exception is raised.

Disallowed administrative operations initiated from a primary that are processed asynchronously by a secondary fail and error details are recorded in the secondary **jommsg*n*.log** file; however, the initiator does not receive an error response.

You cannot execute the following SDS secondary administrative operations from a secondary database that is in delta mode.

JadeDatabaseAdmin Class Method	SDS Administration Command
sdsStartTracking	Start Tracking
sdsResume	Resume
sdsReplayNextJournal	Replay Next Journal
sdsInitiateHostileTakeover	Initiate Hostile takeover

You cannot execute the following SDS secondary administrative operations from a primary database server when the selected secondary database is in delta mode.

JadeDatabaseAdmin Class Method	SDS Administration Command
sdsStartTrackingAt	Start Tracking (at selected secondary)
sdsResumeAt	Resume (at selected secondary)
sdsReplayNextJournalAt	Replay Next Journal (at selected secondary)
sdsInitiateTakeover	Initiate Takeover (to selected secondary)

You cannot execute the following SDS primary administrative operation from a primary database server when the primary database is in delta mode.

JadeDatabaseAdmin Class Method	SDS Administration Command
sdsInitiateTakeover	Initiate Takeover (to any secondary)

SDS_ReasonDeltaModeEntered Tracking Stopped Reason Code

When database tracking is stopped because the database entered delta mode, the reason code passed in the **userInfo** parameter to the **SDS_TrackingStopped** (17388) system event is defined by the **SDS_ReasonDeltaModeEntered** global constant (which has an Integer value of 12) in the **SDSStopTrackingCodes** category.

SDS and Recovery Considerations when Using Partitioned Database Files

Partitioned file structures and most meta data are replicated on SDS secondary databases. For details about database file partitioning, see Chapter 20, "Partitioning Database Files", in the *JADE Developer's Reference*.

You can change certain partition attributes such as location on the secondary in order to support a different storage strategy; for example, a tiered strategy on the primary and all partitions on the same default volume on the secondary.

Database tracking logic supports the replay of partitioned file operations, including file-level reorganization and compaction. Certain database file and file partition operations are replayed on SDS secondary databases but some are not. The operations that are replayed in SDS are also reapplied by roll-forward recovery.

The following tables list database file and file partition operations that modify state. The second column indicates whether the operation is replayable; that is, the operation will be audited and reapplied by roll-forward recovery and SDS secondary replay. The third column indicates whether the operation can be executed on an SDS secondary database. In general, if the operation is replayable, it is not valid to execute the operation directly to an SDS secondary. Conversely, if an operation is not replayable, the operation can be executed on a secondary database, allowing the affected state to differ from the primary database.

The following table lists database file operations.

Operation	Replayable	Valid on SDS Secondary?
Set Partitioned	Yes	No
Create Partition	Yes	No
Set Partition Modulus	Yes	No
Freeze	No	Yes
Thaw	No	Yes
Mark Offline	No	Yes
Mark Online	No	Yes

The following table lists file partition operations.

Operation	Replayable	Valid on SDS Secondary?
Freeze	No	Yes

Operation	Replayable	Valid on SDS Secondary?
Thaw	No	Yes
MarkOffline	No	Yes
MarkOnline	No	Yes
Move	No	Yes
SetLabel	Yes	No
SetLocation	No	Yes

When considering use of partitioned database files in a Synchronized Database Environment (SDE):

- A file or partition that is not frozen on a primary can be frozen on a secondary. If a file or partition is frozen on the secondary and updated on the primary, the file or partition must be located on writeable media to allow the database tracker to replay updates made to the primary version.
- It is possible and valid to have partitions:
  - Online on a secondary database that are offline on the primary. This could be useful if secondary query applications need access to historical archived data that was taken offline on the production primary.
  - Offline on a secondary database that are online on the primary, provided that the offline partitions are not updated on the primary database. Ideally, the partitions should be frozen on the primary to protect against accidental updates. If the SDS tracker encounters updates that must be applied to an offline file or partition, tracking will halt. User intervention will be required to bring the required file or partition online before resuming tracking.
- When SDS is being used to implement a disaster recovery strategy, you should ensure that backup copies of partitions that are taken offline at the primary or secondary sites can be restored at either site in the event of a disaster.

RPS Considerations

Partitioned file structures and meta data are replicated on full replica and mapped extent RPS databases.

File partitioning is not applicable on a working set RPS database.

Synchronized Database Service Functionality

This section, describing SDS functionality, contains the following topics.

- Database Roles in a Synchronized Database Environment
  - Primary Database Role
  - Secondary Database Role
  - Subroles
- Synchronizing with a Primary
- Restart Recovery
- Delayed Replay
- Deferring SDS Transactions

- Version Information
- Security and Authentication
- SDS Takeover Operations
  - Negotiated Takeovers
  - Hostile Takeovers
  - Considerations before Performing a Takeover
  - Preparing for a Takeover Operation
  - Performing a Negotiated Takeover
  - Performing a Hostile Takeover
- Backing Up and Restoring Secondary Databases
  - Quiesced Database Backup
  - Online Database Backup
  - Offline Database Backup
- Creating a Secondary Database from an Online Backup of a Primary Database
- Reorganizing an SDS Database

For details, see the following subsections.

Database Roles in a Synchronized Database Environment

In a Synchronized Database Environment, the terms *primary* and *secondary* signify the intended function or role of each copy of the database. A secondary database with the **Native** subrole:

- Cannot be updated by application programs
- Is modified only by the replay of audit records generated on the primary database server
- Can assume the role of the primary database if the primary database becomes unavailable

An SDE is made up of exactly one primary database server and one or more secondary database servers, usually hosted on different machines that may be in the same room or in entirely different geographic locations.

The configuration parameters required to initialize SDS modules are defined in the [SyncDbService] section of the JADE initialization file. For details, see "Synchronized Database Service Section [SyncDbService]" under "Synchronized Database Service Sections", in the *JADE Initialization File Reference*.

Primary Database Role

Any existing database server node is capable of operating as the *primary database server* in a hot standby configuration. This mode of operation is activated by parameters in the [SyncDbService] section of the JADE initialization file.

Primary server nodes must be run with the **server** parameter in the command line set to **remoteServer** (that is, multiuser) in order to get full SDS functionality. Running the primary node with the **server** parameter set to **localServer** (that is, single user) is useful for performing maintenance tasks, deployments, and so on (for example, when running the batch JADE Schema Load Utility **jadloadb**).

Note If an SDS primary server node is started in single user mode, full SDS functionality is not available as the SDS service does not get started. As there is no attempt to communicate with the SDS secondary server, functions such as calls to the **Object** class **sdeCauseEvent** and **sdsCauseEvent** methods will not send events to the reciprocal node.

Secondary Database Role

A secondary database server establishes a connection to its specified primary server to receive control and database tracking information.

Subroles

A secondary node can have one of the following subroles.

- **Native** (that is, the JADE subrole, documented in this chapter)
- **Relational** (that is, the RPS node subrole, documented in Chapter 2, "Relational Population Service (RPS) Support")

Synchronizing with a Primary

Whenever a secondary server initially connects to the primary database, it will almost certainly be behind in terms of applied transactions and it will need to catch up with the primary system. This synchronization is achieved by the replay of audit records in journals transferred from the primary to the secondary database.

A secondary database can be taken offline at any time for any purpose. When it reconnects to the primary system, the synchronization process is repeated. When a secondary database server is stopped or it crashes when transactions are in progress and it is subsequently restarted, the SDS server remains in a 'recovery required' state until all interrupted transactions have been resolved.

Interrupted transactions are resolved when the commit or abort audit records are eventually replayed. While interrupted transactions exist, read-access to the secondary database is not permitted.

Selecting the Journal Synchronization Mode for a Secondary Database

The two modes for journal synchronization are:

- Journal block write, which mirrors journal blocks on the secondary as they are written to the primary
- Journal switch, which transfers journals periodically

You can configure the mode of synchronization for individual secondary databases, by using the appropriate value in the **SyncMode** parameter in the [SyncDbService] section of the JADE initialization file. For more details about journal synchronization modes, see the following subsections.

Journal Block Write Mode

The *journal block write* synchronization mode (the default synchronization mode) writes journal blocks across a network connection to secondary databases as they are written to the primary database journal. In the journal block write mode, SDS can establish and maintain very close synchronization between primary and secondary databases.

When the secondary database is synchronized with the primary, audit blocks are transferred and written to the secondary database journal at the same time they are written to the journal on the primary. In effect, primary journal writes are mirrored to the secondary journal. These mirrored journal writes are performed asynchronously so that primary database transaction processing is not held up waiting for acknowledgements from secondary databases.

The principal benefits of using the journal block write synchronization mode are as follows.

- Synchronization of journals on the primary and secondary databases is closer than is possible with the journal switch mode.
- Loss of audit data is minimized and a disaster recovery therefore recovers most, if not all, transactions that committed on the primary database.
- Following a takeover, restoration of database access can be faster than with the journal switch mode.

A block write secondary is caught up to the primary when it is actively replaying in the current primary write journal.

Journal Switch Mode

The *journal switch* mode makes intermittent use of the network communications for the transmission of journals and control information. The journal is transferred to the secondary when the journal is complete and writing switches to a new journal.

The benefit is to provide journal shipping with a minimum demand on your network and processor capacity. You can control journal file size by using the **JournalMinSize**, **JournalMaxSize**, or **JournalSwitchInterval** parameter in the [PersistentDb] section of the JADE initialization file.

A journal switch secondary is synchronized when it is replaying the last complete journal. As long as a secondary is connected to the primary and communications are working, the secondary knows the current journal on the primary so that it can accurately report the state of the secondary in relation to the primary.

Synchronization Mode and Disaster Recovery

The mode of database synchronization that you choose is tied to the importance of two key factors of database recovery, as follows.

- The length of time required to restore database access following an interruption
- The amount of data that will be lost because of an interruption

Before choosing a synchronization mode, you need to consider the impact of losing data if the primary database becomes unavailable. The more closely synchronized the databases, the smaller the transaction loss will be in a disaster recovery takeover situation.

If most of your transactions can be easily reproduced, having closely synchronized databases might not be so important to you.

The workload and performance of the database, the host machines, and the network are tightly integrated. A heavy workload in any one component can impact the performance of any of the other components. The more closely synchronized the databases, the more sensitive your environment becomes to heavy workloads in any one component.

Disaster Recovery in Journal Block Write Mode

If you have a secondary database that is synchronized with its primary database when the primary becomes unavailable, you are obviously in a very good position to recover the database quickly and with a minimal loss of data.

As all audit data written to the primary database is sent to the secondary as it is written, in a best-case disaster recovery scenario, zero-transaction loss can be achieved.

However, as a transaction commit on the primary database does not wait for acknowledgements that audit data is stable on disk on the secondary database, this asynchronous synchronization mode does not guarantee zero-transaction loss.

Disaster Recovery in Journal Switch Mode

You can still be in a reasonably good position to recover quickly and with a minimal loss of data if you operate SDS with a delayed level of synchronization in journal switch mode.

Because you have a database already set up to take over the operations of the primary database, you can apply outstanding journals as quickly as possible and be back online in a minimal and predictable length of time.

Restart Recovery

On SDS secondaries, the restart recovery mechanism ensures that integrity is re-established after restart, by applying journal updates beginning at the farthest-back Log Sequence Number (LSN), where the oldest incomplete transaction starts, and it continues through journals until all available changes have been applied (this ensures that the files are synchronized with the journals). The recovery then performs a "revert" pass, which undoes the effects of all transactions back to farthest back LSN, which ensures that the database is in a consistent state before resuming replay.

Native secondaries interrogate the **DisableNativeRedintegrateOnRestart** parameter and RPS secondaries interrogate the **DisableRPSRedintegrateOnRestart** parameter. These parameters both default to **true**. (These parameters can be defined in the [SyncDbService] section of the JADE initialization file.) Secondary restart recovery is disabled by default for native secondaries, and enabled by default for RPS secondaries.

To enable secondary restart recovery for native secondaries, set the value of the **DisableNativeRedintegrateOnRestart** parameter to **false**; conversely, to enable secondary restart recovery for RPS secondaries, set the value of the **DisableRPSRedintegrateOnRestart** parameter to **false**.

JADE implements an SDS secondary restart recovery audit. These journals, which have the .**rlog** extension, are numbered created, written, and removed the same as the normal .**log** journals. When a block in a file is made free, information from the block header is audited to the .**rlog** journal so that the "revert" pass can correctly recreate block state when reversing that free operation, going backwards undoing the effects of the transactions.

The .**rlog** and .**log** journals are structurally identical, and can be selected for dumping using the JADE Database utility (**jdbutil**). The .**rlog** journals have no operational requirements; they are created and removed as necessary by secondary replay and restart recovery.

Delayed Replay

While a primary database is open and active with updating transactions in progress, journals continue to be generated and transferred to connected secondary servers. You can delay the application of transactions at one or more secondary databases by disabling tracking.

This can be very useful in scenarios such as application deployment, especially one involving a database reorganization that cannot be guaranteed to succeed. In scenarios involving such upgrades, you should disable tracking on at least one secondary database. When invoked, the schema load and reorganization processes on the primary appends redo information to journals, which continue to be transferred to attached secondary servers even when tracking is disabled.

Once the application deployment has completed successfully on the primary system, you can enable tracking again, allowing the secondary to replay the schema load and reorganization processes to regain synchronization with the primary database. If an application load should fail for any reason, the secondary database with tracking disabled remains synchronized with the primary at the point prior to the upgrade.

Deferring SDS Transactions

The [SyncDbService] section of the JADE initialization file on secondary database servers can contain the **MaxDeferredTransactions** parameter, which enables you to configure the threshold of transactions that can remain deferred as a result of lock conflicts.

Lock conflicts arise when reader processes acquire share locks on objects that prevent replay processes from completing transactions.

The default value is **300**, the minimum value is **10**, and the maximum value is **5,000**.

When the number of deferred transactions exceeds the configured limit, replay is paused until the backlog of deferred transactions has reduced to zero (**0**).

With replay paused, journal mirroring can continue; however, when the secondary is not replaying in the current write journal, it is reported as *catching up* and it will remain in a catching up state until the issue is resolved.

When the limit is reached, the following is an example of the diagnostics that are output to the **jommsg.log** file and the SDS work log.

```
2008/06/21 11:24:25.860 01740-11b8 SDS: Queued transactions=501 exceeds max
deferred transactions=500
2008/06/21 11:24:26.738 01740-11b8 SDS: Waiting for 501 delayed transaction(s) to
complete
2008/06/21 11:24:57.012 01740-11b8 SDS: Waiting for 324 delayed transaction(s) to
complete
2008/06/21 11:25:27.693 01740-11b8 SDS: Waiting for 235 delayed transaction(s) to
complete
2008/06/21 11:25:58.353 01740-11b8 SDS: Waiting for 202 delayed transaction(s) to
complete
```

Version Information

The connection establishment protocol exchanges version information between nodes to ensure compatible communication protocols versions, SDS and database, audit structure, and protocol versions.

Not all participant nodes need to be running identical code file versions.

Security and Authentication

As you are likely to want to employ the public Internet Protocol (IP) network infrastructure as an economic means to network geographically separated sites, SDS enables you to institute the security and authentication measures.

The **AllowedHost** and **RestrictedHostAccess** parameters in the [ConnectionParams] section of the JADE initialization file enables you to specify by name or IP address a list of hosts from which the node is willing to accept in-bound SDS connections.

When host restrictions are enforced, the primary database server refuses connections from any non-listed hosts. For details, see "Connection Parameters Section [ConnectionParams]", in the *JADE Initialization File Reference*.

SDS Takeover Operations

A secondary database can take over the role of primary database at any time; for example, for scheduled maintenance where you want to move processing to a new location. An SDS *takeover* is a process that enables a specified secondary database to assume the role of the primary database.

For details, see the following subsections.

- Negotiated Takeovers
- Hostile Takeovers
- Considerations before Performing a Takeover
- Preparing for a Takeover Operation
- Performing a Negotiated Takeover
- Performing a Hostile Takeover

Negotiated Takeovers

When a takeover operation is initiated from the primary database, a negotiated takeover of the specified secondary database is performed. In a negotiated takeover, the primary database first relinquishes its role as the primary database and then requests the specified secondary database to assume the primary database role.

Before a primary database relinquishes its role, it must achieve a database quiet point (a point in time when there are no active transactions). If a quiet point is not achieved within the configured maximum interval to wait for a quiet point interval (in the **MaxWaitForQuietPoint** parameter of the [PersistentDb] section of the JADE initialization file), the takeover operation fails.

When the primary database is ready to relinquish its role, it audits a ‘change to primary’ record in the current journal, closes the journal, and ships this to the specified secondary server. It is then up to the secondary database to assume the role of a primary database.

When the secondary database has replayed all transactions up to and including the ‘change primary’ audit record, the secondary database is then synchronized with the primary database at the takeover quiet point. If there are no reader processes running on the secondary database at this point, it can assume the primary role immediately.

However, if reader processes are executing, the readers may hold locks that are causing the effects of one or more replayed transactions to be isolated (that is, the effects are hidden from all readers).

When initiating a negotiated takeover, you must decide between a conditional and a forced takeover. Base your decision on what is more important at the time: allowing long-running processes to complete, or the takeover because maintenance on the primary host cannot wait.

- In a *conditional* takeover, the reader processes are given precedence.

  The primary database does not relinquish its role until it has received notification to do so by the secondary database. If reader or replay conflicts exist on the secondary database, the takeover operation is abandoned.

- In a *forced* takeover, the takeover operation is given precedence. The primary database relinquishes its primary role, assuming a secondary database role after it has shipped the journal containing the 'change to primary' audit record to the specified secondary database.

  When the secondary database processes the 'change to primary' record with the **force** parameter, it attempts to force through the visibility of currently isolated transactions if conflicts with reader processes exist. This is achieved by terminating user processes with locks that conflict with isolated transactions.

  In this mode, transactions are forced through one at a time until none remain. When all isolated transactions have been committed, the secondary database assumes the primary database role.

  Notes Processes running on a secondary server node can be forcibly terminated when a forced takeover takes place if they have objects locked that are blocking replaying transactions.

  Server applications specified in the **ServerApplication** parameter in the [JadeAppServer] section and the **ServerApplication** parameter in the [JadeServer] section of the JADE initialization file that are terminated in a forced takeover operation are not automatically restarted.

Hostile Takeovers

When a takeover is initiated from a secondary database, a hostile takeover is performed, where the secondary database assumes a primary role without involving the original primary database.

Caution You should consider a hostile takeover only in a disaster recovery situation in which the primary site is lost and it is no longer viable to restore and recover the primary database.

Considerations before Performing a Takeover

You could consider performing a takeover when the primary database will be unavailable for an extended time. (See also "Negotiated Takeovers", earlier in this chapter.) In some instances, it might be less disruptive to wait for the original primary system to become available again, rather than to perform a takeover.

You should base the decision to perform a takeover operation on the following considerations.

- If the primary host or the network has failed, the time required to restore them or to revert to backup hardware at the primary site
- If the primary database has been damaged, the time required to restore the database from backup using standard restore and recovery techniques
- If the journals on the secondary are behind, the time required to synchronize the database
- If the journals cannot be synchronized due to a primary host or network failure, the impact of losing transactions or having to reprocess transactions

Preparing for a Takeover Operation

Before performing a takeover operation, you should perform the following actions.

1. At a minimum, shut down all update-capable processes on the primary database (when possible, in the negotiated case).

   Stopping update processing ensures that there are no active transactions when the takeover is initiated, as a database quiet point on the primary database is a prerequisite to the takeover action. If updating processes and users are left operating on the primary database, they will encounter exceptions if they attempt to update non-environmental objects when the primary database has converted to a secondary database.

2. If the reason for the takeover is to perform scheduled maintenance on the primary host, it is preferable to shut down all nodes connected to the primary host before initiating the takeover.

You do not have to stop inquiry processes on the secondary database that is to assume the primary role. When the takeover is accomplished and the secondary database has changed roles to a primary database, the inquiry processes can begin to perform updating transactions without having to restart that node.

Performing a Negotiated Takeover

Initiate a negotiated takeover from a primary database by performing one of the following actions.

- Using the **Initiate Takeover** command from the JADE SDS Admin utility connected to the primary database server. For details, see "Taking Over the Primary Database Role" under "Using the SDS Administration Application", later in this chapter.
- Invoking the **JadeDatabaseAdmin**::**sdsInitiateTakeover** method from a user-written administration application running on the primary database server.

Performing a Hostile Takeover

A hostile takeover is initiated from a secondary by performing one of the following actions.

- Using the **Initiate Hostile Takeover** command from the JADE SDS Admin utility connected to the secondary database server. For details, see "Taking Over of the Primary Database Role from a Secondary Database" under "Using the SDS Administration Application", later in this chapter.
- Invoking the **JadeDatabaseAdmin**::**sdsInitiateHostileTakeover** method from a user-written administration application running on the secondary database server.

Backing Up and Restoring Secondary Databases

This section contains the following topics.

- Quiesced Database Backup
- Online Database Backup
- Offline Database Backup

For details, see the following subsections.

Quiesced Database Backup

A database quiesced backup is not permitted on a secondary database. If you attempt a quiesced backup on a secondary system, an exception is raised (that is, *3206 - Operation not permitted on a secondary database*).

Online Database Backup

The online backup of secondary databases is supported so that you can periodically take backups to archive storage.

Performing the backup on a secondary database relieves the primary system from this burden.

Offline Database Backup

You can perform an offline backup of a secondary database using the **jdbutil** or **jdbutilb** JADE Database utility when the database is offline; that is, the secondary database server has been shut down.

Note As roll-forward recovery on a secondary database that terminates with outstanding transactions does not undo the incomplete transactions, any active transactions at the end of recovery therefore remain in an incomplete state.

If it is necessary to convert a restored secondary database into a primary database to support update processing, you must perform a hostile takeover operation. For details, see "Performing a Hostile Takeover" under "SDS Takeover Operations", earlier in this chapter.

Creating a Secondary Database from an Online Backup of a Primary Database

An online backup of a primary database that is restored and recovered while it still has a primary database role results in a journal timestamp mismatch when the database is started in a secondary database role following the backup recovery.

To create a secondary database from an online backup of a primary database:

1. Restore the backup on the secondary host.
2. Restore the journals that would be required to recover the online backup to the secondary **current** journals subdirectory (only the first journal is actually required).
3. Start the secondary database server.

The secondary database initialization logic handles starting up from an online backup and takes the necessary steps to ensure that backup recovery is conditioned correctly for a secondary database role. (Secondary database backup recovery differs from the backup recovery for a primary database.)

Reorganizing an SDS Database

To reorganize an SDS database, run the reorganization as:

- **Replayable**, and SDS tracking replays the reorganization process on the secondary database.
- **Non-Replayable**, and re-clone the secondary database from the primary after the reorganization. (This option may be preferable for lengthy reorganizations.)

For more details about:

- Reorganizations in SDS, see "SDS Reorganization Considerations", in Chapter 14 of the *JADE Developer's Reference*.
- Reorganizing a primary database and replaying the reorganization on the secondary when the journals are received to have the effects of the reorganization applied, see "Replayable Reorganizations", in Chapter 14 of the *JADE Developer's Reference*.
- Restarting any server applications that have been stopped and enabling secondary database users who were signed off from the primary database during reorganization to sign on (if this was previously disabled) following reorganization or schema transition, see "Stopping and Starting Server Applications in Secondary Systems", in Chapter 10 of the *JADE Developer's Reference*.

Configuring the Synchronized Database Service

To support SDS, the JADE initialization file contains the [ConnectionParams] and [SyncDbService] sections, summarized in the following subsections. For details, see the *JADE Initialization File Reference*. For details about developing applications using SDS, see Chapter 10, "Synchronized Database Service (SDS) Development Considerations", in the *JADE Developer's Reference*.

[ConnectionParams] Section

When using the Synchronized Database Service (SDS), each communication end-point in a primary to secondary network pairing requires connection parameters to be specified.

To specify connection parameters for a specific server, the value of the **MyName** parameter in the [SyncDbService] section is appended to the [ConnectionParams] section name. For example, the connection parameters for an SDS server with a **MyName** parameter value of **WilBur** are specified in the [ConnectionParams.WilBur] section.

The [ConnectionParams.<*name*>] section can contain the parameters summarized in the following table.

Parameter	Default	Specifies…
AllowedHost<*n*>	Not specified	The enumerated list of all host names or IP addresses that can connect to the primary database server when the **RestrictedHostAccess** parameter is set to **true**
NetworkSpecification<*n*>		Connection parameters for the primary database server node to listen for incoming connections from secondary database server nodes
RestrictedHostAccess	False	Whether the primary database server refuses connections from any host that is not included in the enumerated list specified in the **AllowedHost** parameters
ServerNodeSpecifications		The network connection parameters used by a secondary database server node to connect to this primary database server node with the specified name across a network
SocketBufferSize	128K bytes	The TCP/IP buffer size, to match the link characteristics

[SyncDbService] Section

The [SyncDbService] section, which enables you to configure SDS, can contain the parameters summarized in the following table.

Parameter	Default	Description
AuditCauseEvents	False	Specifies whether non-immediate user events caused on persistent objects are audited in the transaction journal for replay on secondary databases.

Parameter	Default	Description
ConnectionPollInterval	30	Defines in seconds the interval at which the secondary database polls the primary database to determine reachability via the communication paths. (Configured on secondary databases.)
DatabaseRole	Not specified	Activates SDS on a database server node and determines the starting database role for the server.
DatabaseSubrole	Not specified	Activates the SDS secondary as an RPS node.
DisableNativeRedintegrateOnRestart	true	Disables secondary restart recovery for native secondaries.
DisableRPSRedintegrateOnRestart	true	Disables secondary restart recovery for RPS secondaries.
JournalReadBuffers	4	Specifies the number of buffers to use when reading a journal file on disk.
JournalReplayBlocksize	128K bytes	Specifies the size in bytes of each read buffer used when replaying a journal file.
JournalXferBlocksize	128K bytes	Specifies the size in bytes used for journal file disk and network I/O operations.
JournalXferCompression	True	When enabled, causes SDS to compress journal data blocks for transmission across the network.
MaxDeferredTransactions	300	Pauses transaction replay until the backlog of deferred transactions has reduced to zero (**0**).
MaxSecondaries	16	Specifies the maximum number of secondary databases that can attach to a primary database server.
MyName	Not specified	Specifies a unique name for a JADE system operating within a Synchronized Database Service environment (SDE).
PrimaryServerName	Not specified	Specifies the name on the secondary database server of the JADE system with the primary role.
ReadAccessDisabled	False	Specifies whether read access to the database is disabled.
ReconnectInterval	300 seconds	Specifies the frequency at which a secondary database server attempts to reconnect to its primary server when a connection fails or the secondary database loses its connection to the primary database.

Parameter	Default	Description
SyncMode	JournalBlockWrite	Specifies the mode in which journals are transferred to secondary databases from the primary database. The default **JournalBlockWrite** value transfers blocks as they are generated. Set the value to **JournalSwitch** if you want to transfer journal data when the journal is complete.
TrackingDisabled	False	Specifies whether the database tracking (journal replay) process is disabled.

For details, see "Synchronized Database Service Section [SyncDbService]", in the *JADE Initialization File Reference*.

JADE Initialization File Configuration Examples

This section contains examples of JADE initialization files sections and parameters used to configure an SDE. (See also "Creating your First SDS Environment", later in this chapter.)

Example 1

The following fragment of a JADE initialization file shows the SDS parameters appropriate for a JADE system in a primary database role at Site A.

```
## Site A primary Role parameters ##
## Synchronized Database Service Parameters ##

[SyncDbService]
DatabaseRole=PrimaryRole
MyName=SiteA_Payroll
MaxSecondaries=32
AuditCauseEvents=true

[ConnectionParams.SiteA_Payroll]
# SiteA_Payroll's connection parameters used
# when it's the primary database
NetworkSpecification1=TcpIp,Enabled,61001
# Set RestrictedHostAccess if you want to restrict
# the secondary hosts that can connect to this primary
RestrictedHostAccess=false

[ConnectionParams.SiteB_Payroll]
# connection params used by SiteA_Payroll to connect
# to SiteB_Payroll after a role change
ServerNodeSpecifications=TcpIp,porpoise.siteb.ja.net,62001
```

The following fragment of a JADE initialization file shows the SDS parameters appropriate for a JADE system in a secondary database role at Site B.

```
## Synchronized Database Service Parameters ##
[SyncDbService]
DatabaseRole=SecondaryRole
MyName=SiteB_Payroll
```

```
# Our primary server is SiteA_Payroll
PrimaryServerName=SiteA_Payroll
# attempt to reconnect to primary every 5 minutes
# when disconnected
ReconnectInterval=300
# poll primary database every 30 seconds
ConnectionPollInterval=30
# start off with tracking enabled
TrackingDisabled=false
# start off with read-access enabled
ReadAccessDisabled=false
AuditCauseEvents=true

[ConnectionParams.SiteA_Payroll]
# connection parameters used by SiteB_Payroll
# to connect to # SiteA_Payroll when it's the primary
ServerNodeSpecifications=TcpIp,orca.sitea.ja.net,61001

[ConnectionParams.SiteB_Payroll]
# connection parameters used by SiteB_Payroll
# to listen for and accept connections when
# SiteB_Payroll becomes the primary system
NetworkSpecification1=TcpIp,Enabled,62001
RestrictedHostAccess=false
```

Example 2

This example of a fragment of a JADE initialization file uses section qualifier names (described in "Two-Level Section Names" under "Format of the JADE Initialization File", in the *JADE Initialization File Reference*) so that the JADE initialization file is shared, allowing the parameters for a primary database and two secondary database systems to be specified in a single initialization file. The **name** command line parameter is used to select the sections relevant to each site.

```
## Common Synchronized Database Service Parameters ##
[SyncDbService]
ConnectionPollInterval=30
ReconnectInterval=300
TrackingDisabled=false
ReadAccessDisabled=false
MaxSecondaries=32
AuditCauseEvents=true

[SiteA.SyncDbService]
DatabaseRole=PrimaryRole
MyName=SiteA_Payroll

[SiteB.SyncDbService]
# remote database role server
DatabaseRole=SecondaryRole
MaxDeferredTransactions=1000
MyName=SiteB_Payroll
PrimaryServerName=SiteA_Payroll
ReconnectInterval=60

[SiteC.SyncDbService]
```

```
# local backup server
DatabaseRole=SecondaryRole
MyName=SiteC_Payroll
PrimaryServerName=SiteA_Payroll
MaxDeferredTransactions=100
ConnectionPollInterval=5

[ConnectionParams.SiteA_Payroll]
# SiteA_Payroll's connection parameters used
# when it's the primary database
NetworkSpecification1=TcpIp,Enabled,61001
# connection parameters used by secondary servers
# to connect to SiteA_Payroll when it's the primary
ServerNodeSpecifications=TcpIp,orca.sitea.ja.net,61001
# enable if you want to specify a restricted host list
RestrictedHostAccess=true
AllowedHost1=porpoise.siteb.ja.net
AllowedHost2=beluga.siteb.ja.net
AllowedHost3=dolphin.sitec.ja.net

[ConnectionParams.SiteB_Payroll]
# SiteB_Payroll's connection parameters used
# when it's the primary database
NetworkSpecification1=TcpIp,Enabled,62001
# connection parameters used by secondary servers
# to connect to SiteB_Payroll when it's the primary
ServerNodeSpecifications=TcpIp,porpoise.siteb.ja.net,62001
# enable if you want to specify a restricted host list
RestrictedHostAccess=true
AllowedHost1=hector.sitea.ja.net
AllowedHost2=orca.sitea.ja.net
AllowedHost3=dolphin.sitec.ja.net

[ConnectionParams.SiteC_Payroll]
# SiteC_Payroll's connection parameters used
# when it's the primary database
NetworkSpecification1=TcpIp,Enabled,63001
# connection parameters used by secondary servers
# to connect to SiteB_Payroll when it's the primary
ServerNodeSpecifications=TcpIp,dolphin.sitec.ja.net,63001
RestrictedHostAccess=true
AllowedHost1=hector.sitea.ja.net
AllowedHost2=orca.sitea.ja.net
AllowedHost3=porpoise.siteb.ja.net
AllowedHost4=beluga.siteb.ja.net

## Client/Server network parameters ##
[SiteA.JadeClient]
ServerNodeSpecifications=TcpIp,orca,60001

[SiteA.JadeServer]
NodeName=SiteA DbServer
NetworkSpecification1=TcpIp,Enabled,60001

[SiteB.JadeClient]
```

```
ServerNodeSpecifications=TcpIp,porpoise,60002

[SiteB.JadeServer]
NodeName=SiteB DbServer
NetworkSpecification1=TcpIp,Enabled,60002

[SiteC.JadeClient]
ServerNodeSpecifications=TcpIp,dolphin,60003

[SiteC.JadeServer]
NodeName=SiteC DbServer
NetworkSpecification1=TcpIp,Enabled,60003
```

Creating your First SDS Environment

This section provides instructions for creating your first SDS environment; for example, when establishing a test environment.

Recommendations

When creating an SDS environment, you should use two-level JADE initialization file section names to store both the primary and secondary information in a single initialization file so that it can be simply copied to the secondary system during cloning.

For details, see "Two-Level Section Names" and "Sharing JADE Initialization Files", in the *JADE Initialization File Reference*.

Assumptions

When creating an SDS environment, the following is assumed.

- A TCP/IP connection can be established from the secondary environment to the primary.
- The secondary environment has sufficient processing resource and disk space (for the database, binaries, and journals).

Creating an SDS Environment

Use a database backup of a primary system and then restore it (with no recovery) to create the secondary system, so that the integrity of the files is checked during the restore operation.

Some of the issues that you must consider in a production environment include the following.

- Selection of the appropriate journal synchronization mode, by using the **SyncMode** parameter in the [SyncDbService] section of the JADE initialization file. For details, see "Selecting the Journal Synchronization Mode for a Secondary Database" under "Synchronizing with a Primary", earlier in this chapter.
- Calculation of the bandwidth required for timely journal transfers.
- If using the journal switch synchronization mode, selection of appropriate parameter values for journal switches (for example, the [PersistentDb] section of the JADE initialization file **JournalCloseAction**, **JournalMaxSize**, **JournalMinSize**, and **JournalSwitchInterval** parameters).
- Backup of the secondary environment (including journals).

- Operational steps required to complete the transition of the secondary system to a fully functional primary system to which users can connect and use.

Using the SDS Administration Application

The SDS Administration application enables you to monitor and administer your primary and secondary databases. The following is an example of a command line required to run the SDS Administration application.

```
C:\Jade\bin\jade.exe name=PrimarySite schema=JadeMonitorSchema app=JadeSDSAdmin
path=c:\jade\system ini=c:\jade\system\jadeSDS.ini server=multiUser
```

» To access the SDS Administration utility

1. Ensure that the JADE Remote Node Access utility is running. For details about the JADE Remote Node Access utility, see "Using the JADE Remote Node Access Utility", in the *JADE Remote Node Access Utility User's Guide*.
2. Select the **SDS Administration** program icon from your JADE program folder, to run the SDS Administration application.

When you successfully invoke the **JadeSDSAdmin** application, the SDS Admin database view is then displayed, enabling you to monitor and administer systems in a Synchronized Database Environment (SDE).

A collapsed node in a database view displays a summary of the details in the Attributes column for that database. Expanding a node displays all attributes details. The database view contains a node for the primary and each secondary system, with attributes of an entity of a collapsed or expanded primary or secondary system node listed in the **Attributes** column at the right of the form.

An expanded primary system node on the database view provides a summary of primary database attributes, whose details you can display by expanding the primary system and summary nodes. An expanded secondary system node provides a summary of transactions, journal state, and operational state attributes, whose attribute details you can display by expanding the operational state, journal state, and transactions nodes, respectively.

Note The context menu, accessed when you right-click on a node or element, contains commands applicable to the entity from which you access the context menu. For example, you can close the current journal when you access the context menu from the primary system node or you can start or stop tracking when you access the context menu from a secondary system node.

The state of a secondary database is indicated by the background color of secondary system nodes in the primary database and secondary database forms, as follows.

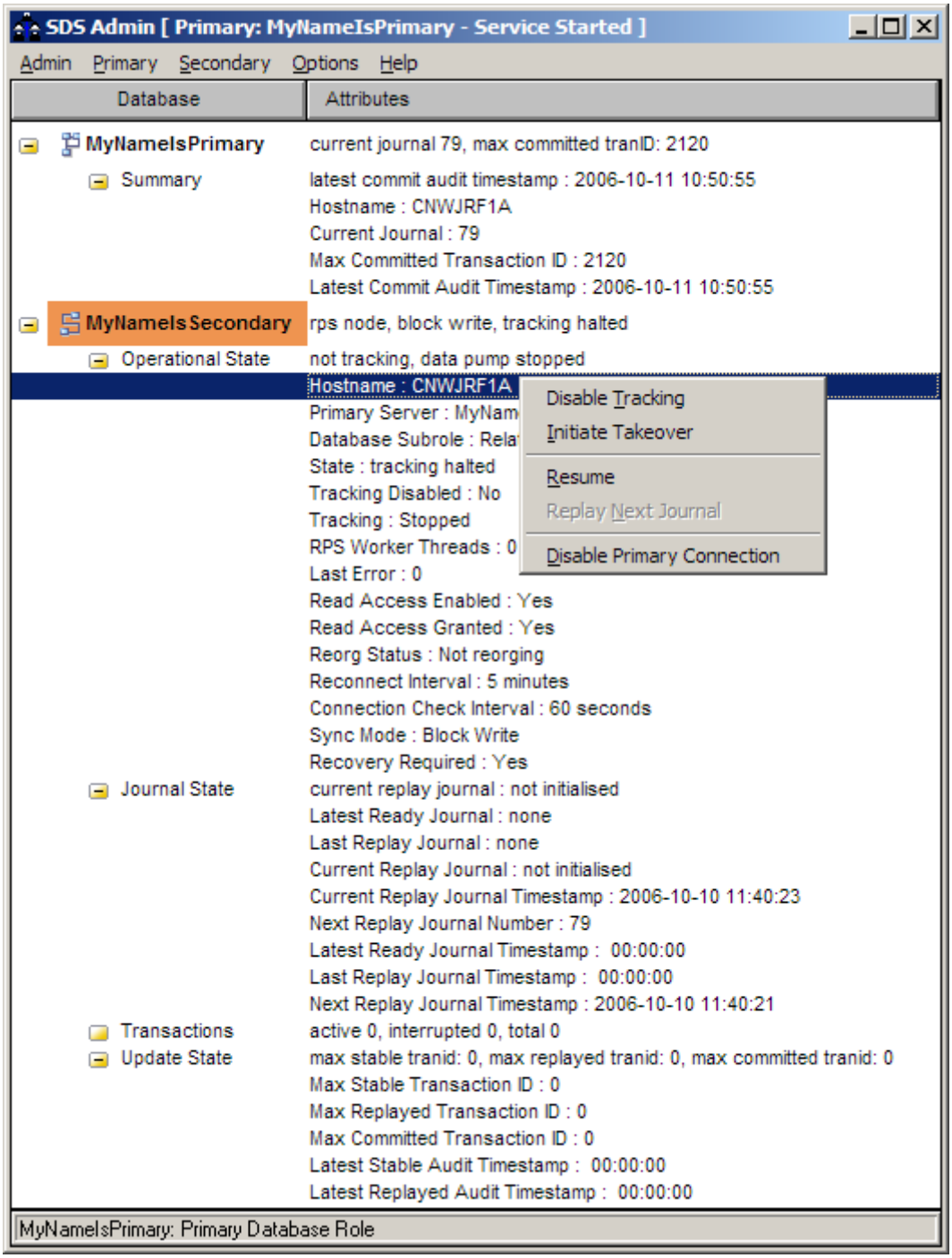
- Bright green, which indicates that the secondary database is synchronized with the primary database.
- Yellow, which indicates that the secondary database is catching up with the primary database.
- Orange, which indicates that database tracking has halted. If tracking has halted because of an error condition, the **Last Error** attribute contains the associated error code.
- Blue, which indicates that the secondary database is replaying a reorganization.
- Dark gray, which indicates that the secondary database is disconnected from the primary database.
- Red, which indicates a problem on a secondary server caused the halting of an operation (for example, when a journal is missing or when out of disk space). Alarm events require operation intervention to resolve the problem before resuming the operation that halted.

The value of the **Last Error** attribute in the operational state details provides the associated JADE error code (for example, **3125** if a required transaction journal was not found).

You can display the legend for the database state at the bottom of the primary and secondary database forms and change the default state colors. For details, see "Displaying the Legend for the Database State", later in this chapter.

Administering and Monitoring a Primary Database

The SDS Admin primary database view enables you to monitor the primary database and one or more secondary databases attached to the primary database. The following diagram shows an example of the primary database view in which the **Primary** and **Secondary** database nodes and the **Journal State** and **Operational State** entity nodes in the secondary database have been expanded.

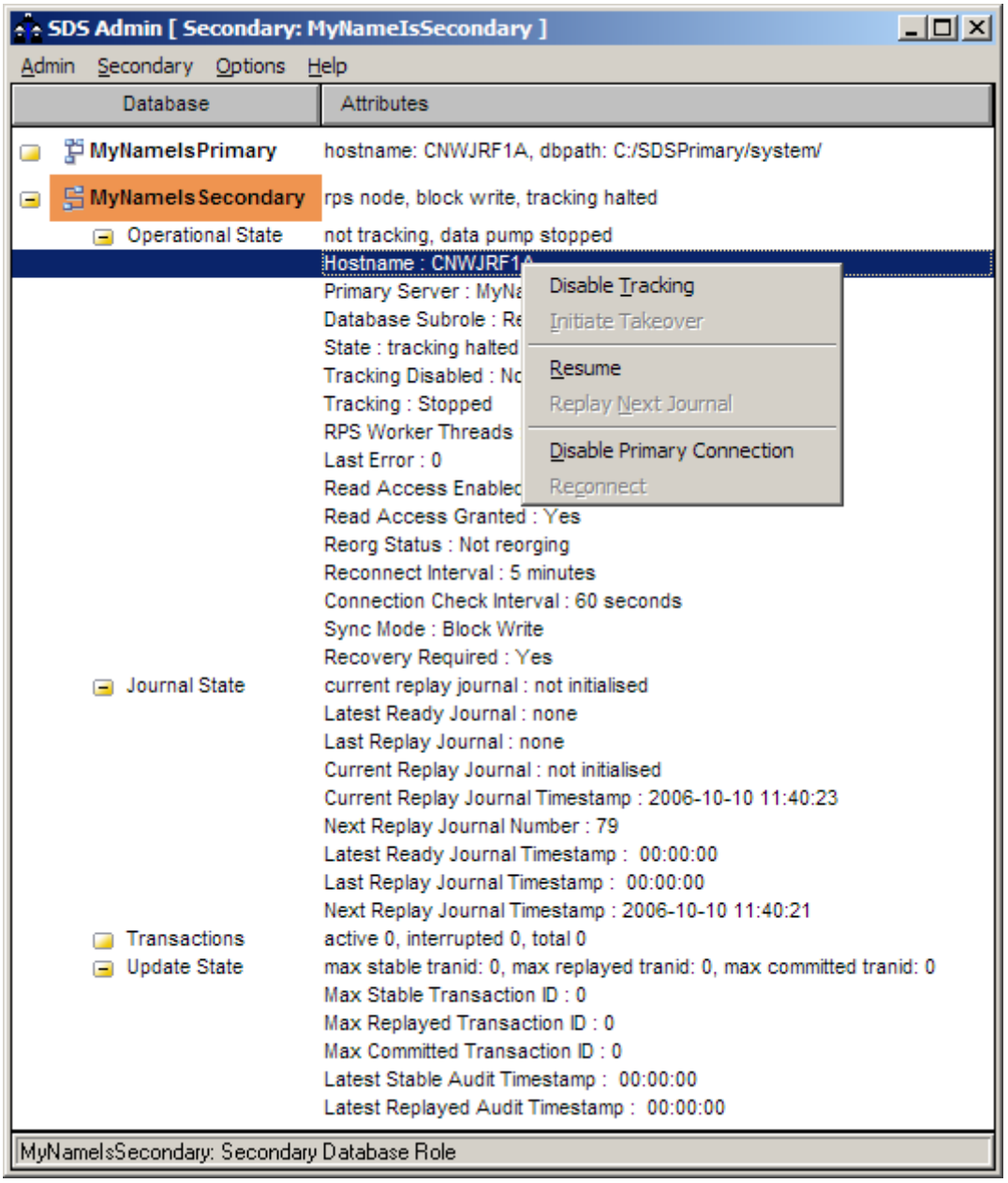


The attribute details for the primary database, displayed in the **Attributes** column at the right of the form when you have expanded the primary database node and the lower-level summary node, are as follows.

- Primary database name
- Host name
- Current journal number
- Maximum committed transaction identifier
- Timestamp of the latest committed audit

Administering and Monitoring a Secondary Database

The SDS Admin secondary database view enables you to monitor a secondary database. The following diagram shows an example of the secondary database view in which the **Secondary** database node and the **Journal State** and **Operational State** entity nodes have been expanded.



When attached to a secondary database server, the database view displays a single secondary database entry only.

You cannot start node sampling remotely on a primary server from an SDS secondary server.

For details about using the SDS Administration application, see the following section.

The attribute details for each secondary database, displayed in the **Attributes** column at the right of the database view when you have expanded a secondary database node and the lower-level nodes, are as follows.

- Secondary database name, specified in the **MyName** parameter in the [SyncDbService] section of the JADE initialization file
- Operational state
  - Host name
  - Primary database name, specified in the **PrimaryServerName** parameter in the [SyncDbService] section of the JADE initialization file
  - Database subrole (**Native** or **Relational**)
  - State (for example, **catching up** or **disconnected**)
  - Tracking status
  - Number of RPS worker threads
  - Last error number
  - Read access enabled (**Yes** or **No**)
  - Read access granted (**Yes** or **No**)
  - Reorganization status (for example, **Not reorging**, **Seeking Approval**, or **Offline phase**)
  - Reconnection interval (in minutes or in seconds)
  - Connection check interval (in seconds)
  - Synchronization mode, specified by the **JournalBlockWrite** (default) or **JournalSwitch** value of the **SyncMode** parameter in the [SyncDbService] section of the JADE initialization file
  - Recovery required (**Yes** or **No**)

    Note When a secondary server is restarted in an interrupted mode, the recovery required attribute is set. This attribute is reset when the status of the outstanding interrupted transactions have been resolved.
- Journal State
  - Latest ready journal number
  - Last replay journal number
  - Current replay journal number
  - Current replay journal timestamp
  - Next replay journal number
  - Latest ready journal timestamp
  - Last replay journal timestamp
  - Next replay journal timestamp

- Transactions, summarized in the collapsed node as active, interrupted, and total transactions
  - Transaction id
  - User name
  - Transaction start time
  - Status
- Update State (block write synchronization mode only)
  - Transaction id of the maximum stable audit record
  - Transaction id of the maximum replayed audit record
  - Transaction id of the maximum committed audit record
  - Timestamp of the latest stable replayed audit record
  - Timestamp of the latest replayed audit record

Using SDS Administration Application Menus

The menu bar on the SDS Admin database view contains the menus listed in the following table.

Menu	Enables you to…
Admin	Administer the primary or a secondary database
Primary	Stop and start the Synchronized Database Service and close the current journal
Secondary	Perform secondary database operations
Options	Specify options associated with the database view
Help	Access the standard Common User Access (CUA) help options

For details, see the following subsections.

Note The Primary menu is displayed only when the SDS Administration application is attached to a primary database.

Using the Admin Menu

Use the Admin menu commands to perform one of the actions listed in the following table.

Command	For details, see…
Initiate Takeover / Initiate Hostile Takeover	Taking Over the Primary Database Role
Make Primary	Making a Database with an Undefined Role the Primary Database
Refresh	Refreshing the SDS Admin Database View
Exit	Closing the SDS Administration Application

Actions that do not apply to the current database role are disabled (for example, the **Initiate Hostile Takeover** command is displayed only when the SDS Administration application is attached to a secondary database).

The **Make Primary** command is displayed in the Admin menu only when the SDS database is initialized in non-SDS mode; that is, the database role is undefined. For details, see "Making a Database with an Undefined Role the Primary Database", earlier in this chapter.

Taking Over the Primary Database Role

A secondary database can take over the role of primary database at any time; for example, for scheduled maintenance where you want to move processing to a new location.

A takeover initiated from a secondary database is considered a *hostile* takeover. For more details about taking over the primary database role and about negotiated and hostile takeovers, see "SDS Takeover Operations", earlier in this chapter.

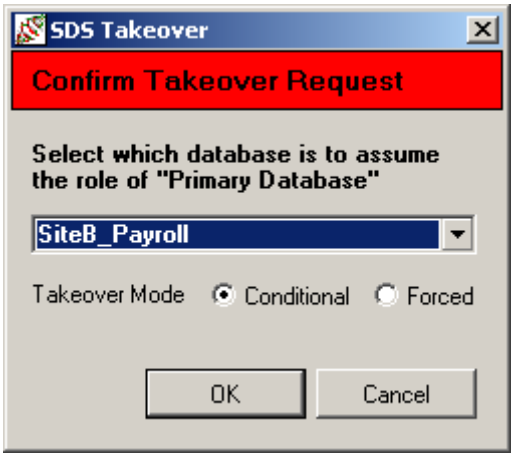
To take over the primary database role

1. Select the **Initiate Takeover** command in the Admin menu.

   Note The **Initiate Takeover** or **Initiate Hostile Takeover** command is enabled only when the secondary system is disconnected from the primary system.

   When accessed from a secondary database, this is displayed as the **Initiate Hostile Takeover** command. For more details, see "Taking Over of the Primary Database Role from a Secondary Database", later in this chapter.

   If you initiated this action from the primary database, the SDS Takeover dialog, shown in the following diagram, is then displayed.



2. In the combo box, select the secondary database that is to assume the primary database role.
3. If you want the takeover operation to have precedence over reader processes so that the takeover is forced if conflicts exist, select the **Forced** option button.

   By default, conditional takeovers are performed; that is, the takeover is performed only when there are no reader process conflicts. If reader or replay conflicts exist on the secondary database, the takeover operation is abandoned.
4. Click the **OK** button. Alternatively, click the **Cancel** button to abandon the takeover action.

The SDS Admin database view is then updated to reflect the role change when the next refresh action occurs.

Making a Database with an Undefined Role the Primary Database

If you initiate the **JadeSDSAdmin** application and the SDS mode is undefined (that is, the value of the **DatabaseRole** parameter in the [SyncDbService] section in the JADE initialization file is not specified), the SDS Admin primary database view is displayed but with no database node and with only the Admin, Options, and Help menus displayed in the menu bar.

An exception is raised if the **MyName** parameter in the [SyncDbService] section of the JADE initialization file is not a valid name.

To make the current node the primary database server

- Select the **Make Primary** command in the Admin menu.

Note The **Make Primary** command is displayed in the Admin menu only when the SDS database is initialized in non-SDS mode; that is, the database role is undefined.

The node is then made the primary, the Primary and Secondary menus are displayed in the menu bar, and the collapsed primary node is displayed in the SDS Admin primary database view.

Refreshing the SDS Admin Database View

You can refresh database details displayed on the database view pane at any time.

To refresh the primary or secondary system details, perform one of the following actions

- Press F5
- Select the **Refresh** command in the Admin menu.

All information currently displayed on the SDS Admin database view is then updated. (See also "Automatically Refreshing the SDS Admin Database View", later in this chapter.)

Closing the SDS Administration Application

To close the SDS Admin application

- Select the **Exit** command in the Admin menu.

The SDS Admin application is then closed. When the **Save Settings on Exit** option is set (for details, see "Saving Settings when Exiting from the SDS Admin Application", later in this chapter), the application and user options settings are then saved in the JADE initialization file for use when the SDS Administration utility is next started up.

Using the Primary Menu

The Primary menu enables you to stop or start an SDS service and to close the current journal. Use the Primary menu commands to perform one of the actions listed in the following table.

Command	For details, see…
Stop Service / Start Service	Stopping and Starting a Synchronized Database Service
Audit Stop Tracking	Stopping Audit Tracking
Close Current Journal	Closing the Current Journal

Stopping and Starting a Synchronized Database Service

To stop the SDS service on the primary database server

- Select the **Stop Service** command in the Primary menu.

The SDS service is then stopped on the primary database server and the menu command changes to the **Start Service** command.

Note When you stop a service, a subset of secondary attributes is displayed.

To start the SDS service on the primary database server

- Select the **Start Service** command in the Primary menu.

The SDS service is then started on the primary database server and the menu command changes to the **Stop Service** command.

Stopping Audit Tracking

To disable audit tracking

- Select the **Audit Stop Tracking** command in the Primary menu.

A **Stop Tracking** audit record is then written to the current journal. When this record is replayed on an RPS node or on an SDS secondary, tracking halts at that point in the audit trail.

Notes This action neither forces a quiet point nor closes the current journal.

The main purpose for this in an RPS context is to establish a journal trigger that coincides with a point-in-time on the primary database, to enable establishing a snapshot of the target database frozen at that time.

Closing the Current Journal

You can close the current journal so that it is made available for transfer to the secondary databases.

To close the current journal

- Select the **Close Current Journal** command in the Primary menu.

The current journal is then closed, made available for transfer to secondary databases, and the journal details displayed in the Primary Database (for the primary role) and Secondary Databases table (for secondary roles) are updated to reflect the latest, current, and next journal numbers.

Using the Secondary Menu

The Secondary menu enables you to monitor and control a secondary database in an SDS environment.

Use the Secondary menu commands to perform one of the actions listed in the following table.

Command	For details, see…
Enable Read Access / Disable Read Access	Enabling or Disabling Read Access of Persistent Objects in the Database
Enable Tracking / Disable Tracking	Enabling or Disabling Tracking

Command	For details, see…
Initiate Takeover / Initiate Hostile Takeover	Taking Over of the Primary Database Role from a Secondary Database
Resume	Resuming Journal Replay
Replay Next Journal	Replaying the Next Journal
Disable Primary Connection	Disabling a Connection to the Primary Database
Reconnect	Reconnecting a Secondary Database

Enabling or Disabling Read Access of Persistent Objects in the Database

To enable read access to persistent objects in the database

1. In the database view pane, select the secondary database whose read access you want to enable.
2. Select the **Enable Read Access** command in the Secondary menu.

Read access to persistent objects in the database is then enabled, the value of the **Read Access Enabled** attribute in the operational state details changes to **Yes**, and the menu command changes to the **Disable Read Access** command.

Read access is not granted immediately when interrupted transactions remain pending after a restart operation. Read access is granted only when all remaining interrupted transactions complete. (For more details, see "Synchronizing with a Primary", earlier in this chapter.)

To disable read access to the secondary database

1. In the database view pane, select the secondary database whose read access you want to disable.
2. Select the **Disable Read Access** command in the Secondary menu.

The selected secondary database then no longer has read access to persistent objects in the database, the value of the **Read Access Enabled** attribute in the operational state details changes to **true**, and the menu command changes to the **Enable Read Access** command.

When read access is disabled on the secondary database, inquiry applications cannot access persistent objects resident in the database server. Any attempt by a user application to access persistent objects when read access is disabled raises an exception.

Enabling or Disabling Tracking

To enable secondary database tracking of primary database transactions

1. In the database view pane, select the secondary database whose tracking you want to enable.
2. Select the **Enable Tracking** command in the Secondary menu.

Tracking is then enabled on the selected secondary database, the value of the **Tracking is** attribute in the operational state details changes to **enabled**, and the menu command changes to the **Disable Tracking** command.

To disable secondary database tracking of primary database transactions

1. In the database view pane, select the secondary database whose tracking you want to disable.
2. Select the **Disable Tracking** command in the Secondary menu.

Tracking is then disabled on the selected secondary database, the value of the **Tracking is** attribute in the operational state details changes to **disabled**, and the menu command changes to the **Enable Tracking** command.

Taking Over of the Primary Database Role from a Secondary Database

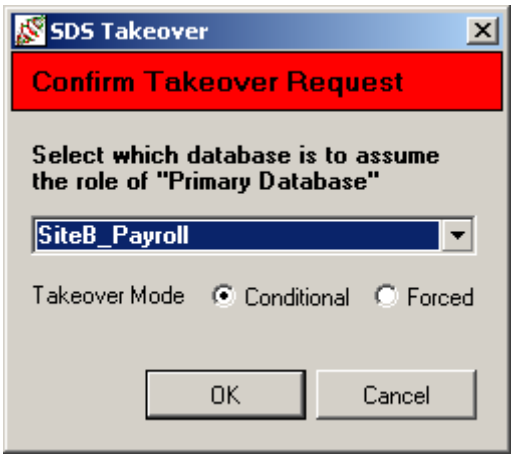
Caution You should consider a hostile takeover only in a disaster recovery scenario when the primary database is no longer available.

For details about taking over the primary database role, see "SDS Takeover Operations", earlier in this chapter.

To take over the primary database role when the operational state is disconnected

1. Perform one of the following actions.
   - Select the secondary database in the database view pane that is to take over the role of primary database and then select the **Initiate Takeover** command in the Admin menu or the Secondary menu.
   - Select the secondary database in the database view pane that is to take over the role of primary database and then select the **Initiate Hostile Takeover** command in the Admin menu.

     Alternatively, when the secondary system is disconnected from the primary system, the **Initiate Takeover** command is enabled.
2. If you initiated this action from the primary database, the SDS Takeover dialog, shown in the following diagram, is then displayed.



3. In the combo box, select the secondary database that is to assume the primary database role.
4. If you want the takeover operation to have precedence over reader processes so that the takeover is forced if conflicts exist, select the **Forced** option button.

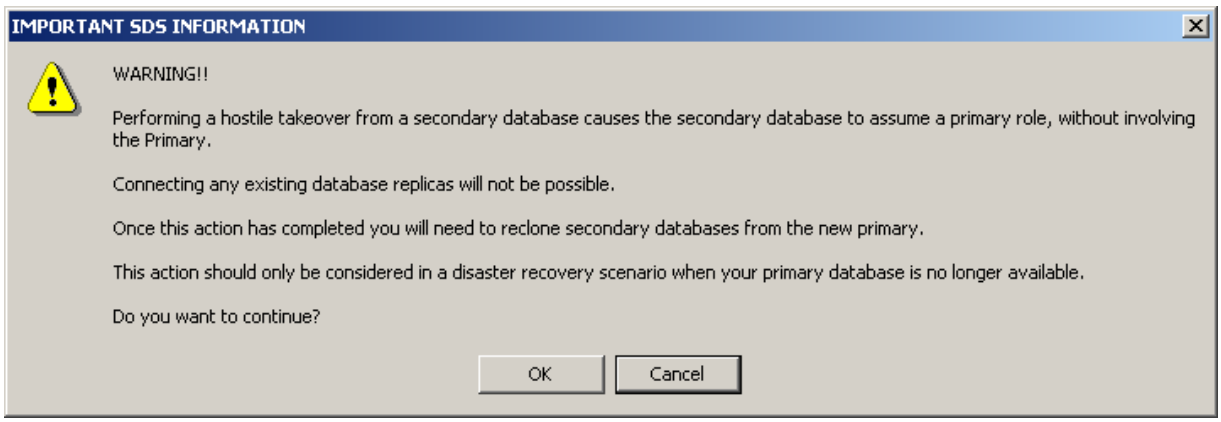
   By default, conditional takeovers are performed; that is, the takeover is performed only when there are no reader process conflicts. If reader or replay conflicts exist on the secondary database, the takeover operation

is abandoned.

5. Click the **OK** button. Alternatively, click the **Cancel** button to abandon the takeover action.

If you initiated this action from a secondary database, click the **OK** button in the SDS warning message box if you are aware of the consequences of initiating the takeover of the primary role from the secondary database and you want to continue.

The SDS warning message box is shown in the following diagram. (Alternatively, click the **Cancel** button to abandon taking over the primary database role.)



The database view is then updated as the role change progresses.

Resuming Journal Replay

You can resume replay of journals on a secondary database when tracking is enabled; for example, when tracking was halted because of missing journals or insufficient disk to create a journal. (For details about disabling tracking, see "Enabling or Disabling Tracking", earlier in this section.)

Note You cannot resume tracking on a secondary database that requires a release upgrade.

When journal replay is resumed, the following actions are performed on the secondary server.

1. Scans its current journal directory for new arrivals and updates the latest ready journal information.
2. Starts replaying from the next replay journal if it is resident on the secondary server.
3. Requests the next required journal from the primary database server, if connected.

If you want to initiate the replaying of the next ready journal on the secondary database server when tracking is disabled (that is, replay is suspended), use the Secondary menu **Replay Next Journal** command. For details, see the following subsection.

» To resume journal replay from the primary or secondary server

1. Select the secondary database in the database view pane whose journals you want to resume replay.
2. Select the **Resume** command in the Secondary menu. (This command is disabled when the value of the **Tracking is** attribute in the operational state is **disabled**.)

The replaying of journals is then resumed on the selected secondary database.

Replaying the Next Journal

On a secondary database when tracking is disabled, you can initiate a replay of the next ready journal. (For details about disabling tracking, see "Enabling or Disabling Tracking", earlier in this section.)

To replay the next journal

1. Select the secondary database in the database view pane whose next journal you want to replay.
2. Select the **Replay Next Journal** command in the Secondary menu. (This command is disabled when the value of the **Tracking is** attribute in the operational state is **enabled**.)

The replaying of journals is then resumed on the selected secondary database.

Disabling a Connection to the Primary Database

You can disable a secondary connection to the primary database server to cause the secondary server to close the connection to its primary server, leaving the connection closed. This places a secondary database in an offline mode in which it can continue tracking and providing read-only database access while not receiving further journals from the primary database.

Use the **Reconnect** command to attempt to re-establish a connection to the primary server. For details, see "Reconnecting a Secondary Database", later in this chapter.

To disable a secondary database connection to the primary server

1. Select the secondary database in the database view pane whose connection you want to disable.
2. Select the **Disable Primary Connection** command in the Secondary menu. (This command is enabled only when the secondary database is connected to the primary server.)

The connection to the primary server from the selected secondary database is then disabled, the value of the **State** attribute in the operational state details changes to **disconnected**, the **Disable Primary Connection** menu command is disabled, and the **Reconnect** menu command is enabled.

Note When you disconnect a secondary database, only a subset of details for that secondary database are available for display on the primary system.

Clearing a Disrupted Network Connection

You can also use the **Disable Primary Connection** command to clear and re-establish a disrupted network link. In the event that a connection becomes disrupted because of a problem in the network path, use the **Disable Primary Connection** command to drop the failed connection and the **Reconnect** command to attempt to establish a fresh network connection.

For details, see "Disabling a Connection to the Primary Database", in the previous section.

Reconnecting a Secondary Database

To reconnect a secondary database to the primary database

- Select the **Reconnect** command in the Secondary menu. (This command is enabled only when the secondary database is disconnected from the primary database.)

The secondary is then requested to attempt to reconnect to its configured primary server.

In addition, the value of the **State** attribute in the operational state details changes to **connected**, the **Reconnect** menu command is disabled, and the **Disable Primary Connection** menu command is enabled.

Using the Options Menu

The Options menu enables you to maintain options associated with the database view.

Use the Options menu commands to perform the actions listed in the following table.

Command	For details, see…
Automatic Refresh	Automatically Refreshing the SDS Admin Database View
Show Legend	Displaying the Legend for the Database State
Font	Changing the Font Displayed in the SDS Admin Database View
Refresh Interval	Specifying the Number of Seconds at which the Database View Refreshes
Save Setting on Exit	Saving Settings when Exiting from the SDS Admin Application

Automatically Refreshing the SDS Admin Database View

You can toggle the automatic refreshing of the database view, which is automatically refreshed at the interval that you specify by using the **Refresh Interval** command. For details, see "Specifying the Number of Seconds at which the Database View Refreshes", later in this chapter.

» To toggle the automatic updating of the SDS Admin database view

- Select the **Automatic Refresh** command in the Options menu.

A check mark (✓) is displayed to the left of the command in the Options menu when the updating of the form is automatic.

For details about initiating the updating of an SDS Admin database view for the primary system or a secondary system at a specific time, see "Refreshing the SDS Admin Database View", earlier in this chapter.

Displaying the Legend for the Database State

You can toggle the display of the legend for the database state in the SDS Admin database view for the primary or a secondary database. The state of a secondary database is indicated by the background color of secondary system nodes in the database view pane.

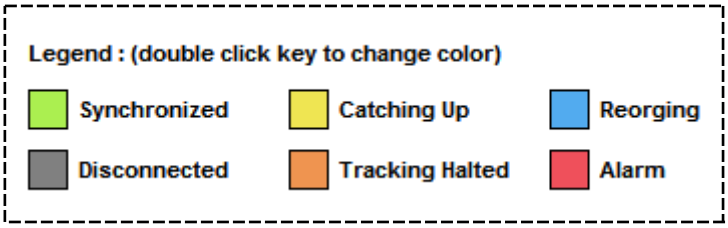
By default, the legend is not shown at the bottom of the SDS Admin database view.

» To toggle the display of the state legend in the SDS Admin database view

- Select the **Show Legend** command in the Options menu.

A check mark (✓) is displayed to the left of the command in the Options menu when the legend is displayed at the bottom of the database view pane.

The default background colors of secondary system states are shown in the following diagram.



To change a default color for a secondary database state

1. In the legend pane of the SDS Admin database view, double-click on the color square whose secondary database state you want to change. For example, double-click the square at the left of **Disconnected** if you want to change the color that indicates that a secondary database is disconnected from the primary database.

   The common Color dialog is then displayed.

2. Click on the required color. The selected color is then outlined in black. (This common dialog also enables you to define your own custom color, if required.)

3. Click the **OK** button.

The square in the legend pane and the background color of secondary databases that currently have that state are then updated accordingly. To make all your required color display changes, repeat steps 1 through 3.

Changing the Font Displayed in the SDS Admin Database View

You can change the font and font attributes (that is, the font style and size) of text in the database view pane from the default values of **Arial**, **Regular**, and **8** points.

To change the font displayed in the SDS Admin database view

1. Select the **Font** command in the Options menu. The common Font dialog is then displayed.

2. Make the font, style, and size selections that you require and then click the **OK** button. You can select a monospaced font (for example, **Courier New**) or a proportional font (for example, **System**).

When you have made your font selections, focus is then returned to the SDS Admin database view.

Specifying the Number of Seconds at which the Database View Refreshes

You can change the number of seconds at which the database view is refreshed. By default, the attributes displayed in the database view pane are refreshed every five seconds.

To specify the number of seconds at which the SDS Admin database view refreshes

1. Select the **Refresh Interval** command in the Options menu.

2. In the submenu, select the number of seconds at which you want the database view automatically refreshed.

A check mark (✓) is displayed to the left of the number of seconds that you select in the **Refresh Interval** submenu.

For details about toggling the automatic refreshing of the database view or initiating the updating of an SDS Admin database view for the primary system or a secondary system at a specific time, see "Automatically Refreshing the SDS Admin Database View" or "Refreshing the SDS Admin Database View", elsewhere in this chapter.

Saving Settings when Exiting from the SDS Admin Application

You can specify that the SDS Admin settings are saved when the application is closed. The current form size and position (that is, the window state), font, secondary database state colors, and automatic refresh settings are saved in the JADE initialization file for reloading when the SDS Administration utility is next started up.

To save your SDS Admin database view settings on exit

- Select the **Save Settings on Exit** command from the Options menu.

A check mark (✓) is displayed to the left of the command in the Options menu when settings are saved in the JADE initialization file for subsequent SDS Administration utility work sessions.

Note Saving settings when the SDS Administration utility closes down does not retain expanded nodes on SDS Admin database views (that is, all Primary system, secondary system, summary, operational state, journal state, and transaction nodes are collapsed when an SDS Admin database view is opened.

Using the Help Menu

Use the commands in the SDS Administration utility Help menu to access the standard Common User Access (CUA) help options.

The SDS Administration utility Help menu commands are listed in the following table.

Command	For details, see…	Description
Index	Accessing the Help Index	Opens the JADE online help in Adobe Reader
About	Accessing Information about the SDS Administration Utility Release	Accesses information about the release of the SDS Administration utility

Accessing the Help Index

Use the SDS Administration utility Help menu **Index** command to open online help. The online help window displays a list of topics down the left, to enable you to select the online help topic for which you require assistance.

To access the online help, perform one of the following actions

- Select the **Index** command from the Help menu
- Press F1

The JADE online help is then displayed in Adobe Reader.

Accessing Information about the SDS Administration Utility Release

Use the SDS Administration utility Help menu **About** command to access information about the current release of the SDS Administration utility; for example, the database path, the company to whom JADE is licensed, the server node operating system, and the JADE release version.

To access the SDS Administration utility version information

- Select the **About** command from the Help menu.

The About SDS Admin dialog is then displayed.

Chapter 2 Relational Population Service (RPS) Support

This chapter covers the following topics.

- Overview
- RPS Node Creation Example
- RPS Mapping
  - Primary Keys
  - Default Mapping Values
  - Special Columns
  - Special Tables
  - JADE to RDBMS Type Mapping
- RPS Node
- RPS Data Store
- RPS Datapump Application
- Column-Mapping Methods
- Table Creation and Alter Scripts
- Replicating Data
- Schema Instantiation
- Point-in-Time Snapshots
- RPS Audited SQL Script Execution
- Handling Exceptions
- Restart Handling
- Managing an RPS Node
  - Calling the RPS Manager from a Web Service Consumer
- Site-Specific RPS Mapping Customization
- Relational Population Service (RPS) Restrictions
- RootSchema-Defined Types and Features
  - Using Methods Defined in RootSchema Classes

Overview

The Relational Population Service (RPS) provides automatic replication of objects from a production JADE database to one or more Relational Database Management System (RDBMS) target databases.

The supported RPS relational target databases are Microsoft SQL Server 2000, Microsoft SQL Server 2005, Microsoft SQL Server 2008, Microsoft SQL Server 2012, and Microsoft SQL Server 2014.

Notes The RPS node must run the 64-bit version of JADE. It requires a 64-bit SQL Server ODBC driver and Data Source Name (DSN).

JADE licenses are not transferred automatically between databases in an SDE. It is your responsibility to apply new licenses to any existing databases in an SDE. In addition, to ensure proper operation, you must apply the primary license to every secondary.

An RPS system consists of:

- An RPS mapping defined in the JADE production database, which maps classes, attributes, and methods that have no parameters and that return a non-exclusive object reference of a JADE schema to relational database tables and columns.
- An RPS node (or multiple RPS nodes), which runs as a relational SDS server and executes the **Datapump** application to pump the JADE data to the RDBMS target database.
- An RDBMS target database for each RPS node, which contains the tables and columns defined in the RPS mapping.
- The RPS Manager application, which enables you to create the RDBMS tables and columns from the RPS mapping, load the data instances from the JADE production database, and control the execution of the **Datapump** application.

For details about:

- Using the wizard to specify the RPS mapping, see Chapter 15, "Using the Relational Population Service (RPS) Wizard", of the JADE *Development Environment User's Guide*
- Patch control for RPS mappings, see "Patch Control for RPS Mappings", in Chapter 3 of the *JADE Development Environment Administration Guide*

RPS Node Creation Example

» To create and run an RPS node

1. Create an RPS mapping for the selected schema, by using the Relational Population Service wizard in the JADE development environment. (For details, see "Defining an RPS Mapping", in Chapter 15 of the *JADE Development Environment User's Guide*.)
2. Configure an SDS secondary node. (For details, see "Configuring the Synchronized Database Service", in Chapter 1.)
3. Set the **DatabaseSubrole** parameter in the [SyncDbService] section of the JADE initialization file to **RelationalRole**.
4. Use the RPS Manager application to create an RPS database. For details, see "Creating the RPS Database", later in this chapter.
5. Using the appropriate RDBMS utilities, create the RDBMS database to be used with the database name specified in the RPS mapping.
6. Create an ODBC Data Source Name (DSN), by using the Microsoft Data Source Administrator program from the Control Panel **Administrative Tools** option **Data Sources (ODBC)** applet to connect to the RDBMS database.

7. Start the RPS node in your SDE.
8. Start the RPS Manager application on the RPS node. (For more details, see "Using the RPS Manager Application", later in this chapter.)
9. Start the RPS configuration. (For details, see "Configuring your RPS Node", later in this chapter.) In the Configure RPS Node dialog:
   - Specify the connection string with which to connect to the RDBMS database if you did not specify the connection string in the RPS mapping in step 1 of this instruction, in the **Default Connection String** text box.
   - Specify the database name if you did not specify it in the RPS mapping in step 1 of this instruction.
   - Specify or select the directory for script and data files, in the **Default Working Directory** text box.
   - Specify the appropriate schema and application name values in the Data Pump Application group box, if you want to specify your own non-GUI **Datapump** application. (For details, see "Configuring your RPS Node" and "Implementing User-Defined Output Control", elsewhere in this chapter.)
10. Set up the RDBMS, by selecting the **Setup RDBMS** command from the RPS menu on an RPS secondary system. The table creation scripts are then generated, the SQL scripts executed, and the data then extracted before it is loaded into the relational database. (For details, see "Setting Up RDBMS", later in this chapter.)
11. Start the **Datapump** application, by selecting the **Start Datapump** command from the Jade RPS Manager window RPS menu.

The RPS node should now track changes to the production JADE database and update the RDBMS database, as required.

RPS Mapping

The Relational Population Service Wizard, documented in Chapter 15, "Using the Relational Population Service (RPS) Wizard", of the *JADE Development Environment User's Guide*, enables you to select and map entities defined in JADE schemas to entities in a relational database.

A single JADE schema can have more than one mapping.

The wizard enables you to select the classes and many-to-many relationships to include and the tables to which they are mapped. Each RPS node uses a specified RPS mapping to populate one RDBMS database. As all mapping metadata becomes part of the system specification in the JADE database, you can extract, load, and version it.

The following table shows the entity mapping concepts.

JADE Concept or Entity	SQL Entity	Relational Concept
Class	Table	Relation
Instance	Row	Tuple
Object identifier (OID)	OID primary key column	OID primary key attribute
Attribute	Column	Attribute
Method	Column	Attribute
Reference	OID column	OID attribute

Primary Keys

The primary key used in JADE tables is **OID** for overwrite tables.

The default primary keys for historical tables are **oid**, **_edition**, **_operation**, and **_tranid**. You can add **_timestamp** as a primary key for historical tables, either in addition to or as a substitute for **_tranid**. The key order is as follows.

```
[_tranid], [_timestamp], [oid], [_edition], [_operation]
```

You can specify an alternative primary key for **No Delete** tables in the Relational Population Service wizard. For details, see "Mapping Classes to Tables" under "Maintaining RPS Column Mappings", in Chapter 15 of the *JADE Development Environment User's Guide*.

Default Mapping Values

The Relational Population Service wizard assumes and automates various default mapping rules. However, the user can alter or override many of the default mapping rules described in the following subsections.

Entity Name Mapping

Entity names used in the JADE schema map to an entity of the same name in the RDBMS database. This can be overridden in the Relational Population Service wizard. This applies to table and column names.

Class Mappings

A persistent class can be mapped to multiple tables or duplicated in several tables so that feature values of an object are split across rows in several tables. You can map multiple classes in a subclass hierarchy to a single table. Each JADE subclass instance produces a row in the table.

When subclass mapping is selected, the **_type** special column in the table is used to identify the subclass for each row. The RPS mapping contains a user-defined constant value to associate with each concrete class in the hierarchy. An example of a type hierarchy where subclass mapping might be useful is a bank account hierarchy comprising an abstract base **Account** class and several concrete subclasses representing different account types.

See also "Special Columns", later in this chapter, and "Mapping Classes to Tables" under "Adding an RPS Database Map", in Chapter 15 of the *JADE Development Environment User's Guide*.

Attribute Mapping

JADE primitive types are mapped to the closest RDBMS type supported by the target data source. Data type mapping is tailored for each target data source, because not all support the complete set of SQL data types.

For details about the default type mappings, see "JADE to RDBMS Type Mapping", later in this chapter.

Method Mapping

Any non-updating JADE method that has no parameters and returns a non-exclusive object reference or a primitive type can be mapped to a table column. These methods are referred to as *column-mapping methods*.

OID Columns

Object identifiers (oids) are mapped to a 16-character fixed-length string using a **"ccccc.nnnnnnnnnn"** format to an **INTEGER/INTEGER** data type, or to an **INTEGER/BIGINT** data type.

Every target table includes a mandatory column with the name **oid** that contains the OID of the object corresponding to the table row.

Edition

The object edition is mapped to an integer column. This column is mandatory for historical tables and it is optional for overwrite tables.

Primary Keys

Overwrite mode tables use the **oid** column as the primary key.

Historical mode tables use the combination of OID, operation, edition, and optionally transaction id and timestamp as primary keys, to ensure uniqueness.

Embedded References

Single valued references are mapped to the same type as an **OID** column.

Special Columns

The following subsections contain the special columns that you can include in a table.

Operation Column

The **_operation** special column is a character column that defines the update operation, as follows.

- **"L"**, for load
- **"C"**, for create
- **"U"**, for update
- **"D"**, for delete

Timestamp Column

The **_timestamp** special column contains the timestamp from the journal for the operation converted to local time. You can specify the **_timestamp** special column as an additional primary key in historical tables.

TranID Column

The **_tranid** special column contains a unique 64-bit transaction identifier. You can specify the **_tranid** special column as an additional primary key in historical tables.

Type Column

When a table is a subclass mapping, the **_type** special column contains a user-defined constant value associated with the class of the JADE object.

Special Tables

The following tables are defined in the RDBMS database for use by the RPS node.

Exception Logging Table

The optional Exception Logging table with the name **JADE_EXCEPTION_LOG** is created in the target database when you select table-based error logging for the database using the Relational Population Service wizard. For details, see "Handling Exceptions", later in this chapter.

Transaction Table

The mandatory table called **JADE_TRANSACTIONS** is created in each target database. This table is required to support restart recovery. See also "Restart Handling", later in this chapter.

Control Table

The mandatory **JADE_CONTROL_INFO** table is created in each target database and used in conjunction with the transaction table for handling restart recovery.

JADE_TRAN_INFO Table

The optional **JADE_TRAN_INFO** table records the transaction id, transaction start time, and transaction end time. This table has the following columns.

- **tranid bigint**
- **start_time datetime** (SqlServer 2000 or 2005) or **start_time datetime2** (SqlServer 2008 or higher)
- **end_time datetime**

The creation and use of the **JADE_TRAN_INFO** table is controlled by the **UseTranInfoTable** parameter in the [JadeRps] section of the JADE initialization file.

If the **UseTranInfoTable** parameter is defined and it is set to **true**:

- The table is created, if it does not exist
- A row is added to the table for every transaction to record id, start time, and end time
- The row is committed to the relational database as part of the transaction
- Rows are never deleted from the table by the JADE RPS node

It is your responsibility to delete the rows from the table when they are no longer required.

JADE to RDBMS Type Mapping

This section describes the mapping of JADE types to RDBMS database types and is used in conjunction with the transaction table for restart handling.

JADE to SQL Server Type Mapping

The following table defines the available mappings of JADE types to SQL Server Data types.

JADE Type	SQL Server 2000	SQL Server 2005	SQL Server 2008 or Later
Binary (unbounded)	IMAGE*	IMAGE	IMAGE

JADE Type	SQL Server 2000	SQL Server 2005	SQL Server 2008 or Later
		VARBINARY(max)*	Same as SQL Server 2005
Binary (length)	BINARY(length)	Same as SQL Server 2000	Same as SQL Server 2000
	VARBINARY(length)*		
	IMAGE		
Boolean	BIT*	Same as SQL Server 2000	Same as SQL Server 2000
	CHAR(1) or NCHAR(1)		
Byte	TINYINT*	Same as SQL Server 2000	Same as SQL Server 2000
Character	CHAR(1) or NCHAR(1)*	Same as SQL Server 2000	Same as SQL Server 2000
	TINYINT		
Date	DATETIME*	Same as SQL Server 2000	DATE*
Decimal	DECIMAL(prec, digits)*	Same as SQL Server 2000	Same as SQL Server 2000
	MONEY		
Integer	INTEGER*	Same as SQL Server 2000	Same as SQL Server 2000
Integer64	BIGINT*	Same as SQL Server 2000	Same as SQL Server 2000
JadeBytes	IMAGE*	Same as SQL Server 2000	Same as SQL Server 2000
OID	CHAR(16) or NCHAR(16)*	Same as SQL Server 2000	Same as SQL Server 2000
	INTEGER/INTEGER		
	INTEGER/BIGINTEGER		
Real	FLOAT*	Same as SQL Server 2000	Same as SQL Server 2000
String (unbounded)	TEXT or NTEXT*	TEXT or NTEXT	Same as SQL Server 2005
		VARCHAR(max)* or NVARCHAR(max)	Same as SQL Server 2005
String (length)	VARCHAR(length) or NVARCHAR(length)*	Same as SQL Server 2000	Same as SQL Server 2000
	CHAR(length) or NCHAR(length)		
	TEXT or NTEXT		

JADE Type	SQL Server 2000	SQL Server 2005	SQL Server 2008 or Later
StringUtf8 (unbounded)	NTEXT*	NTEXT	Same as SQL Server 2005
		NVARCHAR(max)*	Same as SQL Server 2005
StringUtf8 (length)	NVARCHAR(length)*	Same as SQL Server 2000	Same as SQL Server 2000
	NCHAR(length)		
	NTEXT		
Time	DATETIME*	Same as SQL Server 2000	TIME*
	INTEGER		INTEGER
TimeStamp	DATETIME*	Same as SQL Server 2000	DATETIME2*
	INTEGER		INTEGER

In this table:

- Default property values are indicated by an asterisk (*).
- Column-mapping methods that return a:
  - **String** primitive type have a default mapping type of unbounded string.
  - **Binary** primitive type have a default mapping type of unbounded binary.
  - **Binary** (length) primitive type and that are mapped to **BINARY[length]**, trailing zeros are inserted for *length* columns. If they are mapped to the default **VARBINARY[length]**, only the valid data is output.
  - **Decimal** primitive type have a default mapping type of **Decimal[38,19]**.

  These column-mapping method return types accommodate the largest possible values that could be returned.

  Tip To change the mapping for columns to a more appropriate size for the data, use the Relational Population Service wizard. For details, see "Maintaining RPS Column Mappings", in Chapter 15 of the *JADE Development Environment User's Guide*.)

- When mapping the **String** or **StringUtf8** primitive type:
  - ANSI padding is set to **off** in the SQL Server connection.
  - If the length is less than 255 characters, the map reflects the actual string length; for example, **VARCHAR(30)** or **NVARCHAR(30)**.
  - If the length is in the range 255 through 8000, the default mapping is to **VARCHAR(max)**, **NVARCHAR (max)**, **TEXT**, or **NTEXT**.
  - Unbounded strings or strings with a length greater than 8,000 map only to **VARCHAR(max)**, **NVARCHAR(max)**, **TEXT** , or **NTEXT**.
  - In SQL Server, the limit for a row in any table is 8060. You can create tables with a total column length

greater than 8060; however an error is raised if the actual data being put into a row exceeds 8060.

Note When mapping longer strings to **VARCHAR** or **NVARCHAR**, take care not to exceed this limit.

- When mapping a **Decimal** primitive type to an SQL Server **MONEY** data type, the JADE decimal attribute must have no more than 14 integer digits and no more than 3 fractional digits.
- The **NVARCHAR**, **NCHAR**, and **NTEXT** SQL Server data types are used in Unicode JADE systems. **StringUtf8** always maps to **NVARCHAR**, **NCHAR**, or **NTEXT** in ANSI or Unicode JADE systems.
- JADE **OID** types are stored in one of:
  - A fixed-string format **"65534.4294967295"** ***(<classId>.<instId>)***, by default
  - An **INTEGER/INTEGER** data type that contains the **_clsno** (class number) and **_instId** (of type **int**)
  - An **INTEGER/BIGINTEGER** data type that contains the **_clsno** (type **int**) and **_instid** (type **bigint**)
- Mapping a JADE **Boolean** primitive type to a character maps to values **'T'** and **'F'**.

Mapping Time, TimeStamp, and Date Primitive Types to the SQL Server DATETIME Type

SQL Server 2000 and 2005 do not have separate date and time data types.

When mapping the **Date**, **Time**, and **TimeStamp** primitive types in SQL Server 2000 and 2005:

- The JADE **TimeStamp** and **Time** primitive types include milliseconds, which are rounded in the SQL Server **DATETIME** data type.

  To save the exact millisecond value, map the **time** or **timestamp** property to an **INTEGER** data type.
- Due to the values allowed in an SQL Server **DATETIME** field, if the value of a **Date** primitive type or the **Date** value of a **TimeStamp** primitive type is less than **January 1, 1753** or greater than **December 31, 9999**, the value is marked as NULL and a message output to the **jommsg.log** file.

SQL Server 2008 and higher has separate date and time data types. Valid dates are in the range **0001-01-01** through **9999-12-31**.

In **Date** primitive types, the initial value of zero (**0**) is not valid and can be distinguished from a valid (or set) value or an invalid date (set by user code).

In **Time** primitive types, the initial value of zero (**0**) is a valid time of 00:00:00 and cannot be distinguished from a set value. An invalid time can be set by user code.

The following is used to map values from JADE to the SQL Server 2000 and SQL Server 2005 **DATETIME** data type.

JADE Primitive Type	JADE Value	SQL Server 2000 and 2005 DATETIME Data Type
Date	0	Null column
	InvalidDate	Null column
	Valid SQL Server date	ValidDate, 00:00:00
	Invalid SQL Server date	Null column with message to log
Time	ValidTime (including 0)	TodaysDate if **SetSQLDateForJadeTime**=<default>

JADE Primitive Type	JADE Value	SQL Server 2000 and 2005 DATETIME Data Type
		1/01/1900 if **SetSQLDateForJadeTime**=<basedate>
	InvalidTime	Null column
TimeStamp	SQLServerValidDate, ValidTime (including 0)	Date, Time
	Invalid SQL Server date, ValidTime	Null column with message to log
	ValidDate, InvalidTime	ValidDate, 00:00:00
	InvalidDate, any Time	Null column
	0, InvalidTime	Null column
	0, ValidTime (including 0)	TodaysDate, ValidTime if **SetZeroTimeStamp**=<default>
		1/01/1900, ValidTime if **SetZeroTimeStamp**=<basedate>
		Null column if **SetZeroTimeStamp**=<null> and time=0

The following table defines the available mappings of JADE types to SQL Server 2008 and higher **DATE** and **TIME** data types that differ from the SQL Server 2000 and 2005 mappings listed in the previous table.

JADE Primitive Type	JADE Value	SQL Server 2008 and Higher DATE and TIME Data Types
Date	Valid SQL Server Date	ValidDate
	Invalid SQL Server Date	Null column with message to log (SQL Server date limits)
Time	ValidTime (including 0)	ValidTime

RPS Node

A specialized SDS secondary database node implements incremental object replication to an RDBMS target database.

RPS Participant in an SDE

When an RPS node participates in an SDE, the RPS node connects to a production (or primary) database server and is similar to an SDS secondary participant in a SDE. The difference is that the RPS node connects to an RDBMS database for object replication.

An SDE can be comprised of a primary database server and any combination of secondary databases and RPS nodes.

RPS Data Store

JADE enables you to use the RPS Manager application to configure the extent of user objects from the production database that are replicated in the RPS data store. Each configurable option is also an implied lifetime; that is, temporary or persistent. In all cases, system and user schema files are replicated in the RPS data store.

The RPS node must maintain at least a subset of objects replicated from the production database.

All data to be loaded into the relational database, including data extracted for reorganizations, must be extracted on the primary node. The primary node is not available during the extract process.

Partitioned file structures and meta data are replicated on full replica and mapped extent RPS databases. File partitioning is not applicable on a working set RPS database.

The database modes (or extents) are **Full** database replica, **Mapped Extent**, and **Working Set**. For details, see the following subsections.

Full Database Replica

When you select the **Full** database replica mode, the RPS node keeps a full copy of the database synchronized with the production database.

The benefits of the **Full** database replication database mode are:

- Can be used to perform extract operations without impacting the operation of the production database.
- The replicated database can be used as an emergency database backup and for disaster recovery.
- Can acquire a primary database role in a negotiated or hostile takeover; that is, you can use a single SDS database for data recovery and RPS purposes.
- Column-mapping methods and virtual properties can access any persistent data.

This mode does not require the RPS data store to be recreated following changes to the RPS mapping.

Mapped Extent

When you select the **Mapped Extent** mode, only the persistent system and user schema database files containing classes in the relational mapping are replicated.

When a JADE database file is replicated, the entire file is replicated and not just the objects in the mapped extent.

The **Mapped Extent** mode does not support takeover from the RPS node.

The benefit of the **Mapped Extent** mode is that you can perform extract operations without impacting the operation of the production database.

Notes When a mapped extent data store has been created, adding further classes to the RPS mapping that are mapped to database files that were not included in the current mapped extent data store, will invalidate the RPS data store. You must recreate the RPS data store from the primary database. For details, see "Creating the RPS Database" and "RPS Node Creation Example", elsewhere in this chapter.

Column-mapping methods and virtual properties executed in **Mapped Extent** mode raise an exception if they access any persistent objects that are not in the files included in the mapped extent.

Working Set

When you select the **Working Set** mode, a working set of persistent objects in the transaction currently being replicated is maintained in one or more files not related to the map files defined in the production database.

The **Working Set** mode consumes the least amount of disk storage capacity but does not support extract data functions, query applications, or takeover from the RPS node.

For historical tables, String Large Object (SLOB) and Binary Large Object (BLOB) values are NULL when the object is updated.

Notes Column mapping methods and virtual properties executed in **Working Set** mode raise an exception if they access any persistent objects that are not in the current transaction.

Column-mapping methods from a subschema copy class in **Working Set** mode raise an exception during extraction on the primary node.

You can neither create nor use a working set database from or with an encrypted primary.

RPS Datapump Application

The RPS **Datapump** application **JadeRpsDataPump** runs on the RPS node from the schema in which the RPS mapping is defined. It connects to the RDBMS database and applies transactions to it, implementing the incremental object replication.

You can specify a non-GUI user-defined **Datapump** application that assumes the data pump role, by specifying the appropriate values in the **Schema** and **Application** combo boxes in the Data Pump Application group box on the Configure RPS Node dialog on an RPS node. (For details, see "Configuring your RPS Node", later in this chapter.)

The user-defined **Datapump** application can be used to set up the application and global environment needed to execute column-mapping methods. The user-defined **Datapump** application must call the **Application** class **rpsDataPumpInitialize** method from its **initialize** method. This method registers the callbacks required to implement incremental object replication. It must also call the **Application** class **rpsDataPumpFinalize** method from its **finalize** method to perform any terminate functions for the **Datapump** application.

When you specify a user-defined **Datapump** application, any objects required for application start-up and user validation must be present in the RPS system, which may require the RPS system to be **Full** database replication mode. For example, if the system uses packages, the global instance of the schema from which the package is imported must be present for the application to start up.

To identify whether a process is executing as a **Datapump** application on an RPS node, call the **Process** class **isUserDataPump** method, which returns **true** if the process is executing as a **Datapump** application; otherwise **false**.

The **Datapump** application is initiated in the schema in which the RPS mapping is located, so that the packages are initialized and context-switching occurs if imported methods are called on the imported class.

The following actions start the RPS **Datapump** application.

- Automatic startup.
  - Check the **Auto Start Datapump** check box on the Configure RPS Node dialog on the RPS node. (For details, see "Configuring your RPS Node", later in this chapter.)

  To allow the **Datapump** application to be started automatically, it must be able to log on to the RDBMS without user intervention. In SQL Server, you can accomplish this by running the RPS node under a Windows login that has the required access rights and database permissions and then setting the DSN properties appropriately.
- Programmatically, by calling the **JadeDatabaseAdmin** class **rpsStartDataPump** method, which must be run on an RPS node.

- Manual startup.
  - Select the **Start Datapump** command in the RPS menu in the Jade RPS Manager window. (For details, see "Starting the Data Pump" under "Using the RPS Manager Application", later in this chapter.)
- Extracting data from and loading data into the relational database.

  This allows a user-defined **Datapump** application to initialize **global** and **app** for column-mapping method values and allows package initialization for imported classes.

On startup, the **Datapump** application checks that the:

- RPS node database type is compatible with the RDBMS database that is being used; for example, if the RPS mapping is to SQL Server 2000 and the RDBMS database is SQL Server 2008 or later, the data types will be those defined in SQL Server 2000. If the mapping is to SQL Server 2008 or later and the RDBMS database is SQL Server 2005, an error prevents the **Datapump** application from running.
- RPS node database type is compatible with the ODBC driver being used; for example, if the mapping is SQL Server 2008 or later, the SQL Server 2008 or later Native Client ODBC driver must be used.
- Control information from the JADE database matches the control information in the RDBMS database. If it does not, check the connection information to ensure that the correct database is being used.

  To resynchronize the database, see "Resetting the Relational Database Identifier", later in this chapter.
- Table and columns defined in the RPS mapping match the RDBMS database. If they do not, correct the RDBMS database and restart the **Datapump** application.

To stop the **Datapump** application, select the **Stop Datapump** command from the RPS menu in the Jade RPS Manager window. (For details, see "Stopping the Data Pump" under "Using the RPS Manager Application", later in this chapter.)

In addition, you can stop the Datapump application programmatically, by calling the **JadeDatabaseAdmin** class **rpsStopDataPump** method.

For details about specifying the location of errors file created when the **Datapump** application on an RPS node terminates abnormally, see "Fault Handling when the Datapump Application Terminates Abnormally", later in this chapter.

The [JadeRps] section of the JADE initialization file contains parameters that enable you to control the timeout occurring between an RPS server and an SQL Server. These parameters are read when **Datapump** application is started.

Parameter	Specifies the number of seconds to wait for…
ConnectionTimeout	Connection between an RPS server and an SQL server. The default value is **15** seconds. Specify zero (**0**) if you do not want the connection to time out.
LoginTimeout	Log-in to the SQL server from the RPS server. The default value is **15** seconds. Specify zero (**0**) if you do not want the log-in to time out.
QueryTimeout	A query result. The default value is **60** seconds. Specify zero (**0**) if you do not want the query to time out or **-1** to not make the query call to set time out (which saves a call to the SQL Server).

Implementing User-Defined Output Control

To implement user-defined output control within a user-defined **Datapump** application, you must define a class that implements the **JadeRpsDataPumpIF** interface and its **updateCallback** method, as shown in the following example.

```
rpsDp_UpdateCallback(tranID: Integer64; timestamp: TimeStamp; oid: Object;
                     operation: Integer; tableName: String;
                     dropRowOperation: Boolean io;
                     outputRowInfo: JadeDynamicObject io) updating;
vars
    count        : Integer;
    indx         : Integer;
    int          : Integer;
    jdoPropName  : String;
    jdoPropValue : String;
begin
    // only Company is set as OutputControl in the RPS mapping,
    // so only rows for the table Company will be passed to this routine
    count := outputRowInfo.propertyCount();
    // log information about updates to this table
    writeToMyLog("update callback=" & tranID.String & "; operation=" &
                 operation.String & ";timestamp=" & timestamp.String &
                 ";oid=" & oid.String & ";tableName=" & tableName);
    writeToMyLog('---' & outputRowInfo.getName & ' ' &
                 outputRowInfo.getOidString & " " & count.String & ' ---');
    if operation = JadeRpsDataPumpIF.ObjectDeleted then
        return;
    endif;
    // log information about each column in row
    foreach indx in 1 to count do
        jdoPropName := outputRowInfo.getPropertyName(indx);
        jdoPropValue := outputRowInfo.getPropertyValueByIndex(indx).String;
        log.info(jdoPropName & " = " & jdoPropValue & ".");
      // look for column for "phone" and update value
      if jdoPropName = "phone" then
          if jdoPropValue.String = "(64) 3 555 4519" then
             outputRowInfo.setPropertyValueByIndex(indx, "(64) 3 999 4519");
             writeToMyLog("   phone changed to (64) 3 999 4519");
          endif;
      endif;
   endforeach;
   // if required, set dropRowOperation parameter to true to not output the
   // row operation to the relational database
end;
```

The **updateCallback** method enables you to inspect, modify columns, or drop rows as the rows are being output to the relational database table.

For create and update operations, the attribute values are populated with the values obtained from the JADE object, including values returned by user-defined column-mapping methods.

The application callback can interrogate attributes and change attribute values within the **JadeDynamicObject**.

Your data pump application must first call the **Application** class **rpsDataPumpInitialize** method, whose **receiver** parameter is an instance of your class that implements the **JadeRpsDataPumpIF** interface.

When database tracking commences, your user-defined **Datapump** application is driven by callbacks invoked in a defined transaction replication cycle.

Not all tables are likely to require output manipulation. To avoid the overhead of creating the dynamic objects and calling the user routine when not required, the Relational Population Service wizard enables you to select the tables for which the callback will be used.

By default, tables are not included in callbacks. (For details, see "Maintaining RPS Column Mappings", in Chapter 15 of the *JADE Development Environment User's Guide*.)

Selecting output callback for a table has no effect if callbacks are not registered in your **Datapump** application or if you have not defined your own **Datapump** application.

Column-Mapping Methods

The Relational Population Service wizard in the JADE development environment enables you to map a table column to a column-mapping method executed on an RPS node to return a derived value. (For details about using the wizard to specify the RPS mapping, see Chapter 15, "Using the Relational Population Service (RPS) Wizard", of the *JADE Development Environment User's Guide*.)

The user-defined column-mapping methods can access other database objects (in addition to **self**). These methods execute on the RPS node and can access persistent instances. The RPS node (which must be **Full** replication mode, to access all object) runs in a mode similar to a native (JADE) SDS system, with kernel cache coherency, transaction isolation, and share lock semantics.

The data pump replication process invokes column-mapping methods when the associated column is to be output, and converts the result to meet the requirements of the relational database.

Column-mapping methods cannot update persistent objects and when invoked by the default **Datapump** application, cannot access application context.

If the methods require a user application context to execute, you must set a user-defined **Datapump** application. The value of the method column remains in the RDBMS database until that object is updated in a subsequent transaction.

When you specify a file name in the **LogNullColumnOnException** parameter in the [JadeRps] section of the JADE initialization file, exception information is written out to the specified file for any handled exceptions in column-mapping methods and virtual properties. If no file name is specified (the default value), no exception information is logged for handled exceptions in column-mapping methods and virtual properties.

Changing the method source does not version the schema or automatically re-populate the affected relational database tables. If repopulation is required, you must version the schema manually, modify the method in the versioned schema, and stop the **Datapump** application on instantiation so that you can reload the data manually.

Handling Exceptions in Column-Mapping Methods

Parameters in the JADE initialization file enable you to control the handling of exceptions in column-mapping methods.

Note You should handle any "expected" exceptions that may occur in a column-mapping method by your JADE user code in an exception handler set from within the column-mapping method.

Global exception handlers are *not* called for column mapping methods have exceptions being handled.

If the default value of **none** is specified for the **NullColumnOnException** parameter in the [JadeRps] section of the JADE initialization file, any exceptions raised in column-mapping methods are passed back to the **Datapump** application exception handler, which terminates the **Datapump** application. The default exception information is logged, and you can then perform one of the following actions.

- Set the **NullColumnOnException** parameter value to **all** or **selected** to set the column value to null for the method or virtual property, and then restart the **Datapump** application.
- Fix the problem that caused the exception and recreate the RPS database.

Set the parameter to:

- **all**, if you want any exceptions raised by column-mapping methods to be handled in the **Datapump** application and the value of the column in the RDBMS database set to **Null**.
- **selected**, if you want exception handling and the setting of the column in the RDBMS database to a **Null** value to apply only to column mapping methods and virtual properties specified in the [RpsIgnoreMethodExceptions] section of the JADE initialization file.

When the value of the **NullColumnOnException** parameter is set to **selected**, column-mapping method and virtual property exceptions are handled only for the specified class and method or virtual property name values listed in the [RpsIgnoreMethodExceptions] section of the JADE initialization file.

The [RpsIgnoreMethodExceptions] section of the JADE initialization file can contain one or more **Method<*n*>** parameters, which have a *class-name*::*method-name* or *class-name*::*virtual-property-name* value. The **<n>** value of the parameter syntax indicates a unique number (**1**, **2**, **3**, and so on) in ascending order; for example:

```
Method1 = MyRpsClass::firstMethod
Method2 = AnotherRpsClass::secondMethod
Method3 = Client::deleteClient
```

Table Creation and Alter Scripts

Creation scripts contain SQL commands to create stored procedures and tables and their columns.

Alter scripts contain SQL commands to modify an existing stored procedure, table, or column so that it matches changes that have been made to the JADE classes, properties, column-mapping methods, and so on, to keep the target RDBMS database consistent with the JADE database. A generated alter script contains the SQL DDL commands required to modify existing tables, columns, and stored procedures that bring into effect the changes that were made in the RPS mapping or mapped classes in the primary database. (For details, see "Alter Scripts" under "Schema Instantiation", later in this chapter.)

For details about generating creation and alter scripts, see "Generating Table Creation SQL Scripts" and "Generating Table Alter SQL Scripts", later in this chapter, or the **RelationalView** class **generateRpsTableCreationScript** method in Volume 2 of the *JADE Encyclopaedia of Classes*.

Replicating Data

Data is replicated in a relational database using one of the following actions.

- You must initiate an initial extract and load to populate an empty relational database from a stable snapshot of the JADE production database. This is normally done when the RPS data store is created for use on the RPS node. This replicates the mapped extent for an RPS mapping to a selected target database.
- The incremental update process using the **Datapump** application keeps the target relational database synchronized with the production JADE database.

As application transactions modify or create objects in the native production database, the operations required to perform these changes are written to the database journal. An automated tracking process uses the information stored in the database journal to keep the relational database synchronized with the mapped extent from the production JADE database.

The tracking process ensures transactional consistency of the mapped extent and that all updates for committed transactions not explicitly excluded by the mapping or filtering are faithfully replicated in the target database.

- When reorganization and schema instantiation require new data to be loaded into the relational database, tables may need to be dropped and reloaded.

Controlling Replication from the Primary

To refresh selected objects in the target database from an application running in the production database, call the **Object** class **updateObjectEdition** method, which increments the edition of the object specified in the method parameter by one.

In an RPS context, an application on the primary can use this method to perform a null update operation on a selected object. Null updates are audited on the primary and applied by RPS replication to the relational target.

Filtering Transactions from the Primary System

On the primary system, you can specify the transactions for which deletions are *not* to be replicated to the relational database. This enables you to perform selected operations on the primary system (for example, housekeeping and archiving tasks) without affecting relational database replicas.

The **Process** class provides the **rpsSuppressTransactionDeletes** method that can be called by your application of the primary system after a **beginTransaction** instruction is executed.

Invoking the **rpsSuppressTransactionDeletes** method on a primary database within a database transaction marks the transaction so that objects deleted within the transaction are not replicated to RPS Relational targets.

You can call the **rpsSuppressTransactionDeletes** method anywhere between the execution of the **beginTransaction** and **commitTransaction** instructions; it does not have to precede the first object deletion. The suppression of the deletion is reset when the process exits from transaction state and applies to all deletes within the transaction.

Invoking the method outside transaction state raises an exception.

Schema Instantiation

When a schema instantiation on the RPS node includes changes to the RPS mapping, the DDL scripts to define new tables or redefine existing tables are generated by the RPS node, and when possible, applied to the target relational schema.

When an RPS mapping is versioned, a **mapping check** audit record is written to the journal on the primary database. The **mapping check** audit record contains a list of schemas and RPS mappings that were versioned. This audit record is recognized and replayed by RPS nodes and ignored by native SDS databases.

When an RPS node replays a **mapping check** record and a versioned RPS mapping matches the configured mapping, the current and latest versions of the RPS map are compared. Any differences are used to generate a Data Definition Language (DDL) script that can be applied to the RDBMS target database, to restore schema consistency.

Replaying a schema instantiation on an RPS node configured for **Full** or **Mapped Extent** replication also performs any required reorganization to persistent objects replicated in the database. A database configured for **Working Set** replication skips reorganization steps, but replays other required aspects of the schema instantiation.

Replaying a schema instantiation requires that the **Datapump** application and database tracking are halted. This halt in tracking occurs even if no changes to the target relational schema are required, in order to satisfy standard requirements for transition to a new schema version.

The **Datapump** application and database tracking can be automatically restarted when the conditions documented in the following subsections are met.

Alter Scripts

A generated alter script contains the SQL DDL commands required to modify existing tables, columns, and stored procedures that bring into effect the changes that were made in the RPS mapping or mapped classes in the primary database.

The alter scripts generated by an RPS node for instantiation are created with the following format.

```
extract_mapping-OID/tableAlter_RPS-mapping-name_mapping-OID.sql
```

The *mapping-OID* value contains the object identifier of the RPS mapping entity and it is appended to the file name to ensure uniqueness.

The alter scripts are written to a subdirectory of the path specified in the **Alter Script Path** text box on the Configure RPS Node dialog in the RPS Manager application. (For details, see "Configuring your RPS Node", later in this chapter.)

Two types of alter script can be generated. The first is an **auto initiate** script that makes use of SQL **ALTER TABLE** commands to modify tables in the target. The second is a **Drop/Reload** script that renames any affected tables and creates new empty tables for reloading of the data.

Comments included at the top of the script describe what has changed and what actions the script will take, as shown in the following example.

```
-- The following change(s) to the table 'Agent'
-- new column 'webSite'
-- could cause an incompatibility between your JADE and
-- SQL databases.  Therefore this table will be renamed
-- and a new one created.  You will also need to extract
-- table data from the JADE database and load it into
-- this new table.  Table renamed rather than dropped,
-- it can be dropped manually when no longer required.
```

The following JADE schema changes are achieved using **ALTER TABLE** commands and result in an **auto initiate** script.

- Changes to the JADE schema, as follows.
  - Adding a property to a class mapped to a table and adding the new property to the table.
  - Changing a JADE primitive type from **Byte** to **Integer** or **Integer64**.
  - Changing a JADE primitive type from **Integer** to **Integer64**.
  - Deleting a property or column-mapping method used in a table if the table is in **Overwrite** mode.
  - Increasing the length of a **String**, **StringUtf8**, or **Binary** property used in a table.

- Changes to the RPS mapping, as follows.
  - For historical tables only, adding an existing property or method to a table if the **DropHistoricalTableOnAddExisting** parameter in the [JadeRps] section of the JADE initialization file is set to **false**.

    Any existing rows will have a column value of **NULL**.
  - Removing a property or column-mapping method used in a table from a table, if the table is in **Overwrite** mode.
  - Changing the mapping type of **String**, **StringUtf8**, or **Binary** columns.

The following JADE schema changes result in a **Drop/Reload** script using a renaming of the old table and a creation of the new table to achieve the table redefinition.

- Changes to the JADE schema, as follows.
  - Deleting a property or column-mapping method used in a table, if the table is in **Historical** mode.
  - Changing the JADE primitive type of a property or column mapping method that is used in a table, except for the cases documented earlier in this section.
- Changes to the RPS mapping, as follows.
  - Adding a property or column mapping method, except for historical tables with the **DropHistoricalTableOnAddExisting** parameter in the [JadeRps] section of the JADE initialization file set to **false**.
  - Adding an existing property or column-mapping method to a table.
  - Adding a filter constraint to or removing a filter constraint from a table.
  - Changing the mapping type of a column (except for length changes to a **String**, **StringUtf8**, or **Binary** property used in a table).
  - Changing the value in the **OID Mapping Options** combo box on the **Define RPS** sheet of the Relational Population Service wizard in an existing RPS mapping from **Map to String** or **Map to String (7.0 format)** to **Map to String (7.1 format)**.
- Changes to the database type, as follows.
  - For **DATETIME** to **DATETIME2** conversion, data is *not* reloaded on change. The millisecond values in the data at the time of the upgrade therefore may not be exact matches to the JADE data.

    Note If you require the data to match, you must manually extract and reload the required tables. For details, see "Extracting Data from the JADE Database" or "Extracting Data and Loading it into the Relational Database", later in this chapter.
  - **ALTER COLUMN** for **VARCHAR**(max) and **VARBINARY**(max).
  - **ALTER COLUMN** for date, time, or timestamp conversion.

Datapump Application Execution

On an RPS node, the **Datapump** application must always be in a state in which it can be executed without error.

If schema changes put the **Datapump** application into a non-executable state, the RPS node cannot continue replay and must be recreated from the primary.

The **Datapump** application can be put into a non-executable state in various ways, including:

- Calling uncompiled methods during application startup.
- Additions or changes to imported packages in the **Datapump** application schema.

To avoid re-creation of the RPS node from the primary, when making schema changes on the primary:

- Avoid schema changes in the **Current** version but using the **Latest Schema Version** when loading schemas.
- Initiate the transition only when a consistent set of changes has been loaded.

Restarting the Datapump Application

When a reorganization transition is replayed, the **Datapump** application and tracking halts. The **Datapump** application restarts automatically if the following conditions are satisfied.

- The RPS **Datapump** application is configured to start automatically; that is, the **Auto Start Datapump** check box is checked on the Configure RPS Node dialog, which sets the **AutoStartDataPump** parameter in the [JadeRps] section of the JADE initialization file to **true**.
- The RPS node is configured to sign-on and authenticate with the target RDB server without user intervention.
- One of the following applies.
  - The RPS mapping has not been modified by the reorganization transition.
  - The RDBMS schema requires modification and the alter script is designated an **auto initiate** script or no changes are required to the RDB schema.
  - The alter script requires table redefinition, data extraction, and subsequent loading into the RDBMS, and the **Auto Execute Drop/Reload** check box is checked on the Configure RPS Node dialog.

For details, see "Configuring your RPS Node", later in this chapter. For details about automatically restarting the **Datapump** application after a connection or timeout error, see "RPS Datapump Application", earlier in this chapter.

Automatically Initiating Scripts

When the **Datapump** application is restarted automatically, **auto initiate** scripts are executed without user intervention.

When an **auto initiate** script has been applied, consistency checking is performed and if no errors are detected, database tracking and replication resumes.

Automatically Initiating Drop/Reload Scripts

To enable the automation of reorganizations with **Drop/Reload** scripts, check the **Auto Execute Drop/Reload** check box on the Configure RPS Node dialog on the RPS node. (For details, see "Configuring your RPS Node", later in this chapter.). By default, the dropping and reloading of scripts is not automatically executed.

By default, data for historical tables is not loaded when the table is recreated. To enable the automatic loading of data for historical tables, check the **Bulk Load Historical Tables** check box on the Configure RPS Node dialog. (For details, see "Configuring your RPS Node", later in this chapter.)

Note Extracting and loading the tables may take significant time, depending on the size of the data to be extracted and loaded.

The **ExtractBufferSize** parameter in the [JadeRps] section of the JADE initialization file on the RPS node specifies the buffer size that is allocated for each concurrent file being written when extracting RPS files. This parameter defaults to **1M**.

If you automate the extraction of table data for the **Working Set** mode, the data must be extracted on the primary SDS node during the transition phase of the reorganization. As the data is then loaded from the extracted files on the RPS node, a mapped disk share must therefore be available between the primary SDS node, the JADE RPS node, and the RDBMS database.

The [JadeRps] section of the JADE initialization file on the primary can contain the **AutoExtractOnPrimary*<n>*** parameter, which determines the automation of data extraction for the **Working Set** mode by specifying that RPS mapping changes and data are to be extracted when a reorganization takes place on an SDS primary node that is supporting an RPS database operating in **Working Set** database replication mode. For details, see the *JADE Initialization File Reference*.

The following is an example of the [JadeRps] section of the JADE initialization file on the primary node to extract data for an RPS node that has a **Working Set** database replication mode.

```
[JadeRps]
AutoExtractOnPrimary1 = ErewhonInvestmentsModelSchema,
                        DocumentationExample, z:\jade\rps1, c:\jade\rps1,
                        ErewhonTables, true, ClientExecution, RpsServer
AutoExtractOnPrimary2 = LibraryExampleSchema, LibraryExample,
                        z:\jade\rps2, c:\jade\rps2, LibraryTables, false,
```

Manual Table Redefinition

If the **Auto Execute Drop/Reload** check box is not selected on the Configure RPS Node dialog and a **Drop/Reload** script is required, an administrative user must intervene to apply the table redefinitions, repopulate, and restart the **Datapump** application.

When the RPS node reconnects to the target database, RPS verifies table consistency before continuing.

An expert database administrator may be able to accomplish the required changes in-place, avoiding the need to completely reload the table. In cases where an administrative user must drop the affected tables and reload them, the generated script can be run and the **Extract & Load** command from the Extracts menu in the Jade RPS Manager window can be used for selected tables to manually apply the required changes.

To reload a table by executing the manual alter script generated by RPS directly without alteration, use the RPS Manager application, as follows.

1. If you accept the default table replacement strategy, execute the generated alter script to rename the original table and create a new empty table.
2. Extract objects for selected tables from the primary database (for the **Working Set** mode) or the RPS database (for the **Full** or **Mapped Extent** mode), using the RPS Manager application. For details, see "Extracting Data from the JADE Database" or "Extracting Data and Loading it into the Relational Database", later in this chapter.
3. Execute the generated bulk insert script (the **bulkInsert.sql** file) to load the extracted data. For details, see "Executing an SQL Script", "Extracting Data and Loading it into the Relational Database", or "Loading Extracted Data into the Relational Database", later in this chapter.

Tip Use the **Extract & Load** command from the Extracts menu in the RPS Manager application to execute the generated bulk insert script and to perform an extract and load as a single action. For details, see "Extracting Data and Loading it into the Relational Database", later in this chapter.

Table Consistency Checking

A table consistency check is performed each time the **Datapump** application connects to the relational target.

The table consistency check imports the definition of all tables associated with class mappings for the database and checks that column names and types match those defined in the JADE schema. This consistency checking detects cases where a table definition or redefinition has not been applied or you have altered the definition of a target table for some reason, introducing a mismatch.

When a mismatch is detected, details about the mismatch are logged and the connection attempt fails. An administrative user must correct the mismatch before retrying the RPS connection.

Point-in-Time Snapshots

You can obtain a point-in-time snapshot of the primary database by causing the tracking process on the RPS node to be paused at a specific time. To facilitate synchronizing this specific time with the primary database, call the **JadeDatabaseAdmin** class **sdsAuditStopTracking** method, which causes a **Stop Tracking** marker to be audited when it is invoked on the primary.

The stop tracking point does not have to coincide with a database quiet point (that is, one at which there are no active transactions).

When tracking is paused on the RPS node, incomplete transactions are not reflected in the target database.

RPS Audited SQL Script Execution

When the **JadeDatabaseAdmin** class **rpsAuditSqlScriptForReplay** method is invoked on the primary, it causes a special RPS callback audit record to be written to the current journal.

The callback record includes the contents of the **sql** parameter of the method.

When an RPS node replays this callback audit record, the contents of the SQL string stored in the record are passed to the **DataPump** application for execution. If the script was audited within a transaction on the primary, it is executed before that transaction has been replicated to the target relational database.

Error Handling Policy

If execution of the SQL script encounters an error, the script contents are saved to a file and database replication is halted. The error file has a name of the **Replay_*YYYYMMDD_HHMMSS*.log** format and it is saved in a **Failed** subdirectory of the directory specified in the **AutoScriptPath** parameter in the [JadeRps] section of the JADE initialization file.

The replication halt allows an administrative user to determine the cause of the failure and to take corrective action, if required. When replication is restarted by starting the data pump or by restarting the RPS node, any audited SQL scripts that failed execution previously are skipped.

Handling Exceptions

Exception handling policies and logging options are defined for an RPS mapping, as follows.

- Halt on exception

  The offending transaction is aborted and database tracking ceases. When database tracking is halted, the tracker process causes an event to which applications executing on the primary database can subscribe. When tracking halts, this is represented as an alarm condition by the JADE SDS Administration application.

- Take a specified alternative action for the operation

  The alternative action exception policy applies only to certain benign exceptions, which are listed in the following table.

SQL Statement	Exception	Alternative Action
INSERT	Row exists	Convert to UPDATE
UPDATE	Updates zero rows	Convert to INSERT
DELETE	Deletes zero rows	Ignore

  An **UPDATE** or **DELETE** operation that affects more than one row is always fatal. Any unexpected exception is considered fatal. Fatal exceptions trigger the halt on exception policy. The alternative action policy does not apply to historical tables. As there are no defined benign exceptions for historical tables, any exception is considered fatal.

You can override these values by using the Configure RPS Node dialog on the RPS node, which provides error and log mapping override options. For details about these exception-related mapping override options, see step 3 under "Configuring Your RPS Node", later in this chapter.

For details about:

- Specifying RPS database map create, update, and delete statement exception policies, see "Setting Up the RPS Options", in Chapter 15 of the *JADE Development Environment User's Guide*.
- Disabling automatic process dumps generated when the **Datapump** application encounters an exception replicating the effects of a transactions, see "Disabling Datapump Application Automatic Process Dumps", later in this chapter.
- Automatically restarting the **Datapump** application after connection or timeout errors, see "Automatic Restart on Connection or Timeout Errors", later in this chapter.
- Controlling the handling exceptions in column-mapping methods, see "Handling Exceptions in Column-Mapping Methods", earlier in this chapter.

Fault Handling when the Datapump Application Terminates Abnormally

The [JadeRps] section of the JADE initialization file can contain the **DatapumpErrorFileDirectory** parameter, which specifies the location of the error file created when the **Datapump** application on an RPS node terminates abnormally. (For details about the **Datapump** application, see "RPS Datapump Application", earlier in this chapter.)

The *location* value is the location in which the .**err** files will be created. If this parameter does not exist in the initialization file, no .**err** files are created. The *location* value follows the same rules as those of the **LogDirectory** parameter in the [JadeLog] section of the JADE initialization file.

If the **Datapump** application fails with an error (that is, connection errors, table mismatch errors, row creation, update, or delete errors, table inconsistency, or extract or load errors), the file ***location*/rps_*unique-id*.err** is created.

The file contains available information about the error, including any ODBC errors reported, the exception stack or both ODBC errors and the exception stack.

Automatic Restart on Connection or Timeout Errors

If the **Datapump** application fails with a connection or timeout error, the [JadeRps] section of the JADE initialization file provides the following parameters to automatically restart the **Datapump** application.

```
[JadeRps]
AutoRestartOnError=true
AutoRestartRetryLimit=10
AutoRestartDelay=10
```

The **AutoRestartOnError** parameter specifies whether or not the **Datapump** application will be automatically restarted after a connection or timeout error. The default value is **false**. If the **AutoRestartOnError** parameter is set to **true**, the **AutoRestartRetryLimit** parameter specifies the number of retries (the default value is 10) and the **AutoRestartDelay** parameter specifies the number of seconds to wait between retries (the default value is 10). See also "Restarting the Datapump Application", later in this chapter.

The following ODBC error states define connection or timeout errors that trigger the automatic restart (if the value of the **AutoRestartOnError** parameter is **true**).

ODBC Error Number	Description
08001	Unable to establish connection
08004	Server rejected the connection
08007	Connection failure during transaction
08S01	Communication link failure
HYT00	Timeout expired
HYT01	Connection timeout expired.

Disabling Datapump Application Automatic Process Dumps

When the **Datapump** application encounters an exception replicating the effects of a transaction, a process dump is taken and recorded in the **jommsg.log** as follows:

```
RPS: **** DataPump Exception: Invoking diagnostic process dump *****
JomLog: >>> Process Dump requested <<<
JomLog: >>> Dumping Process memory and O/S handles ....
JomLog: >>> Process Dump complete, dump file:   dump-file-name
```

To disable this automatic process dump, specify the **DumpOnReplicationException** parameter with a value of **false** in the [JadeRps] section of the JADE initialization file.

Exception Logging

All exceptions and fatal errors are logged to the **jommsg.log** file. In addition, you can use the Relational Population Service wizard or the Configure RPS Node dialog to specify whether you want to record exceptions in a table in the target database. Each logged exception entry contains the following fields.

- Timestamp of the operation from the journal
- Transaction ID
- Error code and description

For details, see "Setting Up the RPS Options" under "Adding an RPS Mapping", in Chapter 15 of the *JADE Development Environment User's Guide*. See also "Exception Logging Table" under "Special Tables", earlier in this chapter.

Restart Handling

When an RPS node is interrupted or shut down with in-progress transactions, restart handling ensures that no transactions are lost and that no previously committed transactions are redone.

Managing an RPS Node

You can run the RPS Manager application on the primary server or RPS node. The commands that are available in Jade RPS Manager window menus depend on the server on which the application runs.

Using the RPS Manager Application

The RPS Manager application enables you to administer your RPS node. The following is an example of a command line required to run the **RPSManager** application.

```
c:\jade\bin\jade.exe schema=JadeMonitorSchema app=RPSManager name=SiteB
path=c:\jade\system ini=c:\jade\system\jade.ini server=multiUser
```

You can run the RPS Manager application on a primary server or a secondary RPS node. The menu bar on the Jade RPS Manager window contains the menus listed in the following table.

Menu	Enables you to…
File	Configure the RPS node, reset the relational database identifier, execute an SQL script, perform administrative functions, and exit from the RPS Manager application
RPS	Start and stop the **Datapump** application, create the RPS database, and back-up the RPS database
Extracts	Extract and load data, and generate SQL scripts
Help	Access the standard Common User Access (CUA) help options

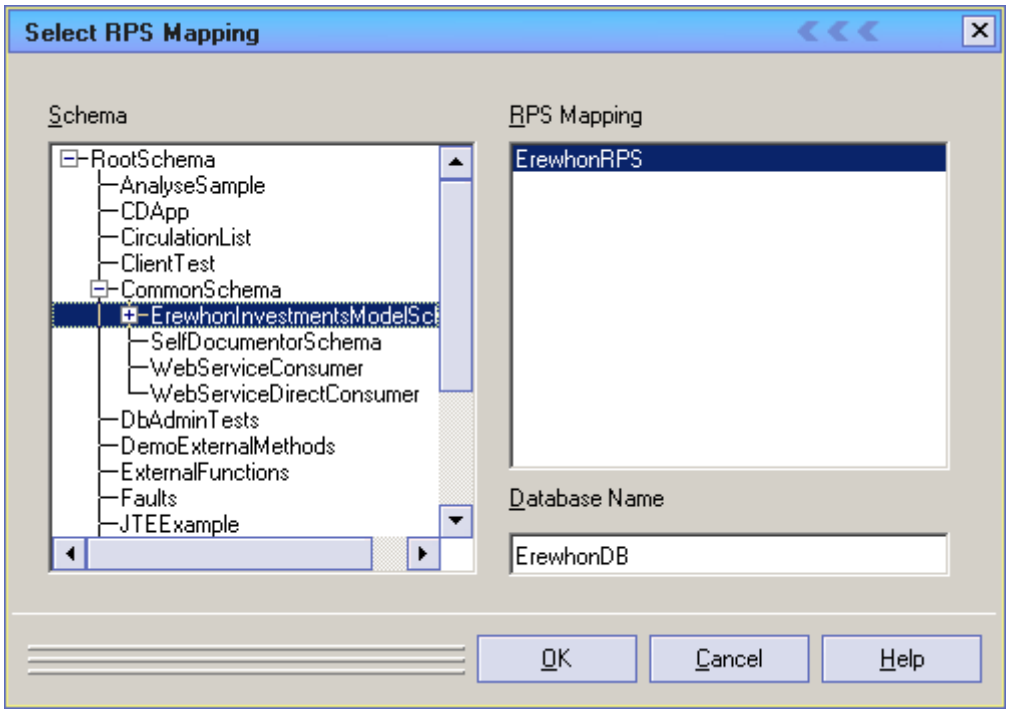
Selecting Your RPS Mapping

On an RPS node, the schema and mapping are defined when you create the RPS node, and you cannot change them.

Note You can select your RPS mapping on an RPS node only if it is being created from a backup of the primary.

On a primary server, the Select RPS Mapping dialog, shown in the following diagram, is displayed when you:

- First access the RPS Manager application and you have not yet selected an RPS mapping
- Click the **Change** button in the Mapping group box



To select your schema and RPS mapping

1. In the **Schema** and **RPS Mapping** list boxes, select the schema and the RPS mapping defined in that schema whose RPS node you want to use.
2. In the **Database Name** text box, specify the name of the target RDBMS database.

   The default value is the database name specified in the **Default Database Name** text box on the **Define RPS** sheet of the Relational Population Service wizard. (For details, see "Setting Up the RPS Options", in Chapter 15 of the *JADE Development Environment User's Guide*.)

   If the target RDBMS database is not specified in the RPS Manager application *or* the Relational Population Service wizard, an RDBMS error may be raised when you attempt to execute the SQL script.
3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

Using the File Menu

Use the File menu commands to perform one of the actions listed in the following table.

Command	For details, see…	Applies to…
RPS Node Configuration	Configuring your RPS Node	RPS node only
Reset Relational Database ID	Resetting the Relational Database Identifier	RPS node only

Command	For details, see…	Applies to…
Consistency Check	Checking the Consistency of the JADE and RDBMS Databases	RPS node only
Execute SQL Script	Executing an SQL Script	RPS node only
Print	Printing Text Displayed in the Jade RPS Manager Window	Primary and RPS node
Font	Selecting the Jade RPS Manager Window Display Font	Primary and RPS node
Clear Display	Clearing Text from the Jade RPS Manager Window	Primary and RPS node
Exit	Exiting from the RPS Manager Application	Primary and RPS node

Actions that do not apply to the current database role are disabled.

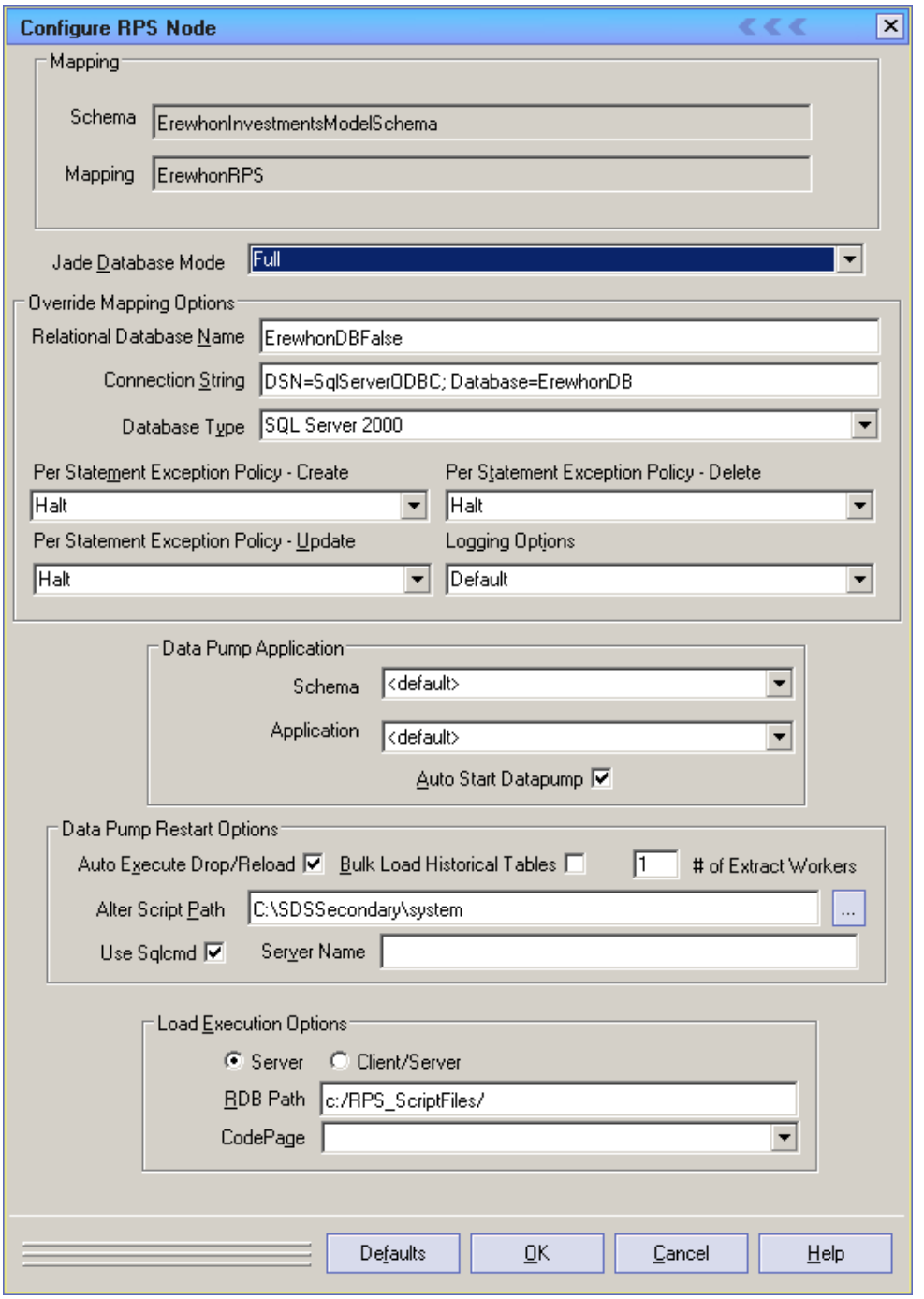
Configuring your RPS Node

You can configure an RPS node only when the **Datapump** application is not running.

To configure the RPS node on a secondary RPS system

1. Select the **RPS Node Configuration** command from the File menu.

   The Configure RPS Node dialog, shown in the following diagram, is then displayed.



The **Schema** and **Mapping** text boxes in the Mapping group box display the schema and mapping for which the RPS data store was created. As these values are read-only, you cannot change them.

2. The **Jade Database Mode** combo box displays the mode in which the RPS data store was created. It also enables you to change the database mode. You can change:
   - **Full** mode to **Mapped Extent** or **Working Set**
   - **Mapped Extent** mode to **Working Set**

   You cannot change the **Working Set** mode.
3. The default values in the Override Mapping Options group box are the values defined on the **Define RPS** sheet of the Relational Population Service wizard. (For details, see "Setting Up the RPS Options", in Chapter 15 of the *JADE Development Environment User's Guide*.)

   To override any default mapping values that you require:

   a. Specify the name of the target RDBMS database in the **Relational Database Name** text box. If you do not specify the target RDBMS database, an RDBMS error may occur when you attempt to execute the generated SQL scripts.

   b. Specify the connection string for the RPS node, in the **Connection String** text box. A connection string for an SQL Server connection contains ODBC DSN information to the RDBMS database; for example:

   ```
   DSN=SQLServer2005;Database=Rps
   ```

   Note For SQL Server, the **Database=*relational-database-name*** value must be specified in the Configure RPS Node dialog only if your ODBC Data Source Name (DSN) does not specify a database. Other DSNs may require other options.

   c. If you want to specify a different database type for the node, select the type that you require in the **Database Type** combo box.

   If you select a database type that differs from that in the RPS mapping and the database already exists with a different type, an alter script is generated and run to modify the appropriate columns, as required

   d. Select the **Alternative Action** option from the **Per Statement Exception Policy – Create** combo box list if you want an alternative update action performed when an RDBMS error occurs on a create row.

   Select the **Halt** option if you want the **Datapump** application to halt when the RDBMS error occurs on a create row.

   e. Select the **Alternative Action** option from the **Per Statement Exception Policy – Delete** combo box list if you want an alternative ignore action performed when an RDBMS error occurs on a delete row.

   Select the **Halt** option if you want the **Datapump** application to halt when the RDBMS error occurs on a delete row.

   f. Select the **Alternative Action** option from the **Per Statement Exception Policy – Update** combo box list if you want an alternative create action performed when an RDBMS error occurs on an update row.

   Select the **Halt** option if you want the **Datapump** application to halt when the RDBMS error occurs on an update row.

   g. Select the **RPS Table** option from the **Logging Options** combo box list if you want any RDBMS errors that occur logged in the RPS table.

   Select the **Default** option if you want RDBMS errors output only to the **jommsg.log** file.
4. If you want to specify your own non-GUI **Datapump** application, select that schema and application in the corresponding **Schema** and **Application** combo boxes in the Data Pump Application group box.

The default value (**<default>**) indicates the RootSchema **JadeRpsDataPump** application is run on the RPS node from the schema in which the RPS mapping is defined. For details about specifying your own data pump application, see "RPS Datapump Application", earlier in this chapter.

5. Check the **Auto Start Datapump** check box in the Data Pump Application group box if you want to automatically start the **Datapump** application when the RPS node is started. By default, this check box is unchecked, indicating that the **Datapump** application must be started from the RPS Manager. (See also "Restarting the Datapump Application", earlier in this chapter.)

6. In the Data Pump Restart Options group box:

   a. Check the **Auto Execute Drop/Reload** check box if you want to restart the **Datapump** application automatically start the **Datapump** application after a reorganization to run the **Drop/Reload** alter script to rename and recreate any tables (as required) and then extract and load the data for these tables.

      By default, this check box is unchecked, indicating that the **Drop/Reload** scripts are not automatically executed and it is the responsibility of your RDBMS administrator to apply the required changes before restarting the **Datapump** application.

   b. If you checked the **Auto Execute Drop/Reload** check box in the previous step, check the **Bulk Load Historical Tables** check box if you want to load data for historical tables when reloading data into the RDBMS. All automatic loads (when setting up the RDBMS and reorganization reloads) then include data for historical tables. See also "Alter Scripts", earlier in this chapter.

      By default, the **Bulk Load Historical Tables** check box is unchecked.

   c. If you want to initiate multiple data extract worker processes on automatic extracts, specify the required number in the **# of Extract Workers** text box.

   d. In the **Alter Script Path** text box, specify the directory on the JADE RPS server node to be used for update scripts generated during reorganizations.

      The alter script path defaults to the JADE database path. This path must be set if you checked the **Auto Start Datapump** check box in step 5 of this instruction or you selected the **Create Tables** command from the RPS menu on an RPS secondary system.

      If you are unsure of your file directory, click the adjacent **Browse** button to access the common Browse for Directory selection dialog that enables you to select the auto script path.

   e. If you do not want to use the **sqlcmd** utility to execute SQL scripts, including alter scripts and server bulk load scripts, uncheck the **Use Sqlcmd** check box. By default, the **Use Sqlcmd** check box is checked.

      The **sqlcmd** utility must be installed on the machine that hosts the RPS node. If it is not installed for any reason, we recommend that you obtain the standalone installation package for the **sqlcmd** utility from the Microsoft download site and use that to install **sqlcmd**. If you do not enter a value in this text box, the script is executed using the ODBC interface.

      If you use **sqlcmd** to execute SQL scripts, including alter scripts and server bulk load scripts, the output of the script is saved in a ***script-name*.log** file in the directory in which the script is found. The output includes the echoed input and any information or error messages output. If the script fails, an invalid error result is returned and the error is reported to you if you are using the RPS Manager and it is recorded in the **jommsg.log** file.

      The advantages of using the **sqlcmd** option are as follows.

      – Error results, which are lost when using the ODBC interface, are correctly reported.

- The error information from SQL Server is saved in the log file.

f. Specify the name of the SQL server instance in the **Server Name** text box.

7. If you are using auto extracts, set the values in the Load Execution Options group box.
   - For server execution (the default):
     i. The RPS node and the RDBMS server must have shared access to a disk directory, as RPS uses the default BULK INSERT process to load data into the RDB for SQL Server execution.

        The extract files are written to the alter script path directory from the RPS node. The load function is started from the RPS node but executes (reads the files and loads the data into RDB) on the RDBMS server.

     ii. In the **RDB Path** text box, specify the absolute path of the alter script path directory to which the data is extracted from the relational database server perspective.

        The load script file name defaults to **bulkInsert.sql**.
   - For client execution, if the RPS node is executing on a different machine to the RDB server and sharing disk is not desired, select the **Client/Server** option button in the Load Execution Location group box.

     RPS uses the Bulk Copy Program (BCP) so that the load can run from directories on the RPS node. The extract files are written to the **Alter Script Path** directory (specified in step 5 of this instruction) from the RPS node, the load function is started and executed on the RPS node, the files are read on the RPS node, and the data is sent to the RDBMS using a client/server connection (which runs BCP in the RPS node for SQL Server).

     Notes Using client execution is slower than the server execution.

     If the connection to the RDBMS node requires a user name and password, the password must be included in the BCP command file as a parameter to the BCP command.

     The load command file name defaults to **bulkinsertbcp.cmd**. The output of the BCP command is piped to the **bcpoutput.log** file in the directory where the BCP command file is located.
   - If you want to specify the code page to be used for loading extracted data, select the required code page in the **CodePage** combo box.
8. If you want to reset the default values in controls in the Override Mapping Options group box to the values specified in the RPS mapping, click the **Defaults** button.
9. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

Resetting the Relational Database Identifier

In some situations, you may be required to reset the relational database identifier in the **JADE_CONTROL_INFO** table (for example, after establishing an RPS data store from the primary when no backup of the RPS data store is available).

» To reset the relational database identifier on a secondary RPS system

- Select the **Reset Relational Database ID** command from the File menu.

The connection to the RDBMS database is then established.

Checking the Consistency of the JADE and RDBMS Databases

Use the **Consistency check** command from the File menu to check the consistency definition between the JADE database and the target RDBMS database.

To check the data store consistency on a secondary RPS system

- Select the **Consistency check** command from the File menu.

The connection to the RDBMS database is then established.

The result of the consistency check is displayed in the Jade RPS Manager window.

Executing an SQL Script

Use the **Execute SQL script** command from the File menu to execute an SQL script.

To execute an SQL script on a secondary RPS system

1. Select the **Execute SQL script** command from the File menu. The common File dialog is then displayed, with the directory defaulting to the working directory specified in the Configure RPS Node dialog.
2. Select the SQL script that you want to execute and then click the **Open** button. Alternatively, click the **Cancel** button to abandon your selections.

The connection to the RDBMS database is then established.

The execute process then executes the data script file on the target RDBMS database. The Jade RPS Manager window displays the progress of the execute process and its completion (that is, whether the script executed without error). No data is returned to JADE by the script.

Printing Text Displayed in the Jade RPS Manager Window

Use the **Print** command from the File menu to print the contents of your Jade RPS Manager window.

To print the current contents of the window

1. Select the **Print** command from the File menu. The common Print dialog is then displayed, to enable you to specify your print options; for example, the printer name and the number of copies.
2. Make the selections that you require.
3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

The current contents of the Jade RPS Manager window are then output to the specified printer.

Selecting the Jade RPS Manager Window Display Font

To specify the font used to display text in the Jade RPS Manager window

- Select the **Font** command from the File menu if you want to change the default font from MS Sans Serif regular 8 points.

The common Font dialog is then displayed, to enable you to make your font selections.

When you have made your font selections, focus is then returned to the Jade RPS Manager window.

Clearing Text from the Jade RPS Manager Window

» To clear the running display from the Jade RPS Manager window

- Click the **Clear Display** command on the File menu.

When you run an RPS Manager application operation, a running display is produced in the Jade RPS Manager window. This display provides a progress report while the operation is running.

At the end of the operation, it provides a completion report highlighting the success or failure of the operation.

Exiting from the RPS Manager Application

Use the **Exit** command from the File menu to exit from the Jade RPS Manager window.

» To exit from the RPS Manager application

- Select the **Exit** command from the File menu.

Using the RPS Menu

Use the RPS menu commands to perform one of the actions listed in the following table.

Command	For details, see…	Applies to…
Create RPS Database	Creating the RPS Database	Primary only
Start Datapump	Starting the Data Pump	RPS node only
Stop Datapump	Stopping the Data Pump	RPS node only
Setup RDBMS	Setting Up RDBMS	RPS node only
Create Tables	Creating Tables	RPS node only
Load Data	Extracting and Loading Data	RPS node only
Backup RPS Database	Backing Up the RPS Database	RPS node only

Actions that do not apply to the current database role are disabled.

Creating the RPS Database

The **Create RPS Database** command in the RPS menu on the primary server provides submenu items that enable you to select the **Full**, **Mapped Extent**, and **Working Set** storage modes for the initial RPS data store being created. Select the:

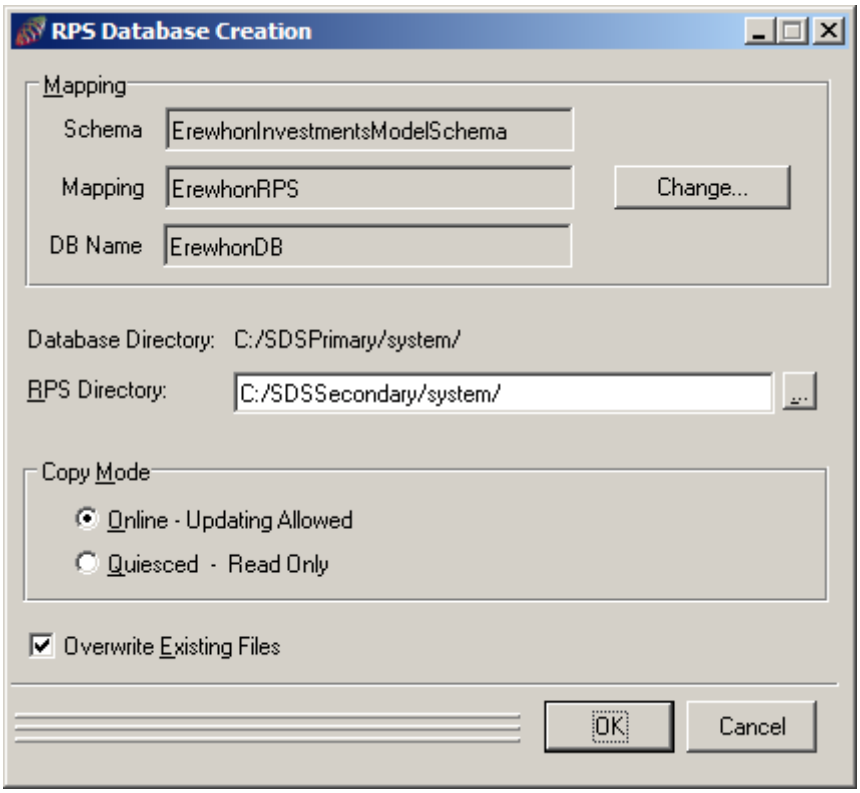
- **Full** value if you want to replicate all persistent system and user schema files in the RPS data store. As this database is structurally equivalent to an SDS secondary database, it can be used as an emergency database backup.
- **Mapped Extent** value if you want to replicate only persistent system and user schema database files that contain classes included in the RPS mapping. When a JADE database file is replicated, the entire file is replicated and not just the objects in the mapped extent.
- **Working Set** value if you want to replicate only a working set of persistent objects in one or more files that are not related to the RPS mapping files defined in the production database.

Select the directory in which the data is stored.

For details about database storage modes and the benefits of each mode, see "RPS Data Store", earlier in this chapter.

» To create the data store for an RPS node on a primary server

1. Select the **Create RPS Database** command from the RPS menu.
2. Select the required storage mode from the submenu that is then displayed. The RPS Database Creation dialog, shown in the following diagram, is then displayed.



3. If you want to change the schema, RPS mapping, or database name, click the **Change** button. The Select RPS Mapping dialog is then displayed, so that you can select the RPS schema, mapping, and database name that you require.

   For details, see "Selecting Your RPS Mapping", earlier in this chapter.

4. In the **RPS Directory** text box, specify the path of the directory in which the data store is to be created. You must specify a directory that is valid on the server. If it does not exist, it is created.

   Alternatively, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Browse for Folder dialog that enables the selection of the RPS data store directory in which the data files are created.

5. In the Copy Mode group box, select the **Quiesced – Read Only** option button if you want a quiesced read-only data store creation operation. When you select this option, the database is placed in a quiescent read-only state while the data is copied into the RPS data store. For details, see "Online Quiesced Backup", in Chapter 3 of the *JADE Database Administration Guide*.

By default, the **Online - Updating Allowed** option is selected, indicating that updates are allowed while the data is copied. For details, see "Online Full Backup", in Chapter 3 of the *JADE Database Administration Guide*.

6. Check the **Overwrite Existing Files** check box to allow any existing files in the specified destination RPS data store directory to be overwritten as the data is copied from the JADE database if the RPS mapping permitted this.

   By default, existing files are not overwritten.

7. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

When the data store creation completes successfully, the RPS Database Create Completed Successfully message box is then displayed advising you of this and the location of the log file that contains details of the operation. Click the **OK** button to return focus to the Jade RPS Manager window.

Creating an RPS Database from a Backup of the Primary

Alternatively, you can create an RPS node from a backup of the primary. You can use the following steps as an alternative to the Create RPS Database function initiated from the primary RPS Manager.

1. Restore a recent online backup to the RPS database location.
2. Start up the RPS database server.
3. Start up the RPS Manager application.

   The Select RPS Mapping dialog is then displayed. Select the schema from the tree view and the RPS mapping from the list box.

4. The Configure RPS Node dialog is then displayed, which displays the settings of parameters configured in the JADE initialization file.

   Check and alter parameters, if required (for example, directory locations and options), and then click the **OK** button on the dialog.

5. Set up the RDBMS from the RPS Manager, as required.

Starting the Data Pump

The **Datapump** application executes in the RPS node, by connecting to the target RDBMS database and applying transactions to it.

» To start the Datapump application on an RPS secondary system

- Select the **Start Datapump** command from the RPS menu. (This command is disabled if the **Datapump** application is currently active.)

The connection to the RDBMS database is then established.

For details about the **Datapump** application, including automatically starting it as a server application when you run the RPS node, see "RPS Datapump Application", earlier in this chapter.

Stopping the Data Pump

To stop the Datapump application on an RPS secondary system

- Select the **Stop Datapump** command from the RPS menu.

  This command is disabled if the **Datapump** application is currently not running.

The connection to the target RDBMS database is then closed and the **Datapump** application terminates.

Tracking is disabled in the Synchronized Database Service (SDS) database.

Setting Up RDBMS

To set up the RDBMS database

- Select the **Setup RDBMS** command from the RPS menu on an RPS secondary system.

The following actions then take place.

1. Table creation scripts are generated and executed.
2. The data is extracted and loaded into the relational database.

The Jade RPS Manager window on the RPS node displays the progress of each process and its completion (that is, whether the script executed without error) or failure (for example, the target relational database failed to open).

When you use the **Setup RDBMS** command from the RPS menu, you cannot tailor values relating to each action as is the case when you select the corresponding commands from the File menu or the Extracts menu in the displayed dialog in which you specify your preferences. (For details, see "Generating Table Creation SQL Scripts", "Executing an SQL Script", and "Extracting Data and Loading it into the Relational Database", elsewhere in this chapter.)

Creating Tables

To create RDBMS tables

- Select the **Create Tables** command from the RPS menu on an RPS secondary system.

The table creation scripts are then generated, by using the default values.

You can customize the generate table creation SQL scripts by specifying the appropriate values on the Generate Creation Scripts dialog. (For details about specifying your table creation scripts generation preferences, see "Generating Table Creation SQL Scripts", later in this chapter.)

Extracting and Loading Data

To load data into the RDBMS database

- Select the **Load Data** command from the RPS menu on an RPS secondary system.

The data is then extracted from the RPS node and loaded into the relational database.

You can customize the extraction of data from the RPS node and load it into the relational database by specifying the appropriate values on the Data Extract and Load dialog. (For details about specifying your data extract and load preferences, see "Extracting Data and Loading it into the Relational Database", later in this chapter.)

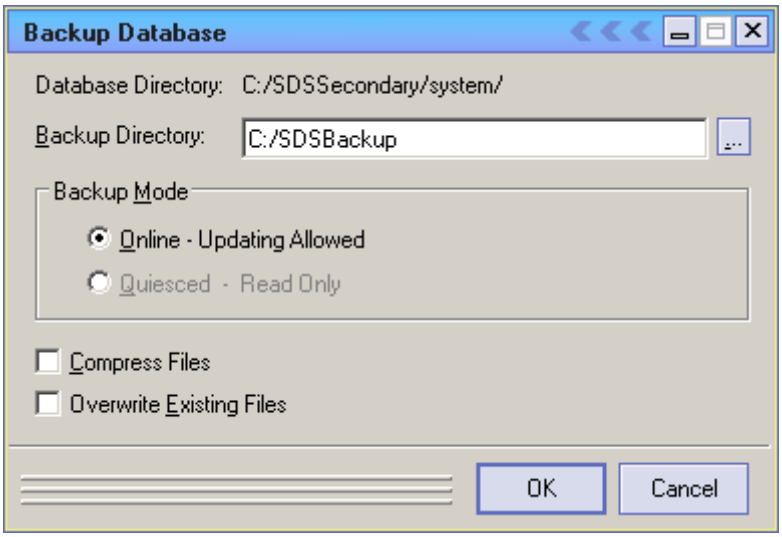
Backing Up the RPS Database

In the RPS Manager application, you can backup your RPS data store without shutting down the server node.

To backup your RPS database on an RPS secondary system

1. Select the **Backup RPS Database** command from the RPS menu.

   The Backup Database dialog, shown in the following diagram, is then displayed.



   As read-only backups cannot be performed on an RPS node, the **Quiesced – Read Only** option button is disabled.

2. In the **Backup Directory** text box, specify the path of the directory to which your database files are to be backed up.

   You must specify a directory that is valid on the server. If it does not exist, it is created. Alternatively, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Browse for Folder dialog that enables the selection of the backup directory in which the backed up database files will be located.

3. Check the **Compress Files** check box, if database files are to be compressed as they are backed up. The initial check box default selection is that database files are not compressed.

4. Check the **Overwrite Existing Files** check box to allow any existing files in the specified destination backup directory to be overwritten as the database is backed up. By default, existing files are not overwritten.

5. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

When the backup is complete, the Backup Complete dialog is then displayed. Click the **OK** button to return focus to the Jade RPS Manager window.

Using the Extracts Menu

Use the Extracts menu commands to perform one of the actions listed in the following table.

Command	For details, see…	Applies to…
Extract	Extracting Data from the JADE Database	Primary and RPS node
Extract & Load	Extracting Data and Loading it into the Relational Database	RPS node only
Load	Loading Extracted Data into the Relational Database	RPS node only
Generate Table Creation Scripts	Generating Table Creation SQL Scripts	Primary and RPS node
Generate Table Alter Scripts	Generating Table Alter SQL Scripts	Primary node

Actions that do not apply to the current database role are disabled.

Extracting Data from the JADE Database

When the RPS node data store is configured or created with the **Full** or **Mapping Extent** persistent database modes, you can extract the data on the RPS node before loading it into the target RDBMS database. If you created the database in the **Working Set** mode, you must extract the data from the primary database. (For details about selecting the database mode, see "Creating the RPS Database", earlier in this chapter.)

The extract process creates the **bulkInsert.sql** file (containing the Data Definition Language (DDL) for loading extracted data) and the data files.

If the extract is done via the RPS Manager on the SDS node, the user-defined **Datapump** application must be specified in the **DataPumpApplication**=*schema-name*,*application-name* or **DataPumpApplication=<default>** value in the [JadeRps] section of the JADE initialization file. Running an RPS extract on an SDS node causes tracking to be stopped during the extract process.

When extracting data, you can override the names of files created by the extract process, including the names of the script file itself, and you can split a data file into two or more partitions, to support the parallel loading of large tables.

The **ExtractBufferSize** parameter in the [JadeRps] section of the JADE initialization file on the RPS node specifies the buffer size that is allocated for each concurrent file being written when extracting RPS files. This parameter defaults to **1M**.

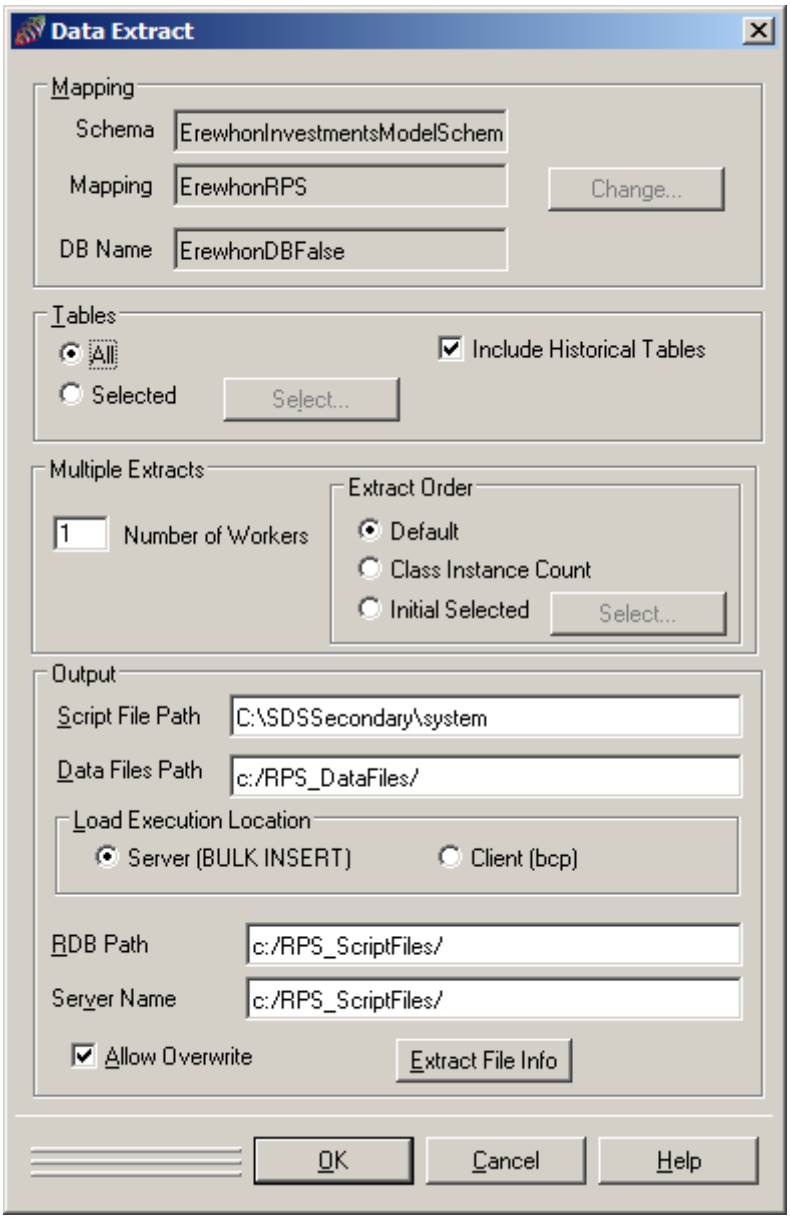
For details about extracting data or starting the extract process programmatically, see the extract data-related methods in "Calling the RPS Manager from a Web Service Consumer", later in this chapter, or the **RelationalView** class **extractData** and **extractDataAll** methods in Volume 2 or the **JadeDatabaseAdmin** class **rpsExtractData** and **rpsExtractDataAll** methods in Volume 1 of the *JADE Encyclopaedia of Classes*.

» To extract data from your persistent data store

1. Select the **Extract** command from the Extracts menu.

   The **Extract** command is disabled on an RPS node when the RPS **Datapump** application is running.

The Data Extract dialog, shown in the following diagram, is displayed.



On a primary, the **Change** button in the Mapping group box is enabled so that you can select another schema, RPS mapping, and database name defined in that schema or another RPS mapping in the displayed schema, if required.

If you require a user-defined **Datapump** application, the data must be extracted on the RPS node.

2. If you want to include data from historical tables, check the **Include Historical Tables** check box in the Tables group box. By default, the **Include Historical Tables** check box is unchecked.

3. If you want to extract selected tables only, select the **Selected** option button in the Tables group box and click the **Select** button that is then enabled.

   The RPS Table Selection dialog is then displayed. For details, see "Selecting Tables", later in this chapter.

Note If the **Include Historical Tables** check box is not checked, historical table names are disabled and cannot be selected for inclusion.

4. If you want to initiate multiple data extract worker processes, perform the following actions in the Multiple Extracts group box.

   a. In the **Number of Workers** text box, specify the number of multiple worker processes you require.

   b. The worker allocation strategy for multiple worker processes is as follows.

      - Default value
      - By class extent size (that is, the **Class Instance Count** option)
      - Selected tables first (that is, the **Initial Selected** option)

      When you select the **Initial Selected** option button, you can click the **Select** button to display the RPS Table Selection dialog, which enables you to select the tables that you want extracted first and the order in which the selected files are extracted. You can select as many tables as you require. Tables that you do not select are processed in default order. For details, see "Selecting Tables", later in this chapter.

5. In the Output group box, perform the following actions.

   a. In the **Script File Path** text box, specify the path of the directory to which the load script is to be created.

      The directory must be valid on the server. If it does not exist, it is created.

   b. In the **Data Files Path** text box, specify the path of the directory to which the generated data files are to be extracted. The directory must be valid on the server. If it does not exist, it is created.

   c. In the Load Execution Location group box, select the **Client (bcp)** option button if the RPS node is executing on a different machine to the RDB server and sharing disk is not feasible.

      The **Server (BULK INSERT)** option is selected by default. For more details about load execution locations, see step 10 under "Configuring your RPS Node", earlier in this chapter.

      Notes Using client execution is slower than the server execution.

      If the connection to the RDBMS node requires a user name and password, the password must be included in the BCP command file as a parameter to the BCP command.

   d. For **Server** load execution, in the **RDB Path** text box, specify the absolute path of the directory to which the data is extracted from the relational database server perspective. The load script file name defaults to **bulkInsert.sql**. The **bulkInsert.sql** script includes the RDB path on which the data files are located on the RDBMS server (that is, where the script file is executed). The **Data Files Path** and **RDB Path** values are the same if the RDBMS server node is the same as the JADE RPS node.

   e. For **Server** load execution if you want to use the SQL Server **sqlcmd** utility to execute the script, in the **Server Name** text box, enter the name of the SQL server to which to connect. If you do not enter a value in this text box, the script is executed using the ODBC interface.

      For **Client** load execution, in the **Server Name** text box, enter the name of the RDB server for connection. The load command file name defaults to **bulkinsertbcp.cmd**. The output of the BCP command is piped to the **bcpoutput.log** file in the directory where the BCP command file is located.

Note The **sqlcmd** utility is normally installed as part of SQL Server. If the utility is not available, do not enter a value in this text box.

If **sqlcmd** is used, the output of the script is saved in a ***script-name*.log** file in the directory in which the script is found. The output includes the echoed input and any information or error messages output.

If the script fails, an invalid error result is returned and the error is reported to you if you are using the RPS Manager and it is recorded in the **jommsg.log** file.

The advantages of using the **sqlcmd** option are as follows.

- Error results, which are lost when using the ODBC interface, are correctly reported.
- The error information from SQL Server is saved in the log file.

f. If you want existing bulk insert extract files overwritten, check the **Allow Overwrite** check box.

As extract files are not overwritten by default, the extract process fails if files exist in the output directory.

g. Click the **Extract File Info** button if you want to:

- Override the names of files created by the extract process, including the names of the script file itself.
- Split a data file into two or more partitions, to support the parallel loading of large tables.

For details, see "Maintaining Extract File Names", later in this chapter.

6. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

The Jade RPS Manager window displays the progress of the extract process and its completion.

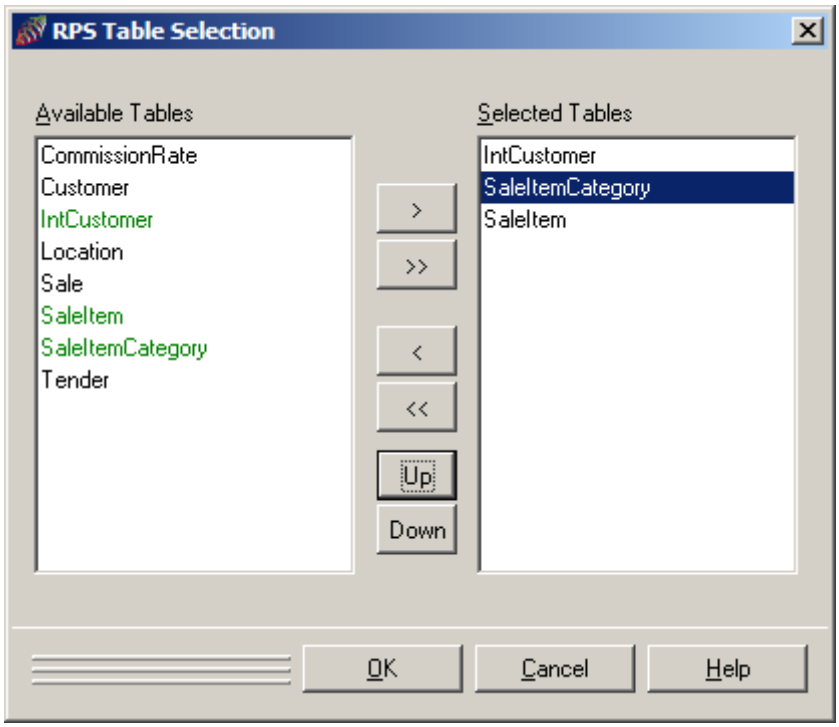
The extract script, bulk load definition files, and bulk load data files for the selected RPS mapping are then created in the specified directory.

By default, the file is extracted to the extract directory or the script file path, if specified, or to the script path on the RPS node specified in the **Alter Script Path** text box on the Configure RPS Node dialog.

Selecting Tables

When you click the **Select** button in the Multiple Extracts group box (for the initial selection order) or the Tables group box of the Data Extract dialog, the RPS Table Selection dialog is then displayed. For details about extracting RPS mapping data, see "Extracting Data from the JADE Database", earlier in this chapter.

The following diagram shows an example of the RPS Table Selection dialog.



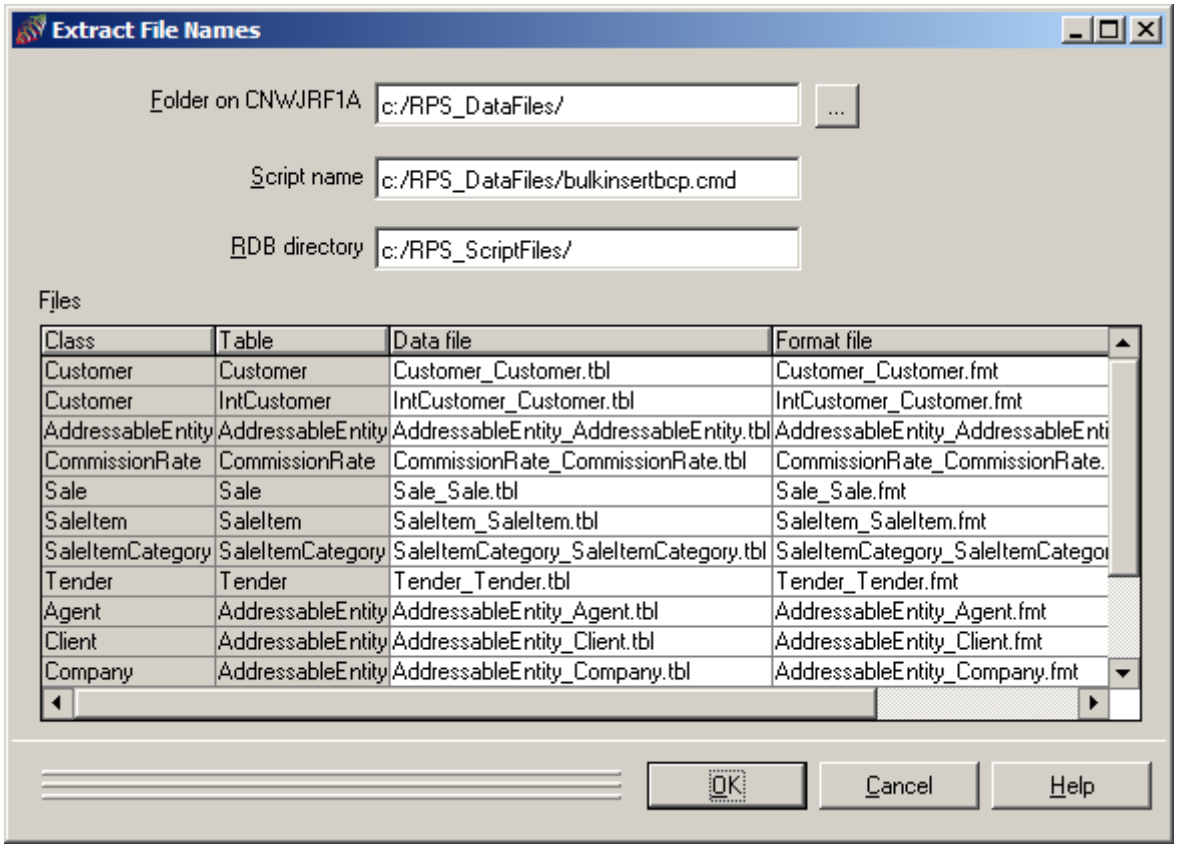
In the **Available Table**s list box, green tables indicate those already selected, gray tables are historical tables that cannot be selected (if you are not extracting historical tables), and black tables are those that are available for selection.

1. To select a table, perform one of the following actions.
   - Select one or more of the tables listed in the **Available Tables** list box (by using the SHIFT or the CTRL key to make multiple selections), and then click the single right-arrow (**>**) button.

     Only the selected table or tables are moved to the **Selected Tables** list box.
   - To select all tables in the **Available Tables** list box, click the double right-arrow (**>>**) button. All tables are then moved to the **Selected Tables** list box.
2. To remove a table from the **Selected Tables** list box, perform one of the following actions.
   - Select one or more of the tables in the **Selected Tables** list box (by using the SHIFT or the CTRL key to make multiple selections) and then click the single left-arrow (**<**) button.
   - To remove all tables in the **Selected Tables** list box, click the double left-arrow (**<<**) button. All tables are then moved to the **Available Tables** list box.
3. To move a selected table up or down the selected order (for initial selected extract order only), select one or more of the tables in the **Selected Tables** list box and then click the **Up** button to move the table up one place in the selected tables order or the **Down** button to move the table down one place.
4. Click the **OK** button when you have selected all tables that you want to include in the data extraction. Alternatively, click the **Cancel** button to abandon your selections. The Data Extract dialog is then displayed.

Maintaining Extract File Names

If you clicked the **Extract File Info** button on the Data Extract or Data Extract and Load dialog (that is, you want to override the names of the extract script file or files created by the extract process or to split a data file into two or more partitions), the Extract File Names dialog shown in the following diagram is then displayed.



» To maintain extract file names

1. In the **Folder** text box, specify the folder to which the table and format files are extracted.

   The directory must be valid on the server. If it does not exist, it is created. Alternatively, click the adjacent browse button (indicated by the … points of ellipsis symbol) to display the common Browse for Directory selection dialog that enables you to select the server and directory (folder) to which your extracted files are output.

2. If you want to change the name of the bulk insert script file (for example, a script of the default name exists in the output location and you do not want to overwrite it), specify the required name in the **Script name** text box.

3. In the **RDB Directory** text box, change the path location to which data is extracted, if required. The RDB directory is the same as the **Folder** directory if the JADE RPS node is the same as the RDBMS server node.

   The absolute path of the directory is the corresponding RPS configuration value. For details, see "Configuring your RPS Node", earlier in this chapter.

4. In the **Data file** or **Format file** column of the **Files** table, modify the file prefix of each data or format file that you want to change.

5. If you want to load large data tables into the target RDBMS database in parallel, use the **# partitions** column to specify the number of partitions you want to create for each split data table, to meet your requirements.

   You can split each data table into a maximum of 99 partitions. When you click the **OK** button, the list of data files is checked for duplicate names.

   If the number of partitions is greater than **1**, the specified table is interpreted as being a list of data files names of the format ***file-name*_01.*file-extension***, ***file-name*_02.*file-extension***, and so on up to ***file-name*_*nn*.*file-extension***, where the ***nn*** value is the specified number of partitions. For example, if you specify your data file name as **Sale.tbl** that has three partitions, the files created by the extract are **Sale_01.tbl**, **Sale_02.tbl**, and **Sale_03.tbl**.

6. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

The Data Extract or Data Extract and Load dialog from which you accessed the Extract File Names dialog is then displayed.

Extracting Data and Loading it into the Relational Database

When the RPS node secondary data store is configured or created with the **Full** or **Mapping Extent** persistent database modes, you can extract the data on the RPS node and then load it into the target RDBMS database. For details and benefits about database modes, see "Configuring your RPS Node", earlier in this chapter.

The extract process creates the data and format script files, which the load process then loads into the target RDBMS database.

When extracting data, you can override the names of files created by the extract process, including the names of the script file itself, and you can split a data file into two or more partitions, to support the parallel loading of large tables.

» To extract data from your persistent data store and load it into the target relational database

- Select the **Extract & Load** command from the Extracts menu. (The **Extract & Load** command is disabled on an RPS node when the RPS **Datapump** application is running.)

The Data Extract and Load dialog is then displayed. The controls on this dialog are the same as those on the Data Extract dialog. The **Extract & Load** command automatically runs the script to load the data extracted from your persistent data store into the target relational database.

For details about using the Data Extract and Load dialog, see "Extracting Data from the JADE Database", earlier in this chapter.

When you click the **OK** button on the Data Extract and Load dialog, the **Datapump** application is started and the data is extracted to the specified files.

After the extract completes, the script is run to load the data.

The Jade RPS Manager window displays the progress of the extract and load process and its completion.

Loading Extracted Data into the Relational Database

When you have extracted data to the data store, the RPS node secondary enables you to load this data into the target RDBMS database.

» To load data from the data store into the target relational database

1. Select the **Load** command from the Extracts menu. The common File dialog is then displayed, with the directory defaulting to the script directory.

2. Select the SQL script that you want to run to load data into the target RDBMS database and then click the **Open** button. Alternatively, click the **Cancel** button to abandon your selections.

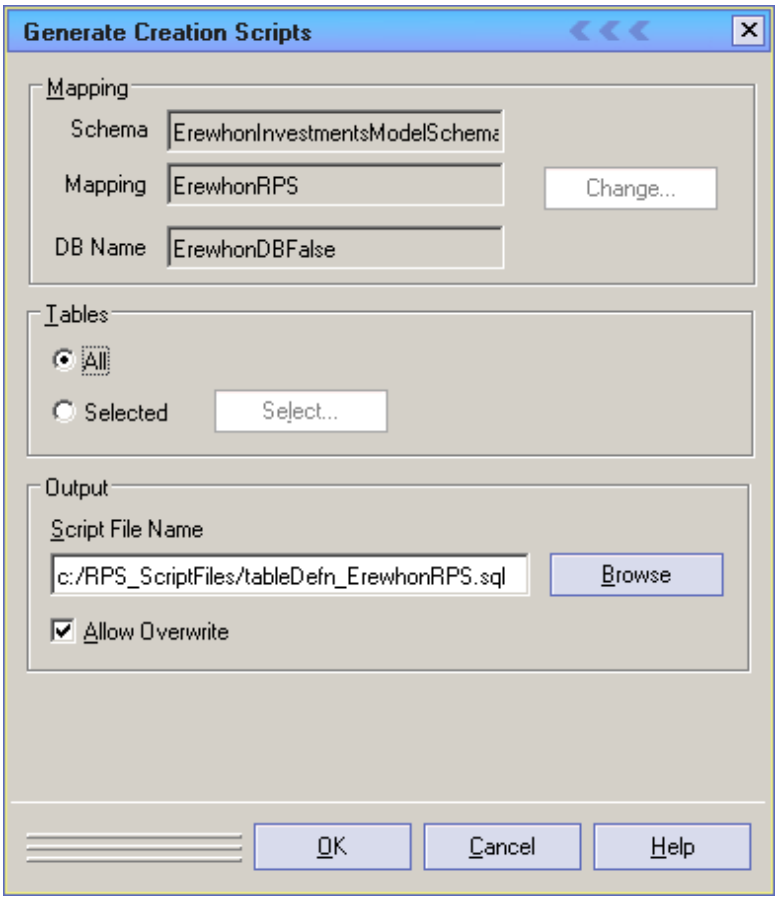
The load process then runs the script on the target RDBMS database. The Jade RPS Manager window displays the progress of the extract process and its completion.

Generating Table Creation SQL Scripts

From the primary or RPS node secondary, generate the script file that creates tables and stored procedures in the target relational database that are required for your RPS mapping. (For details about programmatically generating a script that generates the tables for an RPS mapping, see the **RelationalView** class **generateRpsTableCreationScript** method in Volume 2 of the *JADE Encyclopaedia of Classes*.)

To generate table creation scripts

1. Select the **Generate table creation scripts** command from the Extracts menu. The Generate Creation Scripts dialog, shown in the following diagram, is then displayed.



On a primary, the **Change** button in the Mapping group box is enabled so that you can select another schema, RPS mapping, and database name defined in that schema or another RPS mapping in the displayed schema, if required.

2. If you want to generate creation scripts for selected tables only, in the Tables group box, select the **Selected** option button and click the **Select** button that is then enabled. The RPS Table Selection dialog is then

displayed. For details, see "Selecting Tables", earlier in this chapter.

3. In the **Script File Name** text box, specify the full path and name of the file to which your creation script is generated. The directory must be valid on the server. If it does not exist, it is created. Alternatively, click the adjacent **Browse** button to display the common Browse for Folder dialog that enables the selection of the directory in which the creation script is generated.
4. If you want an existing creation script overwritten, check the **Allow Overwrite** check box. As creation scripts are not overwritten by default, the generation of the creation script fails if a file exists in the output directory.
5. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

The Jade RPS Manager window displays the progress of the creation script generation process and its completion. The tables and stored procedures for the selected RPS mapping are then created in the specified file (for example, **c:\jade\Rps\tableDefn_DocExample.sql**).

Generating Table Alter SQL Scripts

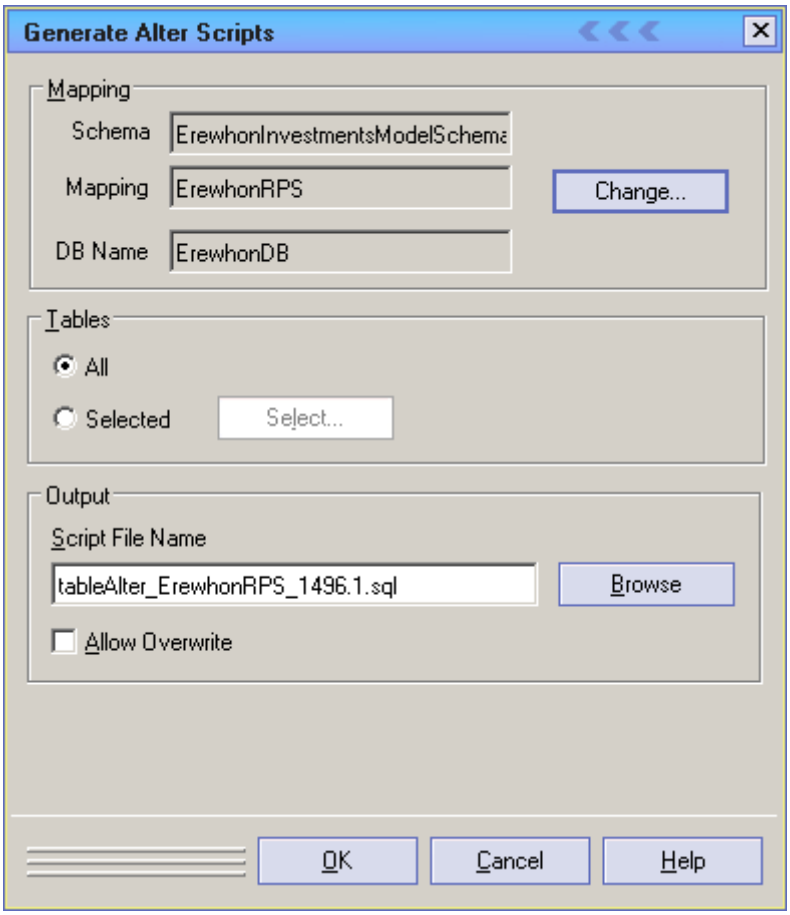
When you have altered your RPS mapping in a versioned schema, from the primary you can generate the script file that alters tables and stored procedures in the target relational database.

Note This script is automatically generated on the RPS node when the reorganization is applied on the node. This option is for information purposes only.

To generate table alter scripts

1. Select the **Generate table alter scripts** command from the Extracts menu.

   The Generate Alter Scripts dialog, shown in the following diagram, is then displayed.



   On a primary, the **Change** button in the Mapping group box is enabled so that you can select another schema, RPS mapping, and scripts for selected database name defined in that schema or another RPS mapping in the displayed schema, if required.

   Note You must have a versioned schema and mapping.

2. In the **Script File Name** text box, specify the full path and name of the file to which your alter script is generated.

   The directory must be valid on the server. If it does not exist, it is created. Alternatively, click the adjacent **Browse** button to display the common Browse for Folder dialog that enables the selection of the directory in which the alter script is generated.

3. If you want an existing alter script overwritten, check the **Allow Overwrite** check box. As alter scripts are not overwritten by default, the generation of the alter script fails if a file exists in the output directory.

4. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

If no prior version of the tables and stored procedures exist, a message box advises you that no prior version is available. If a prior version exists, the Jade RPS Manager window displays the progress of the alter script generation process and its completion.

The tables and stored procedures for the selected RPS mapping are then created in the specified file (for example, **c:\jade\Rps\tableDefn_DocumentationExample.15.sql**).

Using the Help Menu

Use the commands in the Jade RPS Manager window Help menu to access the standard Common User Access (CUA) help options.

These commands are listed in the following table.

Command	For details, see…	Applies to…
Index	Accessing the Help Index	Primary and RPS node
About	Accessing Information about the RPS Manager Application Release	Primary and RPS node

Accessing the Help Index

Use the Jade RPS Manager window Help menu **Index** command to open online help.

The online help window displays a list of topics down the left, to enable you to select the online help topic for which you require assistance.

» To access the online help, perform one of the following actions

- Select the **Index** command from the Help menu
- Press F1

Use the functions available in JADE online help to find the required topics. For details, see "JADE HTML5 Online Help" or "JADE Product Information Library in Portable Document Format", in Chapter 2 of the *JADE Development Environment User's Guide*.

Accessing Information about the RPS Manager Application Release

Use the Jade RPS Manager window Help menu **About** command to access information about the current release of the RPS Manager application; for example, the database path, the company to whom JADE is licensed, the server node operating system, and the JADE release version.

» To access the SDS Administration utility version information

- Select the **About** command from the Help menu.

The About RPS Manager dialog is then displayed.

Calling the RPS Manager from a Web Service Consumer

You can invoke a subset of the RPS Manager functionality from a Web service consumer by using the **RPSAdminWS** application provided by JADE in the **JadeMonitorSchema** schema.

Before you run the **RPSAdminWS** application provided in the **JadeMonitorSchema**, you will probably want to override the default application parameters; for example, the connection name and the port number. You can run

The **JadeWebConfigurator** application that is also provided in the **JadeMonitorSchema** enables you to create and maintain a configuration file for the **RPSAdminWS** application. For details, see Chapter 3, "Configuring Web Applications", in the *JADE Web Application Guide*.

The following example is the command line to run the **RPSAdminWS** application.

```
jade.exe path=s:\jade\system ini=c:\jade\system\jade.ini
         schema=JadeMonitorSchema app=RPSAdminWS
```

Alternatively, you can create a shortcut to run this application from the JADE installation directory in which the JADE executables were installed (which defaults to the **bin** directory).

The JADE Web Application Monitor is then displayed. For details, see Chapter 2, "Monitoring Your Web Sessions", in the *JADE Web Application Guide*. For more details, see the *Web Services* white paper on the JADE Web site at https://www.jadeworld.com/developer-center/resource-library/white-papers.

The Web service provider is **RpsAdminService**. A default WSDL, **RPSAdmin.wsdl**, is provided in the **wsdl** folder in the JADE release medium.

Note You can execute the **RelationalView** class **extractData** and **extractDataAll** methods on an SDS secondary *or* an RPS node, but not on the primary node.

If you specify a user **Datapump** application, you can execute the **JadeDatabaseAdmin** class **rpsExtractDataAll** method on an SDS or RPS node only, not on the primary node.

Running an RPS extract on an SDS node will cause tracking to be stopped during the extract.

If you extract the data via the RPS Manager on the SDS node, you must specify the user **Datapump** application and schema or the **<default>** value in the **DataPumpApplication** parameter in the [JadeRps] section of the JADE initialization file.

The following Web services methods are exposed.

- **extractData**, which provides the same functionality as the **RelationalView** class **extractData** method and which has the following signature.

```
extractData(tableName:            String;
            executionLocation:    Integer;
            scriptFilePath:       String;
            dataFilesPath:        String;
            rdbDataFilesPath:     String;
            rdbName:              String;
            loadHistoricalTables: Boolean;
            serverName:           String;
            extractWorkers:       Integer) webService;
```

  This method extracts the specified table name or all tables (that is, if the value of the **tableName** parameter is **null**), using the specified parameter values.

  The values of the **executionLocation** parameter are defined by the following **RelationalView** class constants.

  - Load_ServerExecute (**0**)
  - Load_ClientExecute (**1**)

- **extractDataAll**, which provides the same functionality as the **RelationalView** class **extractDataAll** method and which has the following signature.

```
extractDataAll(executionLocation:       Integer;
               scriptFilePath:          String;
               dataFilesPath:           String;
               rdbDataFilesPath:        String;
               rdbName:                 String;
               extractHistoricalTables: Boolean;
               serverName:              String;
               extractWorkers:          Integer;
               extractOrder:            Integer;
               extractFirst:            String;
               userDataPumpSchema:      String;
               userDataPumpApp:         String): Process;
```

  This method extracts all tables in the RPS mapping, using the specified parameter values. The **extractOrder** parameter specifies the order in which the tables are to be extracted, using the following **RelationalView** class constants.

  - **ExtractOrderDefault** (**0**), indicating that no extract order is specified.
  - **ExtractOrderClassInstances** (**1**), which outputs tables in order of the number of instances of the class, from highest to lowest.

    Note Determining the number of instances may delay the start of extraction.

  - **ExtractOrderSelectedFirst** (2), which first outputs tables specified in the **extractFirst** parameter and then uses the default order.

The **userDataPumpSchema** and **userDataPumpApp** parameters specify the user application to be used as the **Datapump** application. This application must be correctly configured as a user data pump. To use the default JADE **Datapump** application, the parameter values should be empty strings (**""**).

For details about worker allocation extraction order, see "Selecting Tables" under "Extracting Data from the JADE Database", earlier in this chapter.

When specifying values for the **extractFirst** parameter, table names are delimited by semicolons.

- **extractDataUsingIniFileOptions**, which provides the same functionality as the **RelationalView** class **extractDataUsingIniFileOptions** method and which has the following signature.

```
extractDataUsingIniFileOptions(tableName: String) webService;
```

  This method extracts the specified table name or all tables (that is, if the value of the **tableName** parameter is **null**), using the values of parameters in the [JadeRps] section of the JADE initialization file.

- **getDatabaseParameters**, which provides the same functionality as the **JadeDatabaseAdmin** class **rpsGetDatabaseParameters** method and which has the following signature.

```
getDatabaseParameters(schemaName:     String output;
                      rpsMappingName: String output;
                      storageMode:    Integer output) webService;
```

  This method returns the schema name, RPS mapping name, and the storage mode of the RPS node.

- **rpsExtractData**, which provides the same functionality as the **JadeDatabaseAdmin** class **rpsExtractData** method and which has the following signature.

```
rpsExtractData(tableName:            String;
               executionLocation:    Integer;
               scriptFilePath:       String;
               dataFilesPath:        String;
               rdbDataFilesPath:     String;
               rdbName:              String;
               loadHistoricalTables: Boolean;
               serverName:           String;
               extractWorkers:       Integer) webService;
```

This method extracts the specified table name or all tables (that is, if the value of the **tableName** parameter is **null**), using the specified parameter values. The values of the **executionLocation** parameter are defined by the following **RelationalView** class constants.

  - Load_ServerExecute (**0**)
  - Load_ClientExecute (**1**)

- **rpsExtractDataAll**, which provides the same functionality as the **JadeDatabaseAdmin** class **rpsExtractDataAll** method and which has the following signature.

```
rpsExtractDataAll(executionLocation:       Integer;
                  scriptFilePath:          String;
                  dataFilesPath:           String;
                  rdbDataFilesPath:        String;
                  rdbName:                 String;
                  extractHistoricalTables: Boolean;
                  serverName:              String;
                  extractWorkers:          Integer;
                  extractOrder:            Integer;
                  extractFirst:            String;
                  userDataPumpSchema:      String;
                  userDataPumpApp:         String): Process;
```

This method extracts all tables in the RPS mapping, using the specified parameter values.

The **extractOrder** parameter specifies the order in which the tables are to be extracted, using the following **JadeDatabaseAdmin** class constants.

  - **ExtractOrderDefault** (**0**), indicating that no extract order is specified.
  - **ExtractOrderClassInstances** (**1**), which outputs tables in order of the number of instances of the class, from highest to lowest.

    Note Determining the number of instances may delay the start of extraction.

  - **ExtractOrderSelectedFirst** (**2**), which first outputs tables specified in the **extractFirst** parameter and then uses the default order.

For details about worker allocation extraction order, see "Selecting Tables" under "Extracting Data from the JADE Database", earlier in this chapter.

When specifying values for the **extractFirst** parameter, table names are delimited by semicolons.

- **rpsExtractDataUsingIniFileOptions**, which provides the same functionality as the **JadeDatabaseAdmin** class **rpsExtractDataUsingIniFileOptions** method and which has the following signature.

  ```
  rpsExtractDataUsingIniFileOptions(tableName: String) webService;
  ```

  This method extracts the specified table name or all tables (that is, if the value of the **tableName** parameter is **null**), using the values of parameters in the [JadeRps] section of the JADE initialization file.

- **startDataPump**, which provides the same functionality as the **JadeDatabaseAdmin** class **rpsStartDataPump** method and which has the following signature.

  ```
  startDataPump(userName: String;
                password: String) webService;
  ```

  This method starts the RPS **Datapump** application on the RPS server node for the user name and corresponding password.

- **stopDataPump**, which provides the same functionality as the **JadeDatabaseAdmin** class **rpsStopDataPump** method and which has the following signature.

  ```
  stopDataPump() webService;
  ```

  This method stops the RPS **Datapump** application on the RPS server node.

Site-Specific RPS Mapping Customization

You can tailor table definitions in an RPS mapping to specifically exclude tables or columns. This enables you to customize an RPS mapping for a specific site if your JADE applications are deployed to multiple sites with different RDBMS mapping requirements.

Tables and columns excluded from an RPS mapping are still present in the RPS Mapping schema, but a flag is set to indicate that it is currently excluded from the customized RPS Mapping.

Exclude tables and columns by using:

- The Relational Population Service wizard, documented under "Mapping Classes to Tables" and "Maintaining RPS Column Mappings", in Chapter 15 of the *JADE Development Environment User's Guide*
- Your own routines written using the functionality summarized under "Using Methods to Exclude RPS Mapping Elements", later in this chapter.

Deploying Systems with Excluded Elements

» To customize an RPS mapping

1. In the JADE development environment, use the Relational Population Service wizard to create an all-inclusive RPS mapping. (For details, see Chapter 15, "Using the Relational Population Service (RPS) Wizard", of the *JADE Development Environment User's Guide*.)
2. Deploy your schema to production.
3. Define the excluded elements, by using the Relational Population Service wizard or user-written code.

4. Create the JADE command file to save the excluded tables and columns from the RPS mapping for re-application, by using the:

   - **JadeRpsUtility** application in the non-GUI **jadclient** executable, which enables you to run the **JadeRpsUtility** application in the **RootSchema** application to create an RPS exclusion command file for application deployment scenarios by using the following command line arguments.

     ```
     jadclient path=database-path ini=jade-initialization-file
     schema=schema-name app=JadeRpsUtility endJade action=createJcf
     rpsMapping=rps-mapping-name commandFile=jcf-command-file-name
     ```

     Run this application to create the JCF file (JADE command file) of the name specified in the **commandFile** parameter that includes all excluded table and columns for the RPS mapping specified in the **rpsMapping** parameter. If you do not specify a command file suffix (for example, **.jcf** or **.txt**), it is created with a suffix of **.jcf**. If you do not specify the output location of the file, it is created in your JADE working directory (binary file location) path.

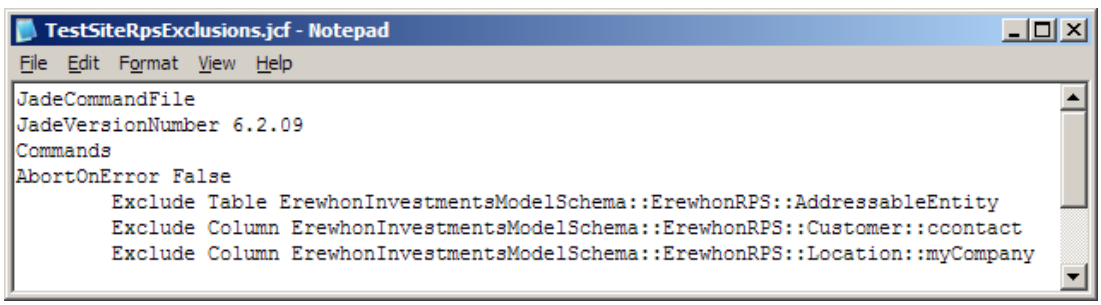
     Note The **jadclient** parameter arguments are case-sensitive.

     The following is an example of the command prompt that runs a non-GUI client application to create an RPS exclusion command file.

     ```
     jadclient path=c:\jade\system schema=RootSchema
     ini=c:\jade\system\jaderps.ini app=JadeRpsUtility endJade action=createJcf
     schema=ErewhonInvestmentsModelSchema rpsMapping=ErewhonRPS
     commandFile=c:\jade\TestSiteRpsExclusions.jcf
     ```

     For details about the **jadclient** non-GUI client application, see "Running a Non-GUI Client Application using jadclient", in Chapter 1 of the *JADE Runtime Application Guide*.

   - **Create Jcf** command in the Relational menu of the JADE development environment for the RPS mapping selected in the Relational Population Service browser. The command file is created with a suffix of **.jcf** in the location of the schema and forms definition files. The following is an example of an RPS exclusion command file.

     Note An RPS mapping that includes excluded elements retains the exclusions if it is extracted and reloaded.

   For details about using the batch JADE Schema Load utility **commandFile** parameter to add excluded tables and columns to a JADE command file, see "Loading a Schema and Forms in Batch Mode", in the *JADE Schema Load Utility User's Guide*.

5. Redeploy your all-inclusive RPS mapping.

6. Reapply the excluded elements in the JADE command file to the deployed schema, by using the batch JADE Schema Load utility **commandFile** parameter to load an existing JADE command (**.jcf**) file containing tables and columns excluded from the RPS mapping, as shown in the following example.

```
jadloadb path=c:\jadeprod\system
commandFile=d:\rps\TestSiteRpsExclusions.jcf
```

The schema and RPS mapping are versioned during the loading of the **.jcf** file and unless otherwise specified, the reorganization occurs.

Using Methods to Exclude RPS Mapping Elements

The following table summarizes the methods defined in the **RelationalView** class (for details, see Volume 2 of the JADE *Encyclopaedia of Classes*) that enable you to programmatically customize an RPS mapping and create the subsequent exclusion command file.

Method	Description
createExcludedJcfFile	Creates the JADE command file (**.jcf** file) to exclude all tables and columns currently excluded in the RPS mapping
excludeTableColumnName	Excludes the specified column in the table from the RPS mapping
excludeTableName	Excludes the table specified in the **tableName** parameter from the RPS mapping
getExcludedTableColumnNames	Returns a reference to an array of excluded column names in the specified table in the RPS mapping
getExcludedTableNames	Returns a reference to an array of excluded table names defined in the RPS mapping
getTableColumnNames	Returns a reference to an array of column names specified by table name in the table
getTableNames	Returns a reference to an array of names of tables defined in the RPS mapping

Relational Population Service (RPS) Restrictions

The JADE RPS implementation excludes the following functionality.

- User JADE applications (including read-only applications) may not execute properly on **Mapped Extent** or **Working Set** nodes, as the required data many not be present.
- Mapping data values stored in primitive arrays is not supported.
- Mapping methods for non-virtual properties are not executed when extracting property values from a JADE object for replication into relational columns.
- Modifying the SQL database being replicated from RPS in order to optimize queries is not recommended.

  This is commonly handled in business intelligence solutions by replicating into a staging database. An Extract Transform Load (ETL) process is then run to load the staging data into a database that is optimized for querying. The staging database is typically a straight replica of the production Online Transaction Processing (OLTP) database and as such, is not generally suitable for Online Analytical Processing (OLAP) purposes.

RootSchema-Defined Types and Features

The **RootSchema** provides:

- The **JadeRpsDataPumpIF** interface that provides the functionality required to filter and manipulate the output sent to the target relational database from RPS.
- Methods defined in the **Application**, **JadeDatabaseAdmin**, and **Schema** classes, providing Application Programming Interfaces (APIs) for RPS Manager functions.
- Methods that enable you to customize an RPS mapping for a specific site. For details, see "Site-Specific RPS Mapping Customization" and "Using Methods Defined in RootSchema Classes", elsewhere in this chapter.

For more details, see the following subsection or Volumes 1 and 2 of the *JADE Encyclopaedia of Classes*.

Using Methods Defined in RootSchema Classes

The following SDS administration methods provided by the **JadeDatabaseAdmin** class are not supported for RPS nodes. (An attempt to call these methods for an RPS node raises a runtime error.)

- sdsDisableReadAccess
- sdsDisableReadAccessAt
- sdsEnableReadAccess
- sdsEnableReadAccessAt
- sdsInitiateHostileTakeover
- sdsInitiateTakeover

All other documented SDS administration methods provided by the **JadeDatabaseAdmin** class are supported for administering RPS nodes. For details, see Volume 1 of the *JADE Encyclopaedia of Classes*.

The following table summarizes the RPS-related methods defined in other **RootSchema** classes and RPS-related interfaces.

Method	Class or Interface	Description
rpsDataPumpFinalize	Application	Performs any terminate function for the **Datapump** application
rpsDataPumpInitialize	Application	Registers the receiver to be notified when a nominated event occurs on callbacks required to implement incremental object replication
getRpsMappingRefs	Class	Adds RPS mappings (instances of the **RelationalView** class) in which the receiver is used
hasRpsReferences	Class	Returns **true** if the receiver is referenced by one or more RPS mappings
createRpsDatabase	JadeDatabaseAdmin	Creates an RPS database for a specified schema and RPS mapping

Method	Class or Interface	Description
getRpsMappedFiles	JadeDatabaseAdmin	Returns the name of the schema and RPS mapping of the receiver, and a reference to an array of all files in the mapping
rpsAuditSqlScriptForReplay	JadeDatabaseAdmin	Causes a special RPS callback audit record to be written to the current journal
rpsExtractData	JadeDatabaseAdmin	Extracts a specified table or all tables on an RPS node, using specified parameter values
rpsExtractDataAll	JadeDatabaseAdmin	Extracts all tables in the RPS mapping on an RPS node, using specified parameter values
rpsExtractDataUsingIniFileOptions	JadeDatabaseAdmin	Extracts a specified table or all tables on an RPS node, using values stored in the [JadeRps] section of the JADE initialization file
rpsGetDatabaseParameters	JadeDatabaseAdmin	Returns the name of the schema and RPS mapping, and the database replication mode (**Working Set**, **Mapped Extent**, or **Full**) of the RPS server node
rpsStartDataPump	JadeDatabaseAdmin	Starts the RPS **Datapump** application on the RPS server node for the user name and corresponding password, and returns a boolean value indicating the success or failure of the action (this method must be run from the RPS node)
rpsStopDataPump	JadeDatabaseAdmin	Stops the RPS **Datapump** application on the RPS server node and returns a boolean value indicating the success or failure of the action
updateCallback	JadeRpsDataPumpIF	Adds, changes, or deletes a transaction in the relational database table
updateObjectEdition	Object	Increments the edition of the specified object by one
rpsSuppressTransactionDeletes	Process	Specifies on the primary system the transactions for which deletions are *not* to be replicated to the relational database
extractData	RelationalView	Extracts a specified table or all tables on an SDS secondary or an RPS node, using specified parameter values
extractDataAll	RelationalView	Extracts all tables in the RPS mapping on an SDS secondary or an RPS node, using specified parameter values
extractDataUsingIniFileOptions	RelationalView	Extracts a specified table or all tables on an SDS secondary or an RPS node, using values stored in the [JadeRps] section of the JADE initialization file
generateRpsTableCreationScript	RelationalView	Programmatically generates a script that creates the tables for an RPS mapping

Method	Class or Interface	Description
getRpsMappedClasses	RelationalView	Adds classes that are involved in the RPS mapping to the specified set
versionRpsMapping	RelationalView	Versions the RPS mapping, if not already versioned, and returns the latest version of the RPS mapping
getAllRpsMappings	Schema	Returns a reference to an array of all RPS mappings in the receiver
getRpsMapping	Schema	Returns a reference to the specified RPS mapping in the receiver