



Database Administration Guide

VERSION 2020.0.02



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 © 2021 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	viii
Who Should Read this Guide	viii
What's Included in this Guide	viii
Related Documentation	viii
Conventions	viii
Chapter 1 Using the JADE Database Utility	10
The JADE Database Utility	10
Installing the JADE Database Utility	11
Executing the JADE Database Utility	11
Opening a JADE Database	13
Closing a Database	13
Using Message Log Files	14
Closing a Message Log File	14
Saving a Message Log File	14
Saving a Message Log File to Another Name or Path	14
Printing the Window or Viewer Contents	15
Specifying Your Print Setup Requirements	15
Accessing a Recent Log File	15
Clearing the Running Display	16
Exiting from the JADE Database Utility	16
Using Database Operational Commands	16
Using the Certify Files Command	17
Using the Compact Files Command	19
Using the Evaluate Free Space Command	20
Using the Delete Files Command	21
Using the Reindex Files Command	22
Using the Reset Timestamps Command	23
Using the Verify Checksums Command	24
Using the Set File Attributes Command	25
Using the Backup Locations Command	26
Using the Backup Database Command	28
Selecting Your File Backup Locations	30
Using the Restore Database Command	31
Restoring Your Database with Roll-Forward Recovery	31
Restoring Your Database with No Recovery	34
Using the Initiate Recovery Command	36
Using the Backup Journals Command	38
Using the Restore Journals Command	40
Using the Verify Journals Command	41
Using the Dump Journals Command	41
Producing Journaling Rates Reports	42
Using the Production Mode Command	43
Specifying Your JADE Database Utility View Options	44
Toggling the Display of the Toolbar	44
Toggling the Display of the Status Bar	44
Selecting the Display Font	44
Selecting the Printer Font	45
Viewing Log Files	45
Setting Your JADE Database Utility Preferences	46
Specifying Your JADE Database Utility Edit Options	47
Undo Command	48
Cut Command	48
Copy Command	49
Paste Command	49
Delete Command	49

Find Command	50
Find Next Command	51
Replace Command	51
Select All Command	52
Word Wrap Command	52
Specifying Your Window Display	52
Obtaining JADE Database Utility Online Help	53
Accessing the Online Help Index	53
Accessing JADE Database Utility Release Information	53
Running the JADE Database Utility in Batch Mode	53
Batch Database Utility Arguments	56
backupDir	56
backupinfoPath	56
baseDir	56
clearFilePaths	56
clearSDSRole	57
compress	57
deltaDir	57
excludeFrozenFiles	57
excludeFrozenPartitions	57
extentSize	58
fileDir	58
file-list	58
file-name	58
ini	58
initSize	58
journalDir	59
journal	59
lastJournal	59
lastSerialNumber	59
newBackupPath	59
newDatabasePath	59
noRecovery	60
nostatus	60
outputDir	60
overwrite	60
part	60
path	60
recoverTo	60
samplingInterval	61
verify	61
version	61
workers	61
Batch Database Utility Commands	61
backup	61
backupJournal	62
certify	63
changeFilePaths	63
File Partition Location Modification Example	64
clearDeltaMode	65
clearFilePath	65
clearFilePathAudited	65
clearRole	66
commandFile	66
compact	66
convertToBackup	67
delete	68
dumpJournal	68
fileAttributes	68
freeSpace	69
freezeSchemaFiles	69

generateEnvironmentIdentity	70
generateServerIdentity	70
generateUUID	70
help	70
journalingRates	71
listBackupinfoFilenames	71
listFiles	71
listPartitions	72
makeBackupinfo	72
mapFile	73
markOffline	73
markOnline	74
productionMode	74
recover	74
reindex	75
resetTS	75
restore	75
restoreFile	76
restoreJournal	77
restorePartitionFile	78
setFilePath	78
setFilePathAudited	79
setUserFileVersions	79
showInfo	80
thawSchemaFiles	80
touchDB	80
unmapFile	81
verifyChecksums	81
verifyJournal	82
Chapter 2 Using the JADE Database Administration Utility	83
Overview	83
Running the JADE Database Administration Utility	83
Command Arguments	85
action	85
path	86
ini	86
server	86
Database Actions	86
ActivateDeltaDb	86
Backup	87
DeactivateDeltaDb	88
EndSnapshot	89
FreezeSchemaFiles	89
ListDbFiles	90
StartSnapshot	90
ThawSchemaFiles	90
File or Partition Actions	91
Certify	91
FreeSpace	91
Freeze	91
Thaw	92
File Actions	92
Compact	92
ListParts	93
MakePartitioned	93
SetFilePath	94
Partition Actions	95
MarkOffline	95
MarkOnline	95

MovePartition	95
PurgePartition	96
RemovePartition	96
SetPartitionLabel	97
SetPartitionLocation	97
Chapter 3 Administering the JADE Database	98
What Is a Backup?	99
Why Are Backups Important?	99
When to Take Backups	100
Types of Failure	100
Physical Database Structures	100
Database Files	100
Journal Files	100
Database Control File	101
Database Identities	101
Database Identity	102
Environment Identity	102
Server Identity	102
Transaction Journal Files	103
Journal Switches and Journal Numbers	103
Quietpoints and Long Transactions	104
Checkpoints	105
Archived Transaction Journals	105
Database Archiving Modes	105
Automated Journal Close Actions	106
Unaudited Database File and Partition Operations	106
Backups	107
Offline Full Backup	108
Online Quiesced Backup	108
Online Full Backup	108
Verification during Backup	109
Choosing a Backup Strategy	109
The Danger in Backing Up the Online Transaction Journal	110
Backing Up Your JADE Development Environment	110
Custom Backup Support	112
Recovery Concepts and Strategies	112
Automatic Restart Recovery	113
Restoring and Recovering the Database	113
Roll-Forward Recovery Restrictions	114
Roll-Forward Recovery of a Standby Database	114
Data Corruption	114
What Causes Data Corruption?	115
Detecting Data Corruption Problems	115
Performance Considerations	116
Checking the Integrity of the Database Control File	116
Database Certification	117
Reindexing Database Files	117
Evaluating Free-Space	118
Compacting Files	118
Resetting Timestamps	119
Transient Database File Analysis	119
Chapter 4 Encrypting the JADE Database	121
JADE Database Encryption Overview	121
Pre-requisite	123
Restrictions	123
Encryption Activity Audit Trail	123
Access Check Options	124

- Default Security 124
- Downgraded (or Non-Existent) Security 124
- Strong Security 124
- Service Principal Names 125
- Running the JADE Database Encryption Utility 126
 - Encrypting a Database 127
 - Decrypting a Database 127
 - Command Arguments 128
 - action 128
 - path 129
 - ini 129
- Database Encryption Actions 129
 - ApplyPendingChanges 129
 - ClearPendingChanges 130
 - DecryptFile and DecryptFiles 130
 - DeleteStoredKey 131
 - DisableDatabaseEncryption 131
 - EnableDatabaseEncryption 132
 - EncryptFile and EncryptFiles 133
 - ExportMasterKey 135
 - ImportMasterKey 135
 - ListStatus 136
 - ListStoredKeys 137

Before You Begin

The *JADE Database Administration Guide* is intended as the main source of information when you are administering your JADE database.

Who Should Read this Guide

The main audience for the *JADE Database Administration Guide* is expected to be database administrators.

What’s Included in this Guide

The *JADE Database Administration Guide* has four chapters.

Chapter 1	Covers using the single user offline JADE Database utility
Chapter 2	Covers using the single user or multiuser online JADE Database Administration utility
Chapter 3	Covers JADE database administration
Chapter 4	Covers using the JADE Database Encryption utility

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 Developer’s Reference	Developing or maintaining JADE applications
JADE Development Environment Administration Guide	Administering the JADE development environment
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, including security
JADE Synchronized Database Service (SDS) Administration Guide	Administering JADE Synchronized Database Services (SDS), including Relational Population Services (RPS)

Conventions

The *JADE Database 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.

Convention	Description
Bold	<p>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.</p> <p>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.</p>
<i>Italic</i>	<p>Parameter values or placeholders for information that must be provided; for example, if instructed to enter <i>class-name</i>, type the actual name of the class instead of the word or words shown in italic type.</p> <p>Italic type also signals a new term. An explanation accompanies the italicized type.</p> <p>Document titles and status and error messages are also shown in italic type.</p>
Blue text	<p>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 "Choosing a Backup Strategy" cross-reference to display that topic.</p>
Bracket symbols ([])	<p>Indicate optional items.</p>
Vertical bar ()	<p>Separates alternative items.</p>
Monospaced font	<p>Syntax, code examples, and error and status message text.</p>
ALL CAPITALS	<p>Directory names, commands, and acronyms.</p>
Small font	<p>Keyboard shortcut keys.</p>

Key combinations and key sequences appear as follows.

Convention	Description
Key1+Key2	<p>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.</p>
Key1,Key2	<p>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.</p>

This chapter covers the following topics.

- [The JADE Database Utility](#)
 - [Installing the JADE Database Utility](#)
 - [Executing the JADE Database Utility](#)
 - [Opening a JADE Database](#)
 - [Closing a Database](#)
 - [Using Message Log Files](#)
 - [Printing the Window or Viewer Contents](#)
 - [Specifying Your Print Setup Requirements](#)
 - [Clearing the Running Display](#)
 - [Exiting from the JADE Database Utility](#)
 - [Using Database Operational Commands](#)
 - [Specifying Your JADE Database Utility View Options](#)
 - [Specifying Your JADE Database Utility Edit Options](#)
 - [Specifying Your Window Display](#)
 - [Obtaining JADE Database Utility Online Help](#)
- [Running the JADE Database Utility in Batch Mode](#)

The JADE Database Utility

The JADE Database utility is a single user offline interface for initiating database maintenance functions. You can:

- Backup your database and journal files offline
- Restore a database and journal files from backup
- Check the integrity and efficiency of your JADE database files, including the compact operation of selected database files
- Compact your database files to reclaim deleted object space
- Verify checksums
- Allocate the initial file size and the extent of file growth
- Initiate roll-forward recovery

Note To run the offline JADE Database utility, you must have exclusive (that is, single user) access to the database files. For details about online single user or multiuser database administration, see [Chapter 2](#).

The JADE Database utility always has two distinct views: a Database View that is used to display the progress of database operations and a Log File View that is used to display the output of a selected activity message log. Each of these views has a different top-level menu, which is determined by the selected view.

You can use the **Interrupt User** command in the JADE Monitor Users view to interrupt a database operation. This action cancels (that is, performs a user abort action of) an in-progress file certify, file compact, file reindex, file freespace evaluate, file usage analysis, make file partitioned, and move instances database operations, as well as in-progress reorganizations.

For details about the parameters that provide advanced diagnostic capabilities when performing certify and freespace evaluation operations, see "[Database Diagnostics Section \[DBUtil\]](#)", in the *JADE Initialization File Reference*.

Installing the JADE Database Utility

The JADE Database utility is installed in your JADE work directory; that is, the directory in which your JADE binary files are located (for example, `c:\jade\bin`) as part of the JADE installation process.

Executing the JADE Database Utility

Use the menu bar or toolbar icons to access the JADE Database utility commands.

» To execute the JADE Database utility, perform one of the following actions

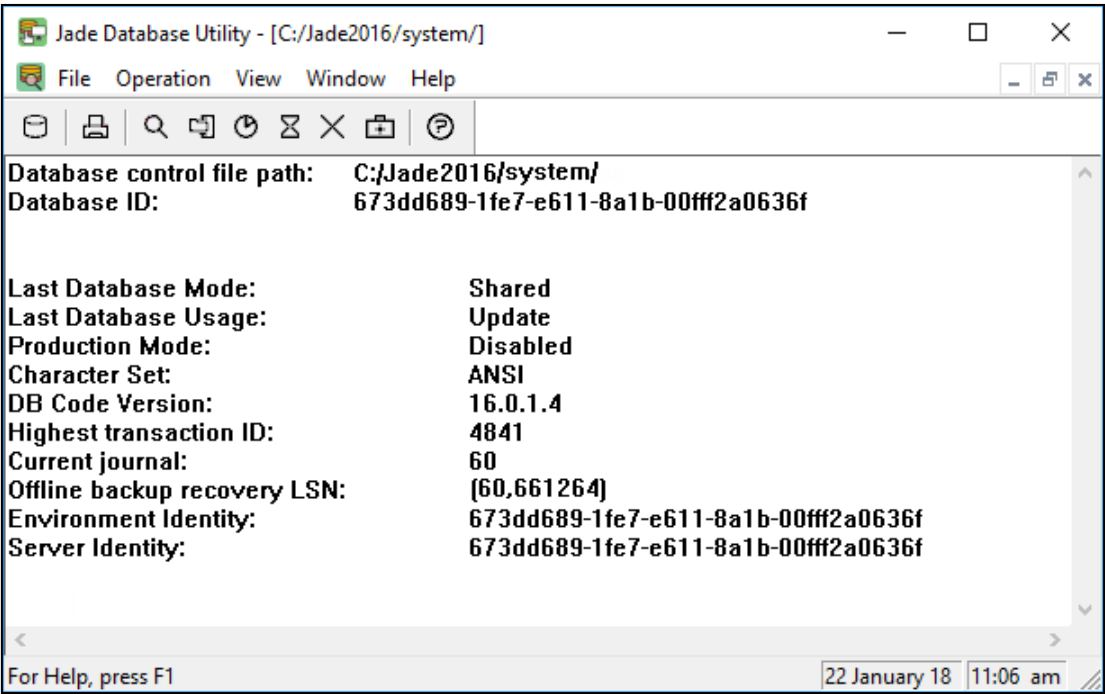
- Click the **Jade Database Utility** icon in the JADE program folder.
- In Explorer or File Manager, access the directory in which your JADE binary files are located (for example, `c:\jade\bin`) and then double-click on the `jdbutil` program file.

Operations such as specifying backup locations or restoring database files or transaction journals are performed when no database is open. You must open a database to perform other operations.

Note When you attempt to open a database and recovery is required, automatic recovery is not performed. (This enables you to defer the recovery of the database, if required.) If you want to initiate recovery, you must manually do so by selecting the **Initiate Recovery** command from the JADE Database utility Operation menu. You can then open the database.

For details, see "[Using the Initiate Recovery Command](#)" and "[Opening a JADE Database](#)", later in this chapter.

The Jade Database Utility background window, shown in the following image, is then displayed.



Note When you first invoke the JADE Database utility, you can perform only the specification of backup locations or the restoration of database files or transaction journals until you open a database. (Click the **Open Database** toolbar button to quickly access the Open Database dialog.)

Use the File menu in the Database Utility window to perform one of the administrative actions listed in the following table.

Command	For details, see...	Description
Open Database	Opening a JADE Database	Closes the current database and opens a new one
Close Database	Closing a Database	Closes the current database
Close Log File	Closing a Message Log File	Closes the currently displayed log file in the viewer
Save	Saving a Message Log File	Saves text in the editor pane of the log file viewer
Save As	Saving a Message Log File to Another Name or Path	Opens the common Save As dialog, to enable you to save the log file displayed in the viewer as a different name or location
Print	Printing the Window or Viewer Contents	Prints the window or log file viewer contents
Print Setup	Specifying Your Print Setup Requirements	Displays the Print Setup dialog, to enable you to modify your print configuration

Command	For details, see...	Description
Recent Files	Accessing a Recent Log File	Displays a list of the most-recently opened log files
Clear Display	Clearing the Running Display	Clears the running display
Exit	Exiting from the JADE Database Utility	Exits from your JADE Database utility session

Use the menu bar or toolbar icons to access the JADE Database utility commands.

Opening a JADE Database

When you first invoke the JADE Database utility, you must open a database before you can perform any operation other than specifying backup locations, or restoring database files or transaction journals if a database is not in a consistent state.

You can open a database only from the Database View.

» To open your JADE database

1. Perform one of the following actions.

Select the **Open Database** command from the File menu. (If a database is currently open, it is first closed.)

- Press Ctrl+O.
- Click the **Open Database** toolbar button.

The Open Database dialog, shown in the following image, is then displayed.



2. In the **Database Path** combo box, specify your database directory or select the directory name from the drop down list of most recently used database paths.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the database directory in which the database control file (**_control.dat**) is located.

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

Any other database that is currently open is then closed and your specified database is opened.

Closing a Database

Use the **Close Database** command from the File menu to close the current database; for example, before backing up another database.

You can close a database only from the Database View.

» To close the current database, perform one of the following actions

- Select the **Close Database** command from the File menu.
- Press Ctrl+C.

The current database is then closed and the focus returns to the JADE Database utility window.

Using Message Log Files

The File menu enables you to close, save, and print JADE Database utility message log files.

Closing a Message Log File

Use the **Close Log File** command from the File menu to close the log file that is currently displayed in the viewer.

You can close a log file only from the Log File View.

» To close the current log file

- Select the **Close Log File** command from the File menu.

The log file that is currently displayed in the viewer is then closed, and the focus returns to the JADE Database utility window.

Saving a Message Log File

Use the **Save** command from the File menu to save a log file displayed in a viewer in the JADE Database utility.

You can save a log file only from the Log File View.

» To save changes to a log file, perform one of the following actions

- Select the **Save** command from the File menu.
- Press Ctrl+S.
- Close the log file viewer.

If the file has been changed, a message dialog advises you that the file has changed, and asks if you want to save it.

Saving a Message Log File to Another Name or Path

Use the **Save As** command from the File menu to change the name of the current log file, or save it to a different location. The **Save As** command uses the common Save As file dialog, to prompt you for a file name and location to which the current log file will be saved.

You can save a log file as another name or path only from the Log File View.

» To rename a log file or save it to a new location

1. Select the **Save As** command from the File menu.

The common Save As dialog is then displayed.

2. Specify the new name of the log file, if required.

3. Optionally select the drive, directory, or folder where you want to save the log file. (The file is saved to your JADE database directory, by default.)
4. Click the **OK** button.

Printing the Window or Viewer Contents

Use the **Print** command from the File menu to print the contents of your JADE Database utility Database View or Log File View.

» To print the current contents of the current view

1. Perform one of the following actions.
 - Select the **Print** command from the File menu.
 - Press Ctrl+P.
 - Click the **Print** toolbar button.

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 Database utility window or viewer are then output to the specified printer.

Specifying Your Print Setup Requirements

Use the **Print Setup** command from the File menu to modify the printing of your JADE Database utility Database View or Log File View contents.

» To modify your print options

1. Select the **Print Setup** command from the File menu. The common Print Setup dialog is then displayed.

Note The print setup dialogs that are displayed and the options that are available are printer-dependent, and cannot be controlled from within JADE.

2. Make the advanced documentation and printer connection selections that you require.
3. Click the **OK** button. Alternatively, click the **Cancel** button to abandon your selections.

Accessing a Recent Log File

Use the **Recent Files** command from the File menu in both the Database View and the Log File View to display the most-recently opened log files, to enable you to quickly access a recent file.

Accessing a recently opened log file from the Database View changes the view to the Log File View. The Database View is restored only when all open log files are closed. (For details, see "[Closing a Message Log File](#)", earlier in this chapter.)

» To access a recent log file

1. Select the **Recent Files** command from the File menu.
- The four most-recently opened log files are then displayed in a submenu, with the latest log file being at the top of the list, and the first of the four that you opened displayed at the bottom of the list.
2. Select the log file that you want to reopen.
- The selected log file is then displayed in the Log File View of the Jade Database Utility window.

Clearing the Running Display

Use the **Clear Display** command from the File menu to clear the running display.

When you run a JADE Database utility operation, a running display is produced on your screen.

This display provides a progress report while the operation is running. At the end of the operation, it provides a completion report highlighting the number of files processed and the number of files found with errors. For a detailed status of the operation, you must view the appropriate log file. (For details, see "[Viewing Log Files](#)", later in this chapter.)

You can clear the running display only in the Database View.

» To clear the running display

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

Exiting from the JADE Database Utility

Use the **Exit** command from the File menu to exit from the JADE Database utility. Your log files are not removed when you from exit from the JADE Database utility. You should therefore delete them manually when you have finished using them.

» To exit from the JADE Database utility

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

You can exit from the JADE Database utility in the Database View or the Log File View.

Using Database Operational Commands

Use the Operation menu in the Jade Database Utility window to perform operational activities on your database. The Operation menu is displayed only in the Database View.

The JADE Database utility Operation menu commands are listed in the following table.

Command	For details, see...	Description
Certify Files	Using the Certify Files Command	Checks physical integrity of database files
Compact Files	Using the Compact Files Command	Compacts the database files based on indexes
Evaluate Free Space	Using the Evaluate Free Space Command	Evaluates the amount of free-space in a file
Delete Files	Using the Delete Files Command	Deletes database files

Command	For details, see...	Description
Reindex Files	Using the Reindex Files Command	Reindexes database files to repair damaged file indexes or to perform free-space garbage collection serially
Reset Timestamps	Using the Reset Timestamps Command	Resets database timestamps
Verify Checksums	Using the Verify Checksums Command	Performs a checksum analysis of your backed-up database files or your restored system to verify that the files have not been corrupted by a hardware or environmental problem during the backup or restore process
Set File Attributes	Using the Set File Attributes Command	Sets the initial file size and the extent of file growth for selected files
Backup Locations	Using the Backup Locations Command	Enables you to add or change logical backup destinations
Backup Database	Using the Backup Database Command	Backs up all or selected database files offline
Restore Database	Using the Restore Database Command	Provides a submenu that enables you to restore your database files from a JADE online or offline backup and performs roll-forward recovery or restore your database files from a JADE online or offline backup with no recovery
Initiate Recovery	Using the Initiate Recovery Command	Initiates the recovery of your database
Backup Journals	Using the Backup Journals Command	Backs up a selected journal offline
Restore Journals	Using the Restore Journals Command	Restores a journal from backup
Verify Journals	Using the Verify Journals Command	Checks the integrity of your database journal files
Dump Journals	Using the Dump Journals Command	Produces a formatted database journal files
Journaling Rates	Producing Journaling Rates Reports	Produces journaling rates text file and CSV file reports
Production Mode	Using the Production Mode Command	Sets or unsets production mode

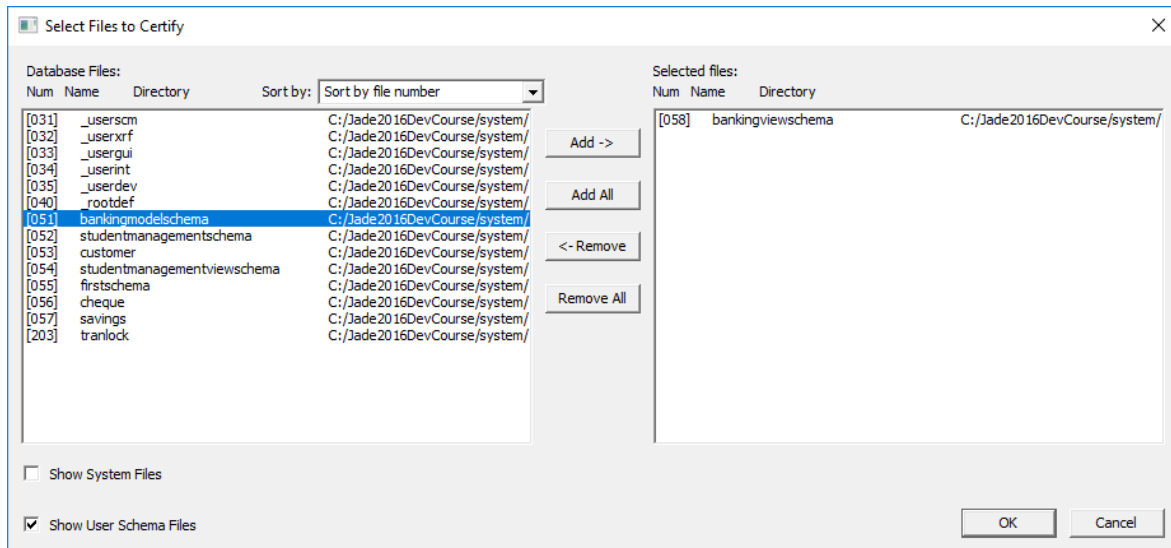
If the **SuppressDialogs** parameter in the **[PersistentDb]** section of your JADE initialization file is set to **false** (the default) and the database encounters a resource exception during a database operation (for example, out of disk, missing files, or allow file replace message), a message is displayed advising you of this. For example, if the backup process has insufficient disk space to complete the operation, the name of the file and the required amount of disk space are displayed, to enable you to free up disk space before continuing with the backup of that file, if required. If this parameter is set to **true**, no message is displayed and an error is returned which is output to your message log file.

Using the Certify Files Command

Use the **Certify Files** command from the Operation menu or click the **Certify Files** toolbar button, to perform a read-only check on the integrity of your database files. For details about certifying files, see "[Database Certification](#)", in Chapter 3.

» To certify database files

1. Select the **Certify Files** command from the Operation menu or click the **Certify Files** toolbar button. The Select Files to Certify dialog, shown in the following image, is then displayed.



2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
3. Select the required database file or files for certification, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.
 - To remove all of the files from the **Selected files** list box, click the **Remove All** button.
 - Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. If you want to look for system database files, check the **Show System Files** check box. (By default, system files are not displayed for selection; that is, user schema files only are displayed when you first access this dialog.)
6. If you do not want user schemas displayed for selection, uncheck the **Show User Schema Files** check box. (By default, user schema files are displayed for selection when you first access this dialog.)
7. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database file certification complete>>
```

Detailed results of the operation are written to the **certify.log** file, which you can view by selecting the **certify.log** command from the View submenu displayed when you select the **View Logs** command. (For details, see "[Viewing Log Files](#)", later in this chapter.)

Using the Compact Files Command

Use the **Compact Files** command from the Operation menu or click the **Compact Files** toolbar button, to optimize the use of storage and to reduce fragmentation. For details, see "[Compacting Files](#)", in Chapter 3.

» To compact your database files

1. Select the **Compact Files** command from the Operation menu or click the **Compact Files** toolbar button. The Select Files to Compact dialog is then displayed.
2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
3. Select the required database file or files for compaction, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.
 - To remove all of the files from the **Selected files** list box, click the **Remove All** button.
 - Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. If you want to look for system database files, check the **Show System Files** check box. (By default, system files are not displayed for selection; that is, user schema files only are displayed when you first access this dialog.)
6. If you do not want user schemas displayed for selection, uncheck the **Show User Schema Files** check box. (By default, user schema files are displayed for selection when you first access this dialog.)
7. If you want to compact multiple database files in parallel, check the **Use multiple workers** check box and then specify in the **workers** text box the number of workers (greater than 1) that you require.

The **Use multiple workers** check box and the **workers** text box are unchecked and display zero (0), respectively, by default.

If you do not specify the number of workers or you specify a number less than two workers, serial compact actions are performed.

- 8. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database file compaction complete>>
```

Detailed results of the operation are written to the **compact.log** file. (To view the log, select the **compact.log** submenu from the View menu **View Logs** command. For details, see "[Viewing Log Files](#)", later in this chapter.)

The files created during the compaction process are listed in the following table.

File	Description
<i>database-file.reo</i>	Working copy of original file – output of compaction
compact.log	Summary of compaction process, including errors

The **database-file.reo** work file is present only during the compaction process, and is left behind in the event of an interruption to the compaction process. The **compact.log** file is always created, and successive compactations are appended to the log file.

Note The **database-file.reo** file is usually created in the same directory as the file that is being compacted. If you want to locate the file in a different directory, specify the name in the **Working Directory** text box of the Set Preferences dialog. For details, see "[Setting Your JADE Database Utility Preferences](#)", later in this chapter.

Using the Evaluate Free Space Command

Use the **Evaluate Free Space** command from the Operation menu or click the **Evaluate Free Space** toolbar button, to analyze and report the amount of free-space remaining in database files. For more details, see "[Evaluating Free-Space](#)", in Chapter 3.

» To evaluate free-space in your files

- 1. Select the **Evaluate Free Space** command from the Operation menu or click the **Evaluate Free Space** toolbar button. The Select Files for Evaluate Free Space dialog is then displayed.
- 2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
- 3. Select the required database file or files for evaluation, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.
 - To remove all of the files from the **Selected files** list box, click the **Remove All** button.
 - Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. If you want to look for system database files, check the **Show System Files** check box. (By default, system files are not displayed for selection; that is, user schema files only are displayed when you first access this dialog.)
6. If you do not want user schemas displayed for selection, uncheck the **Show User Schema Files** check box. (By default, user schema files are displayed for selection when you first access this dialog.)
7. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Free space evaluation complete>>
```

Detailed results of the operation are written to the **fspace.log** file. (To view the log, select the **fspace.log** command from the View menu **View Logs** submenu. For details, see "[Viewing Log Files](#)", later in this chapter.)

Using the Delete Files Command

Use the **Delete Files** command from the Operation menu or click the **Delete Files** toolbar button, to delete test data. (If you delete a test database file and you subsequently start creating new instances of the data, the test database file is recreated.)

Note You can delete user database files only; you cannot delete system database files.

» To delete your database files

1. Select the **Delete Files** command from the Operation menu or click the **Delete Files** toolbar button. The Select Files to Delete dialog is then displayed.
2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
3. Select the required database file or files for deletion, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.
 - To remove all of the files from the **Selected files** list box, click the **Remove All** button.
 - Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. As each file is deleted, the status window is updated to indicate the success (or failure) of the operation. You can abort the deletion run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Delete files complete>>
```

Using the Reindex Files Command

Use the **Reindex Files** command from the Operation menu to reindex database files to repair damaged file indexes or to perform free-space garbage collection serially. For more details, see "[Reindexing Database Files](#)", in Chapter 3.

» To reindex your database files

1. Select the **Reindex Files** command from the Operation menu. The Select Files to Reindex dialog is then displayed.
2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
3. Select the required database file or files for reindexing, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.
 - To remove all of the files from the **Selected files** list box, click the **Remove All** button.
 - Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. If you want to look for system database files, check the **Show System Files** check box. (By default, system files are not displayed for selection; that is, user schema files only are displayed when you first access this dialog.)
6. If you do not want user schemas displayed for selection, uncheck the **Show User Schema Files** check box. (By default, user schema files are displayed for selection when you first access this dialog.)
7. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the reindex run, by clicking the **Cancel** button.

A running display that provides a progress report is produced on your screen while the operation is running. At the end of the operation, the running display provides a completion report, highlighting the number of files processed and the number of files found with errors.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database file reindex complete>>
```

Detailed results of the operation are written to the **reindex.log** file. (To view the log, select the **reindex.log** submenu from the View menu **View Logs** command. For details, see "[Viewing Log Files](#)", later in this chapter.)

Using the Reset Timestamps Command

Use the **Reset Timestamps** command from the Operation menu or click the **Reset Timestamps** toolbar button, to reset the timestamps in your database files. (For details, see "[Resetting Timestamps](#)", in Chapter 3.)

Caution Use this command with care and only when it is necessary to override timestamps when files have been legitimately replaced.

» To reset data file timestamps

1. Select the **Reset Timestamps** command from the Operation menu or click the **Reset Timestamps** toolbar button. (You cannot reset timestamps if database recovery is required.) The Select Files for Timestamp Reset dialog is then displayed.
2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
3. Select the required database file or files whose timestamps you want reset, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.

- To remove all of the files from the **Selected files** list box, click the **Remove All** button.
- Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. If you want to look for system database files, check the **Show System Files** check box. (By default, system files are not displayed for selection; that is, user schema files only are displayed when you first access this dialog.)
6. If you do not want user schemas displayed for selection, uncheck the **Show User Schema Files** check box. (By default, user schema files are displayed for selection when you first access this dialog.)
7. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

When the operation is finished, the following message is displayed in the running report.

```
<<Reset timestamps complete>>
```

Using the Verify Checksums Command

Use the **Verify Checksums** command from the Operation menu to perform a checksum analysis of your backed up database files or your restored system to verify that the files have not been corrupted by a hardware or environmental problem during the backup or restore process.

We recommend that you perform a checksum analysis of any backup that has been moved across media, especially if transferred across a network.

» To verify checksums

1. Select the **Verify Checksums** command from the Operation menu. The Select Directory dialog, shown in the following image, is then displayed.



2. In the **Directory** combo box, select the directory containing your backed up or restored database files from the combo box list, or specify the valid path in the text area of the combo box.

The list portion of the combo box lists up to the last 10 backup paths and the last 10 database paths, to enable you to select the directory that you require.

If you are unsure of the directory, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder selection dialog that enables you to select the path in which your backed up or restored files are located.

The path must exist. If you have previously run the checksum verification in the current JADE release, the path that you used the last time the operation was performed is displayed in this text box.

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

The checksums verification operation is driven from the **backupinfo** file in the specified backup directory or the **restoreinfo** file in the database system directory. Each file whose checksum is verified is displayed in the running report in the JADE Database utility window as the analysis is performed.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Checksum verification complete>>
```

Detailed results of the operation are written to the **checksum.log** file, which you can view by selecting the **checksum.log** command from the View submenu displayed when you select the **View Logs** command. (For details, see "[Viewing Log Files](#)", later in this chapter.)

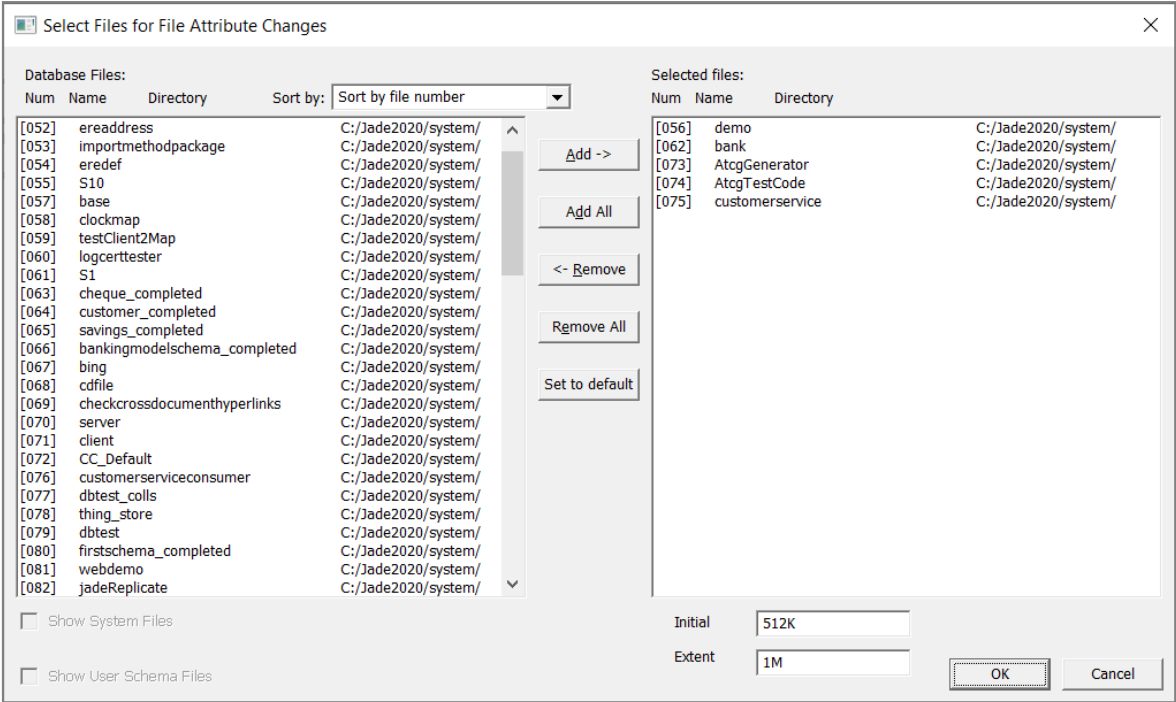
Using the Set File Attributes Command

The **Set File Attributes** command from the Operation menu specifies the initial file size and the extent of file growth for selected database files.

As database files grow in variable increments on demand, and the gradual growth incurs both a direct I/O performance penalty and an indirect performance penalty due to disk fragmentation, use the **Set File Attributes** command to tailor the amount by which your database files grow.

» To set database file attributes

1. Select the **Set File Attributes** command from the Operation menu. The Select Files for File Attribute Changes dialog, shown in the following image, is then displayed.



2. If you do not want the database files in the **Database Files** list box at the left of the dialog sorted by file number (the default), select the **Sort by file name** or the **Sort by file directory and name** item in the **Sort by** drop-down list box.
3. Select the required database file or files for certification, by performing one of the following actions.
 - Select the files from the **Database Files** list box. The selected files are highlighted. When you have selected all of the files that you require, click the **Add** button.
 - To select all of the database files, click the **Add All** button.
 - Select each file by double-clicking the file in the **Database Files** list box.

The selected files are then moved to the **Selected files** list box at the right of the dialog.

4. To remove database files from the **Selected files** list box, perform one of the following actions.
 - Select the files for removal from the **Selected files** list box. When you have selected all of the files for removal, click the **Remove** button.
 - To remove all of the files from the **Selected files** list box, click the **Remove All** button.
 - Select each file by double-clicking the file in the **Selected files** list box.

The selected file names are moved back to the **Database Files** list box at the left of the dialog.

5. Click the **Set to default** button, to restore the initial size and the growth extent increment size for all files displayed in the **Selected files** list box to the default values specified by the **DefFileGrowthIncrement** and **DefInitialFileSize** parameters in the [PersistentDb] section of the JADE initialization file.
6. In the **Initial Size** text box, specify the number of bytes that you require for the initial size of the files displayed in the **Selected files** list box. Set this value explicitly (for example, **524288**) or by using a prefix multiplier (for example, **512K**).

The default value is **128K** and the minimum value is **64K**.

7. In the **Extent Size** text box, specify the number of bytes that you require for the growth increment extent of the files displayed in the **Selected files** list box. Set this value explicitly (for example, **1048576**) or by using a prefix multiplier (for example, **1M**).

The default value is **128K** and the minimum value is **64K**.

8. When your list of selected files is complete, click the **OK** button to start the operation. Alternatively, click the **Cancel** button to abandon your selections.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<File size changes completed>>
```

Using the Backup Locations Command

Use the **Backup Locations** command from the Operation menu to add or change logical backup locations.

The backup location name and directory are displayed in the second and third columns, respectively, of the Select File Backup Locations dialog.

A backup location is a mapping or alias for a directory path that can be used as a parameter in backup operations.

Tip This operation is useful when you are backing up large user database files, as it enables you to selectively backup files not only to different directories but also to different nodes. For details, see "[Selecting Your File Backup Locations](#)", later in this chapter.

» To define your backup files location

1. Select the **Backup Locations** command from the Operation menu. The Backup Locations dialog is then displayed. You can define or maintain backup locations when you do not currently have an open database.
2. Click the **Add** button, to add a new backup location. The Add Backup Location dialog is then displayed.
3. In the **Name** combo box, specify the logical name for a backup location, or select the required value from the list if you have defined the required database file location previously.

You can specify the location name as a parameter in backup and restore operations, to define a logical destination for an individual database file. Each location name can be associated with a different path, if required, and can be located on a different node.

The name must follow the rules for identifiers, with a maximum length of 100 characters; for example, **BackupDest5**.

4. In the **Directory** text box, specify the path name for the backup directory associated with the logical location name. You must specify an existing directory that is valid on the server. (An error dialog is displayed if the specified directory does not exist.)

Alternatively, click the adjacent browse button (indicated by the ... point of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the backup directory in which your logical database file will be located.

5. Click the **OK** button, to validate your added parameters and save the defined backup location in the system registry. Your specified location name is then displayed and selected in the Backup Locations dialog **Location Names** list box and the associated directory name is displayed below this list box.

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

6. Repeat steps 2 through 5 for each database file location that you want to add.
7. When you have added all of the backup file locations that you require, click the **Close** button. The Backup Locations dialog is then closed, and focus returns to the JADE Database utility window.

» To remove an existing backup files location

1. Select the **Backup Locations** command from the Operation menu. The Backup Locations dialog is then displayed.
2. In the **Location Names** list box, select the existing backup location that you want to remove. The directory name of your selected backup location is then displayed below the list box.
3. Click the **Remove** button.
4. Repeat steps 2 and 3 for each existing location name that you want to remove.
5. When you have removed all of the backup file locations that you require, click the **Close** button.

The Backup Locations dialog is then closed, and focus returns to the JADE Database utility window.

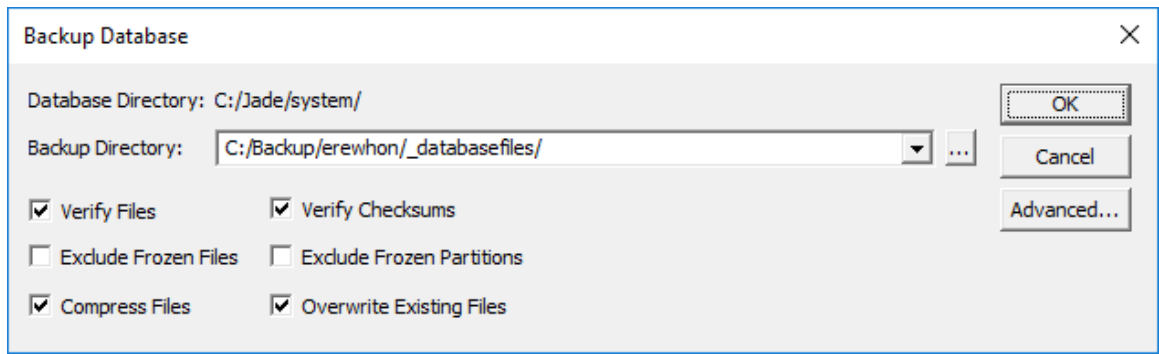
Using the Backup Database Command

Use the **Backup Database** command from the Operation menu to specify the parameters for and to initiate an offline database backup.

You can backup your complete database or selected files only. For more details, see ["Offline Full Backup"](#), in Chapter 3.

» To perform an offline backup of your database

1. Select the **Backup Database** command from the Operation menu. The Backup Database dialog, shown in the following image, is then displayed. (Your current database directory is displayed above the **Backup Directory** combo box.)



2. In the **Backup Directory** combo box, specify the path name for the backup destination directory where database files are copied during the backup process. You must specify a directory that is valid on the server.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the backup directory in which your backed up database files will be located.
3. Uncheck the **Verify Files** check box if you do not want to verify your database files as they are backed up. By default, this check box is checked, indicating that index block-to-target (object, subobject, and so on) verification is performed, to ensure that index entries resolve to the correct records in the file. (See also ["Verification during Backup"](#), in Chapter 3.)
4. Uncheck the **Verify Checksums** check box if you do not want to perform MD5 checksum verification of files as they are backed up, to ensure that corruption is not introduced by hardware or environmental problems. By default, this check box is checked.
5. Check the **Exclude Frozen Files** check box, to exclude frozen (read-only) database files from the backup. By default, this check box is unchecked.
6. Check the **Exclude Frozen Partitions** check box, to exclude frozen (read-only) database partitions from the backup. By default, this check box is unchecked.
7. Uncheck the **Compress Files** check box if you do not want to compress your database files as they are backed up. By default, the **Compress Files** check box is checked.
8. Check the **Overwrite Existing Files** check box, to overwrite any existing files in your specified destination backup directory. By default, the **Overwrite Existing Files** check box is unchecked.
9. Click the **Advanced** button, to exclude files from the offline backup or to backup files to different locations. The advanced backup parameters enable you to select (or deselect) individual files for backup and change the logical location name for each file, if required. For details, see ["Selecting Your File Backup Locations"](#).

10. When you have defined all of your backup parameters and optionally your file backup locations, click the **OK** button, to save your parameters in the system registry and initiate the database backup process. Alternatively, click the **Cancel** button, to abandon your selections and close the dialog without initiating the backup operation.

Tip A *fast* backup is performed if files are neither verified nor compressed. In a fast file backup, database files are backed up in a similar fashion to a standard file copy, using large buffers and asynchronous I/O to speed up the copy process.

The fast backup mode bypasses the database access routines and cache management, and does not verify data as it is backed up.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run by clicking the **Cancel** button.

Caution If any errors are encountered during the backup, the backup cannot be restored.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

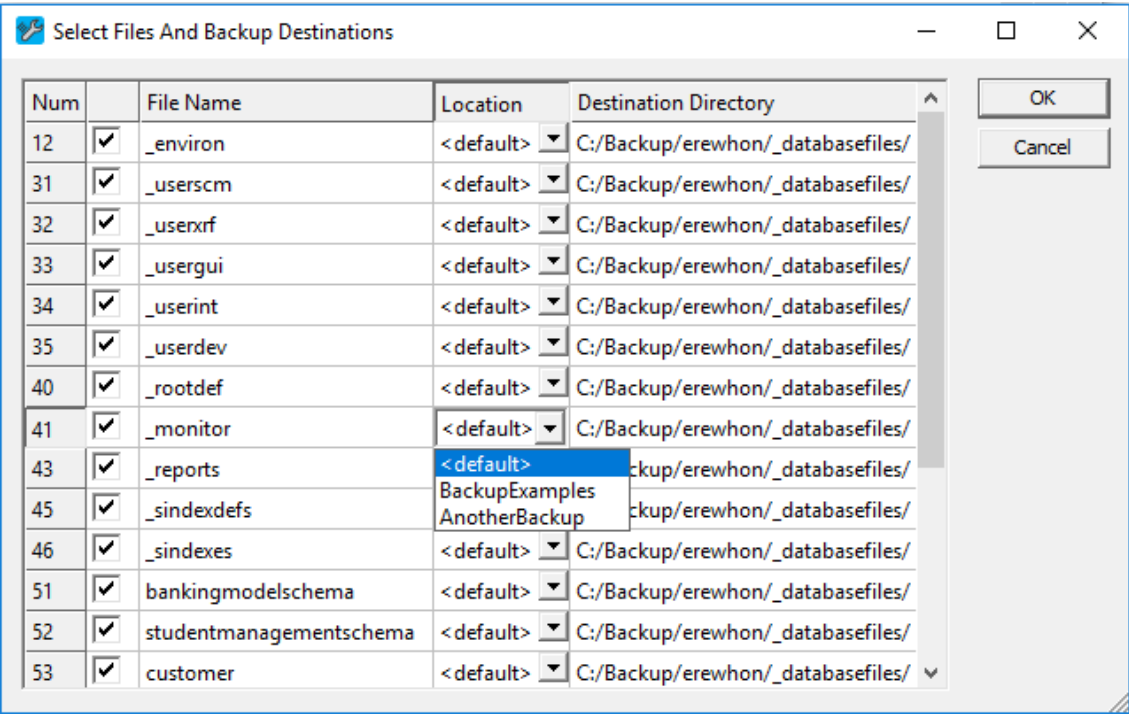
```
<<Database backup complete>>
```

Information or errors from the backup operation are recorded in the **backup.log** file, located in the default database directory.

To view the log, select the **backup.log** submenu from the View menu **View Logs** command. For details, see "[Viewing Log Files](#)", later in this chapter.

Selecting Your File Backup Locations

When you click the **Advanced** button in the Backup Database dialog, the Select Files And Backup Destinations dialog is displayed. An example of the Select Files And Backup Destinations dialog is shown in the following image.



Use this dialog to select (or deselect) individual files for backup and to change the logical location name and destination for each file, if required.

The **Num** column contains a schema-defined file number for each file selected for offline backup, the **File Name** column contains the name of each of your database files, and the **Destination Directory** column displays the destination directory of each database file.

The locations and destination directories are those defined in the **Name** and **Directory** fields of the Add Backup Location dialog, and can be added or removed only by using the Backup Locations dialog. For details, see "[Using the Backup Locations Command](#)", earlier in this chapter.

» **To select the files for backup and the individual file backup locations**

1. In the column of check boxes at the left of the **File Name** column, uncheck each checked box (indicated by a ☒) to exclude the corresponding database file from the backup. Alternatively, check any file for backup that is currently excluded (that is, its check box is unchecked) if you now want it to be backed up.
2. In the **Location** column, click the drop down list box if you want to change the file location for a specific database file.

Each file location that you have specified in the Backup Locations dialog is listed, to enable you to select the logical location for your backed up database file. For example, if you are backing up two large user database files, you may want to back up a file called **Bok** to a **Dest1** file location on drive **d:\backups** and another file called **Internet** to a **Dest3** file location on drive **s:\large\backups**, where you know there is sufficient room for the backed up file.

3. When you have selected all of the files that you want backed up and the logical locations of these backed up files, click the **OK** button. The Backup Database dialog is then redisplayed, to enable you to make any final modification to your parameters before initiating your backup.

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

Using the Restore Database Command

You can restore a database that was backed up from an online or an offline backup. The **Restore Database** command from the Operation menu provides a submenu that enables you to restore your database with roll-forward recovery or to restore your database with no recovery. For details, see the following subsections.

Notes Selecting a **Restore Database** submenu command causes the JADE Database utility to close any database that is currently open.

You must specify or select a directory that is valid on the server.

Restoring Your Database with Roll-Forward Recovery

To restore and recover a database, perform the following actions.

1. Restore all archived transaction journals backed up since the last full backup. (For details, see "[Using the Restore Journals Command](#)", later in this chapter.)
2. Use the **Restore and Roll-Forward** submenu command from the Operation menu **Restore Database** command to restore your database files from a specified backup directory *and* perform a roll-forward recovery of your database.

For definitions of the terms *recover* and *restore*, see "[Recovery Concepts and Strategies](#)", in Chapter 3.

» To restore and recover your database

1. Select the **Restore and Roll-Forward** submenu command from the Operation menu **Restore Database** command.

The Restore Database And Roll-Forward dialog, shown in the following image, is then displayed.

2. In the **Database Path** combo box, specify or select the path to which your backed up database is to be restored. (A list of the most recently used databases is displayed.)

You must specify a directory that is valid on the server.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the database directory to which your backed up database files will be restored.

3. In the From Backup Location group box, select the **Use Location Name** option button to specify the logical location name of the directory that contains the **_control.dat** file for your backed up database.

The **Directory Name** option button is selected by default, to enable you to specify the fully qualified name of the directory that contains your **_control.dat** control file, or to select the required directory from the list portion.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the directory in which your database control files were backed up.

4. In the Roll-Forward Recovery Options group box, select the type of roll-forward recovery that you require. The **Recover transactions to end of last journal** option is selected by default.

If you want to perform a roll-forward recovery:

- Up to and including a specific transaction journal, select the **Recover transactions to end of journal** option button.
 - To a specific date and time, select the **Recover transactions up to date and time** option button.
 - Up to and including a specific audit record, select the **Recover transactions to audit serial number** option button.
5. If you selected the option to recover transactions to the end of a specific journal in the previous step, in the corresponding text box specify the number of the last journal up to which you want to recover; for example, **9** (if you want to recover transaction journal files up to and including **db0000000009.log**).
 6. If you selected the option to recover transactions up to a specified date and time in step 4 of this instruction, specify the required end-date and end-time in the text boxes, in the following formats:

dd\mm\yyyy

HH:MM:SS

Your roll-forward recovery applies all committed transactions up to the specified date and time. Any transactions that completed after the specified date and time are not reapplied.

7. If you selected the option to recover transactions up to and including a specified audit record, specify the required audit record serial number.

Your roll-forward recovery applies all committed transactions until the audit record serial number is greater than the termination serial number.

8. If you do *not* want the database engine to perform destination file checksum verification in-line with the restore operation, uncheck the **Verify Checksums** check box.

This check box, which is checked by default, causes each file restore operation to additionally process the file data written to the destination database directory to verify that the checksum matches the source data checksum.

9. If you want the database recovery mechanism that performs a roll-forward recovery of a backed up Synchronized Database Service (SDS) system to clear the SDS role of that database during the recovery process, check the **Clear SDS role** check box.

When you check this check box, the database role is set to undefined without regard to the prior value of the database role, and a new database identifier is generated at the end of the roll-forward recovery. If this check box is unchecked, attempts to perform a roll-forward recovery of a backup of an SDS secondary system with termination conditions are disallowed.

10. Check the **Exclude Frozen Files** check box, to exclude frozen (read-only) database files from the restore operation. By default, this check box is unchecked.
11. Check the **Exclude Frozen Partitions** check box, to exclude frozen (read-only) database partitions from the restore operation. By default, this check box is unchecked.
12. Click the **OK** button, to save your parameters in the system registry and initiate the restoration and recovery of your backed up database. Alternatively, click the **Cancel** button to abandon your selections.

While the restore operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the restore, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database restore complete>>
```

If the restore was successful, the recovery phase then begins. While roll-forward recovery is running, a progress dialog is displayed, which enables you to monitor the progress of the operation. You can abort the recovery, by clicking the **Cancel** button.

At the end of processing the available transaction journals, the Roll Forward Recovery – Wait For Journal dialog may be displayed.

If no more journal files are to be provided for roll-forward recovery, select the **Finish** button in the Roll Forward Recovery – Wait For Journal dialog. For more details, see "[Roll-Forward Recovery of a Standby Database](#)", in Chapter 3.

At the end of the roll-forward recovery operation, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database recovery complete>>
```

Detailed results of the restore and recovery operations are written to the **recovery.log** file in the location specified by the [ActivityLogDirectory](#) parameter in the [PersistentDb] section of the JADE initialization file (documented in the *JADE Initialization File Reference*).

To view the recovery log file, select the **View Logs** command from the View menu and then select the **recovery.log** command from the submenu. (For details, see "[Viewing Log Files](#)", later in this chapter.)

For details about roll-forward recovery restrictions, see "[Roll-Forward Recovery Restrictions](#)", in Chapter 3.

Caution If any errors are encountered during the restore or roll-forward recovery, or you abort during the operation, you may not be able to use the database.

To perform the restore and recovery operation, the JADE Database utility must have exclusive access to the database. If you select a JADE database that is already in use by another application, an error is reported.

Restoring Your Database with No Recovery

Use the **Restore With No Recovery** submenu command from the Operation menu **Recover Database** command, to restore your database files with no recovery.

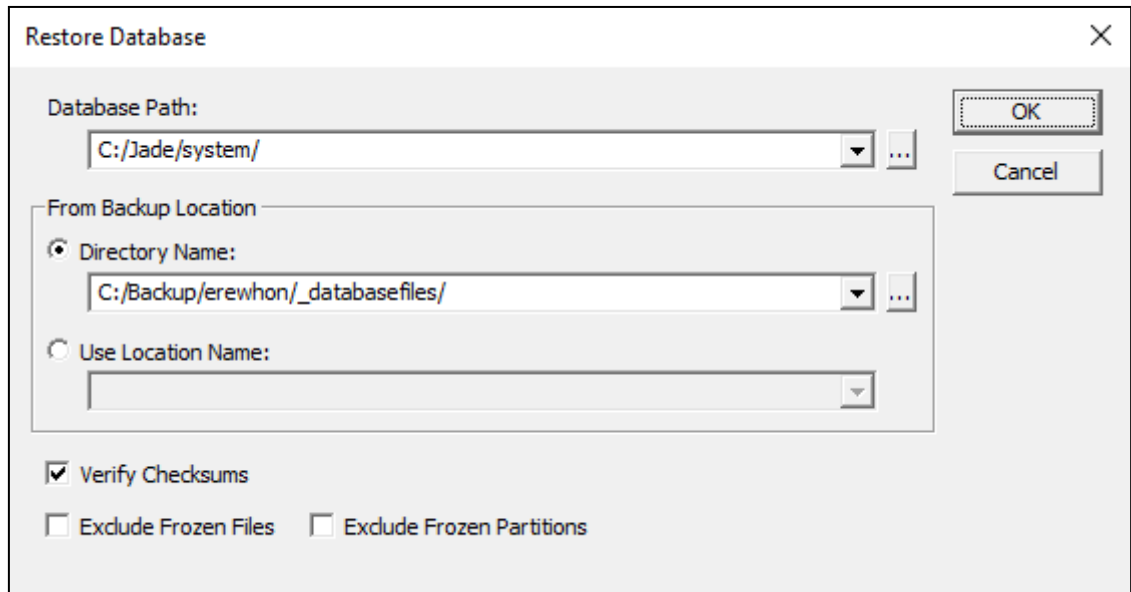
Note If an online backup is restored by using the **Restore With No Recovery** command, at a minimum the transaction journal that was active during the backup is required to recover the backup.

No transaction journal is required after restoring an offline or quiesced backup.

» To restore your database with *no* recovery

1. Select the **Restore With No Recovery** submenu command from the Operation menu **Recover Database** command.

The Restore Database dialog, shown in the following image, is then displayed.



2. In the **Database Path** combo box, specify or select the path to which your backed up database is to be restored. (A list of the most recently used databases is displayed.) You must specify a directory that is valid on the server.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the database directory to which your backed up database files will be restored.

3. In the From Backup Location group box, select the **Use Location Name** option button, to specify the logical location name of the directory that contains the **_control.dat** file for your backed up database.

The **Directory Name** option button is selected by default, to enable you to specify the fully qualified name of the directory that contains your **_control.dat** control file, or select the required directory from the list portion.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the directory in which your database control files were backed up.

4. If you do *not* want the database engine to perform destination file checksum verification in-line with the restore operation, uncheck the **Verify Checksums** check box.

This check box, which is checked by default, causes each file restore operation to additionally process the file data written to the destination database directory, to verify that the checksum matches the source data checksum.

5. Check the **Exclude Frozen Files** check box, to exclude frozen (read-only) database files from the restore operation. By default, this check box is unchecked.
6. Check the **Exclude Frozen Partitions** check box, to exclude frozen (read-only) database partitions from the restore operation. By default, this check box is unchecked.
7. Click the **OK** button, to save your parameters in the system registry and initiate the restoration and recovery of your backed up database. (Alternatively, click the **Cancel** button to abandon your selections.)

While the restore operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the restore, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database restore complete>>
```

Detailed results of the restore operation are written to the **recovery.log** file in the default database directory. To view the recovery log file, select the **View Logs** command from the View menu and then select the **recovery.log** command from the submenu. (For details, see "[Viewing Log Files](#)", later in this chapter.)

Caution If any errors are encountered during the restore operation or you abort during the operation, you may not be able to use the database.

To perform the restore operation, the JADE Database utility must have exclusive access to the database. If you select a JADE database that is already in use by another application, an error is reported.

Using the Initiate Recovery Command

Use the **Initiate Recovery** command from the Operation menu or click the **Initiate Recovery** toolbar button, to perform a roll-forward recovery of previously restored offline backup files to the end of the last transaction, to a specific date and time, or to a specific audit record.

You can initiate recovery only when the database is closed; that is, this command is disabled if you have opened the database for update in the current session.

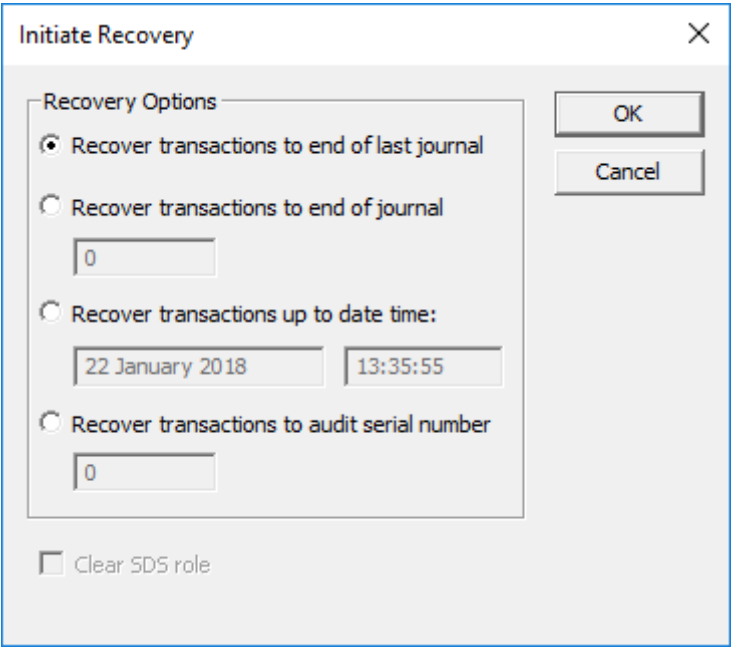
This command enables you to initiate the recovery of a backup that has been restored separately (for example, direct from tape) when you have restored your database to its final production location and then opened the database in the JADE Database utility.

For definitions of the terms *recover* and *restore*, see "[Recovery Concepts and Strategies](#)", in Chapter 3. For details about archived transaction journals, see "[Database Archiving Modes](#)", in Chapter 3.

» To initiate a roll-forward recovery

1. Select the **Initiate Recovery** command from the Operation menu or click the **Initiate Recovery** toolbar button.

The Initiate Recovery dialog, shown in the following image, is then displayed.



2. In the Recovery Options group box, select the type of roll-forward recovery that you require. The **Recover transactions to end of last journal** option is selected by default.
If you want to perform a roll-forward recovery:
 - Up to and including a specific transaction journal, select the **Recover transactions to end of journal** option button.
 - To a specific date and time, select the **Recover transactions up to date and time** option button.
 - Up to and including a specific audit record, select the **Recover transactions to audit serial number** option button.
3. If you selected the option to recover to the end of a specific journal, in the corresponding text box, specify the number of the last journal up to which you want to recover; for example, **9** (if you want to recover transaction journal files up to and including **db0000000009.log**).
4. If you selected the option to recover transactions up to a specified date and time in step 2 of this instruction, specify the required end-date and end-time in the text boxes, in the following formats:

dd\mm\yyyy
HH:MM:SS

Your roll-forward recovery applies to all committed transactions up to the specified date and time. Any transactions that completed after the specified date and time are not reapplied.

5. If you selected the option to recover transactions up to and including a specified audit record, specify the required audit record serial number.
Your roll-forward recovery applies all committed transactions until the audit record serial number is greater than the termination serial number.

6. If you want the database recovery mechanism that restores a backed up SDS system to clear the SDS role of that database during the restore process, check the **Clear SDS role** check box.

When this check box is checked, the database role is set to undefined and a new database identifier is generated at the end of the roll-forward recovery.

If the open database does not have an SDS role, the **Clear SDS role** check box is disabled.

If this check box is unchecked, attempts to perform a roll-forward recovery of a backup of an SDS secondary system with termination conditions are disallowed.

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

While roll-forward recovery is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the recovery, by clicking the **Cancel** button.

At the end of processing the available transaction journals, the Roll Forward Recovery – Wait For Journal dialog may be displayed. If no more journal files are to be provided for roll-forward recovery, select the **Finish** button in the Roll Forward Recovery – Wait For Journal dialog. For more details, see "[Roll-Forward Recovery of a Standby Database](#)", in Chapter 3.

At the end of the roll-forward recovery operation, the progress dialog is closed and the following message is displayed in the running report.

```
<<Database recovery complete>>
```

Information or errors from the roll-forward recovery operation are recorded in the **recovery.log** file, located in the default database directory. (To view the log, select the **recovery.log** submenu from the View menu **View Logs** command. For details, see "[Viewing Log Files](#)", later in this chapter.) For details about roll-forward recovery restrictions, see "[Roll-Forward Recovery Restrictions](#)", in Chapter 3.

Caution If any errors are encountered during the roll-forward recovery operation or you abort the recovery, you may not be able to use the database. To perform this operation, the JADE Database utility requires exclusive access to the database. If you select a JADE database that is already in use by another application, an error is reported.

Using the Backup Journals Command

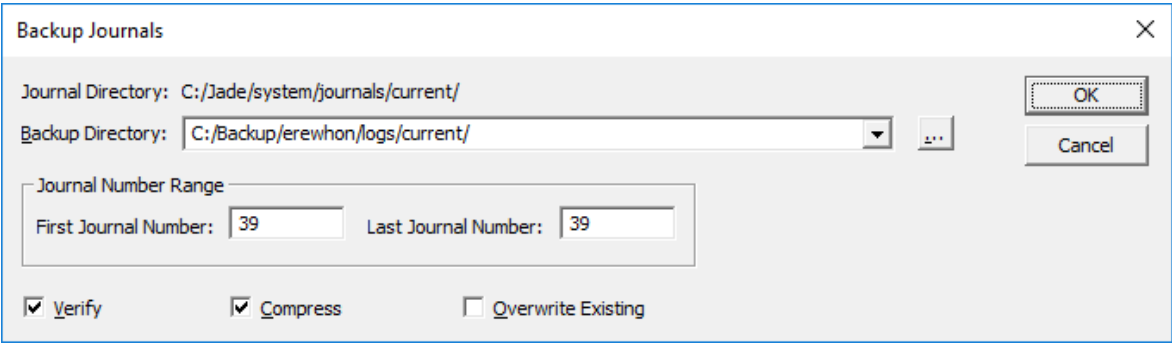
Use the **Backup Journal** command from the Operation menu to initiate the offline backup of a specified transaction journal.

Note It is your responsibility to ensure that you manually back up transaction journals. For details, see "[What Is a Backup?](#)", "[Physical Database Structures](#)", "[Transaction Journal Files](#)", and "[Backups](#)", in Chapter 3.

» To backup your transaction journals

1. Select the **Backup Journals** command from the Operation menu.

The Backup Journals dialog, shown in the following image, is then displayed.



The journal directory in your current database path is displayed above the **Backup Directory** combo box.

2. In the **Backup Directory** combo box, specify the path name for the backup destination directory where transaction journal files are copied during the backup process. You must specify a directory that is valid for the server.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the directory to which your journal files will be backed up.
3. In the **First Journal Number** text box, specify the number of the first journal file that you want to back up. Alternatively, if you want to back up one journal file only, specify only the number of that file. For example, enter **3** in this text box to back up only the journal file **db0000000003.log** located in the path **d:\jade\system\logs\current**.

An error is returned if you do not specify a valid file number in this text box.
4. If you want to back up a range of files, in the **Last Journal Number** text box specify the number of the last journal file that you want to back up; for example, **9** (if you want to back up files up to and including **db0000000009.log**).

If you do not specify a journal file in this text box, only the journal file specified in the **First Log Number** text box is backed up. You can specify a single journal file, or the start and end files in a consecutive range of files. You cannot specify a non-consecutive file range.
5. Check the **Verify** check box, to verify your transaction journals as they are backed up.
6. Check the **Compress** check box, to compress your transaction journal files as they are backed up. By default, backed up files are not compressed.
7. Check the **Overwrite Existing** check box, to overwrite any existing transaction journal files in your specified destination backup directory. By default, existing files are not overwritten.
8. Click the **OK** button, to initiate the offline backup of your transaction journal. Alternatively, click the **Cancel** button, to abandon the backup transaction journal operation.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

Information or errors from the backup transaction journal operation are recorded in the **backup.log** file, located in the default database directory. (To view the log, select the **backup.log** submenu from the View menu **View Logs** command. For details, see "[Viewing Log Files](#)", later in this chapter.)

Caution If any errors are encountered during the journal backup, the backup log file is removed.

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Journal backup complete>>
```

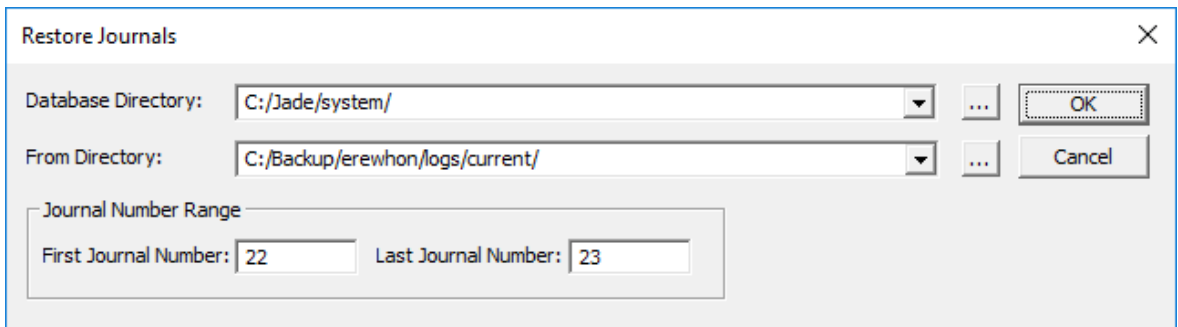
Using the Restore Journals Command

Use the **Restore Journals** command from the Operation menu to initiate the offline restoration of a specified transaction journal. Transaction journals can be restored when you do not currently have an open database.

» To restore your transaction journal

1. Select the **Restore Journals** command from the Operation menu.

The Restore Journals dialog, shown in the following image, is then displayed.



2. In the **Database Directory** combo box, specify or select the path of the database directory, to which journals are restored. You must specify a directory that is valid on the server.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the database directory to which your backed up transaction journals will be restored.

3. In the **From Directory** combo box, specify the fully qualified name of the directory to which your transaction journals were backed up or select the required directory from the drop-down list. Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the directory in which your journals are backed up.

4. In the **First Journal Number** text box, specify the number of the first journal file that you want to restore from backup.

Alternatively, specify only the number of that file if you want to restore one journal file only. For example, enter **3** in this text box to restore only the journal file **db0000000003.log**.

An error is returned if you do not specify a valid file number in this text box.

5. If you want to restore a range of files, in the **Last Journal Number** text box specify the number of the last journal file that you want to restore; for example, **9** (if you want to restore files up to and including **db0000000009.log**).

If you do not specify a journal file in this text box, only the journal file specified in the **First Journal Number** text box is restored. You can specify a single journal file, or the start and end files in a consecutive range of files. You cannot specify a non-consecutive file range.

- Click the **OK** button, to initiate the offline restoration of your transaction journals. Alternatively, click the **Cancel** button, to abandon the restore transaction journal operation.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button. Information or errors from the restore transaction journal operation are recorded in the **recovery.log** file in the location specified by the [ActivityLogDirectory](#) parameter in the [\[PersistentDb\]](#) section of the JADE initialization file (documented in the *JADE Initialization File Reference*).

To view the log, select the **recovery.log** submenu from the View menu **View Logs** command. (For details, see "[Viewing Log Files](#)", later in chapter.)

When the operation is finished, the progress dialog is closed and the following message is displayed in the running report.

```
<<Journal restoration complete>>
```

You can now reopen your database.

Using the Verify Journals Command

Use the **Verify Journals** command from the Operation menu to perform a verification of a specified transaction journal file. This operation verifies the physical integrity of the selected transaction journal file, performing a series of tests that verify the contents of records, including various cross-checks between adjacent records.

» To verify your transaction journal file

- Select the **Verify Journals** command from the Operation menu. The common File dialog (named the Select Journal(s) dialog) is then displayed.
- Select one or more journal files that you want to verify from the appropriate location.
- Click the **Open** button, to start the transaction journal file verification. Alternatively, click the **Cancel** button, to abandon the operation.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button. At the end of the operation, the running display provides a completion report, highlighting the number of verification errors that were encountered.

The following message is displayed.

```
<<Verify journals complete>>
```

Details of the verification operation are written to the *journal-file-name.scan* file, located in the **current** logs directory in your default database directory. (See also "[Using the Dump Journals Command](#)", later in this chapter, for details about producing a formatted recovery log file for diagnostic purposes, if required.)

Using the Dump Journals Command

Use the **Dump Journals** command from the Operation menu to initiate a formatted dump of a specified transaction journal file.

The dump contains an analysis of header and control information from audit records in the transaction journal. This formatted dump can be forwarded to JADE Support for diagnostic purposes, if required.

The formatted transaction journal files are named **dbnnnnnnnnnn.dump.txt**, with the *nnnnnnnnnn* value being a number in the range **1** through **9999999999**, and are output to the directory specified in the **JournalRootDirectory** parameter in the **[PersistentDb]** section of your JADE initialization file (for example, **d:\jade\logs\current\db0000076894.dump.txt**).

» **To dump transaction journal files**

- 1. Select the **Dump Journals** command from the Operation menu. The common File dialog (named the Select Journal(s) dialog) is then displayed.
- 2. Select one or more transaction journal files that you want to dump from the appropriate location.
- 3. Click the **Open** button, to start the transaction journal file dump. Alternatively, click the **Cancel** button, to abandon the operation.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button.

When the operation is finished, the progress dialog is closed, the running display provides a completion report highlighting the number of errors that were encountered, and the following message is displayed.

```
<<Dump journals complete>>
```

Producing Journaling Rates Reports

Use the **Journaling Rates** command from the Operation menu to initiate the production of the journaling rates text file and comma-separated values (CSV) file reports. You can use these reports, for example, to plan capacity for an SDS connection.

The **journalingrates.jra** report contains an analysis of header and control information from audit records in the transaction journal, a separate section for the rates of each journal (for example, peak data and transaction rates), followed by a summary of the journal analysis of all selected journals, if you selected more than one.

The **journalingrates.csv** report, which you can use to load the data into a third-party application such as Microsoft Excel, contains journaling analysis number information.

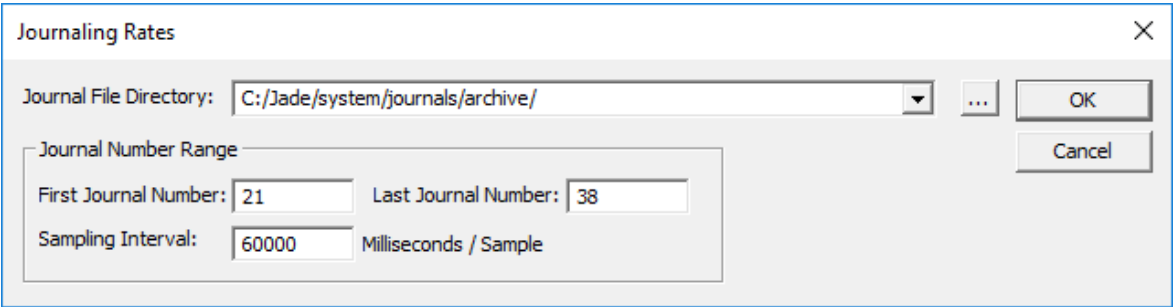
One line is produced for each sampling interval (which defaults to one minute, or 60000 milliseconds), with the comma-separated fields representing:

```
date, time, bytes-per-interval, active-transactions-per-interval,  
commit-transactions
```

» **To output journaling rate details to text and CSV files**

- 1. Select the **Journaling Rates** command from the Operation menu.

The Journaling Rates dialog, shown in the following image, is then displayed.



2. In the **Journal File Directory** combo box, specify or select the path in which the transaction journals that you want to sample are located; for example, `d:\jade\system\journals\current`. Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the journal directory in which the journal or journals are located.

The numbers of the first and last journal files in that directory are then displayed in the **First Journal Number** and **Last Journal Number** text boxes. If that directory contains one journal only, that journal number is displayed in both text boxes.

3. In the **First Journal Number** text box, specify the number of the first journal file whose rates you want to sample. Alternatively, specify only the number of that file if you want to sample one file only. For example, enter **3** in this text box to sample rates of only the journal file `db0000000003.log`. An error is returned if you do not specify a valid file number in this text box.
4. If you want to sample rates of a group of files, in the **Last Journal Number** text box specify the number of the last journal file whose rates you want to sample; for example, **9** (if you want to sample rates of files up to and including `db0000000009.log`). If you do not specify a journal file in this text box, only the rates of the journal file specified in the **First Journal Number** text box are sampled. You can specify a single journal file, or the start and end files in a consecutive group of files. You cannot specify a non-consecutive file range.
5. In the **Sampling Interval** text box, specify the number of minutes at which the journaling rates are sampled, if you do not want the specified journal file or files sampled every minute, (that is, the default value is **1**).
6. Click the **OK** button, to start the journaling rates reports. Alternatively, click the **Cancel** button, to abandon the operation.

While the operation is running, a progress dialog is displayed that enables you to monitor the progress of the operation. You can abort the run, by clicking the **Cancel** button. When the operation is finished, the progress dialog is closed, the running display provides a completion report highlighting the number of errors that were encountered, and the following message is displayed.

```
<<Journaling Rates complete>>
```

Journaling rates reports are output to a comma-separated values file named `journalingrates.csv` and a text file named `journalingrates.jra` in the same directory as the journal file or files selected for analysis.

If you perform this operation again, any existing `journalingrates.jra` and `journalingrates.csv` file reports in that directory are overwritten.

Using the Production Mode Command

Use the **Production Mode** command from the Operation menu to enable or disable production mode for your JADE database. (The **Production Mode** command is checked when production mode is enabled for your JADE database.)

For details, see "[Running JADE Production Mode Databases](#)", in Chapter 1 of the *JADE Runtime Application Guide*.

» To enable production mode

- Select the **Production Mode** command from the Operation menu.

A check mark symbol is then displayed to the left of the item in the menu, indicating that production mode is set for your JADE database.

» To disable production mode

- Select the **Production Mode** command from the Operation menu.

The check mark symbol is then removed from the left of the item in the menu, indicating that production mode is not set for your JADE database.

Specifying Your JADE Database Utility View Options

Use the View menu in the JADE Database utility to change the options associated with the JADE Database utility. The View menu commands, which are available in both the Database View and the Log File View, are listed in the following table.

Command	For details, see...	Description
Toolbar	Toggling the Display of the Toolbar	Toggles to hide or display the Tools toolbar
Status Bar	Toggling the Display of the Status Bar	Toggles to hide or display the status bar
Set Font	Selecting the Display Font	Displays the common Font dialog to select the font of displayed text
Set Printer Font	Selecting the Printer Font	Displays the common Font dialog to select the font used when printing
View Logs	Viewing Log Files	Provides access to the logs for each operation
Preferences	Setting Your JADE Database Utility Preferences	Sets the preferences for your current database utility session
Word Wrap	Word Wrap Command	Toggles the wrapping of text in the JADE Database utility window

Toggling the Display of the Toolbar

The toolbar is displayed by default, across the top of the JADE Database utility window; that is, a check mark symbol is displayed to the left of the **Toolbar** command.

This command is available in both the Database View and the Log File View.

» **To hide or display the toolbar**

- Select the **Toolbar** command from the View menu.

Toggling the Display of the Status Bar

The status bar is displayed by default across the bottom of the JADE Database utility window; that is, a check mark symbol is displayed to the left of the **Status Bar** command.

» **To hide or display the status bar**

- Select the **Status Bar** command from the View menu.

Selecting the Display Font

» **To specify the font used to display text in the JADE Database utility window or log file viewer**

- Select the **Set Font** command from the View menu if you want to change the default font from System bold 12 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 Database utility window.

Note You can specify a different font for the display of operation results in both the JADE Database utility window and the log file viewer.

Selecting the Printer Font

» To specify the font used when printing from the JADE Database utility

- Select the **Set Printer Font** command from the View menu if you want to change the font used when printing from the JADE Database utility window.

The common Font dialog is then displayed, to enable you to make your font selections. When you have made your font selections, focus then returns to the JADE Database utility window.

Note You can specify a different font for the printing of operation results from both the JADE Database utility window and the log file viewer. (For details, see "[Viewing Log Files](#)", in the following section.)

Viewing Log Files

A separate log is created for each type of database utility operation, in the location specified by the [ActivityLogDirectory](#) parameter in the [[PersistentDb](#)] section of the JADE initialization file.

Note The log files that can be viewed are text files that contain progress and any error information about each activity. These log files should not be confused with the transaction journal files, which are used for database recovery. (Transaction journals are non-text files named **dbnnnnnnnnnn.dump.txt**, with the *nnnnnnnnnn* value being a number in the range **1** through **999999999**), and should not be tampered with.)

By default, the JADE Database utility provides an internal viewer to display the log files in the JADE Database utility window. However, you can specify your own external viewer utility; for example, Notepad. For more details, see "[Setting Your JADE Database Utility Preferences](#)", in the following section. Use the **View Logs** command from the View menu to open a message log file in the default internal Log File View viewer or your specified external viewer.

» To open a new message log file

1. Select the **View Logs** command from the View menu. The common Open Activity Log dialog is then displayed. The initial directory that is displayed in the **Look in** combo box is one of the following values.
 - The value of the [ActivityLogDirectory](#) parameter in the [[PersistentDb](#)] section of the JADE initialization file
 - The value of the [LogDirectory](#) parameter in the [[JadeLog](#)] section of the JADE initialization file

If neither of those directories exists, the default value is the last directory in which an activity log file was opened (for example, **c:\jade\logs**).

2. Select the required message log file from the **File name** combo box and then click the **Open** button.

The selected log, shown in the following example of the Log File View of the Jade Database Utility window, is then displayed.

```
reindex.log - Notepad
File Edit Format View Help
<ReorgMgr::initialise>, mode=reindex, replayable
<<Attempting to achieve database quietpoint for reorg>>
checkpoint flush elapsed time=0 ms
checkpoint sync files elapsed time=0 ms
<<Check Point>> LSN=(38,32891629), minRecoveryLSN=(38,32891629), farthestBackLSN=(38,32891629), elapsed time=3 ms, by: reorg misc
<<[LockFilesForReorg]: Database reorg quietpoint achieved>>
File C:/Jade/system/_userscm.dat locked for reorg
Reorg start LSN(38,32891775)
ReorgJob=1 queued: readonly reindex of file: _userscm [31]
ReorgJob=1 commenced: readonly reindex of file: _userscm [31], map changes, updates disallowed

    commencing reorg of C:/Jade/system/_userscm.dat, operation: reindex - logical eof = 446611456
    339736 objects processed
    505198 subobjects processed
    0 errors
    old file size 446693376(446619648 used), new file size 447086592(446988288 used)
ReorgJob=1 completed: readonly reindex of file: _userscm [31], elapsed time = 00:00:05.392
renamed C:/Jade/system/_userscm.dat to C:/Jade/system/_userscm.bak
renamed C:/Jade/system/_userscm.reo to C:/Jade/system/_userscm.dat
_userscm [31] instantiated
Reorg validation in transaction 781, BT=(38,32892301)
_userscm [31] validating in transaction 781, BT=(38,32892301)
<ReorgMgr::initialise>, mode=reindex, replayable
<<Attempting to achieve database quietpoint for reorg>>
checkpoint flush elapsed time=0 ms
checkpoint sync files elapsed time=0 ms
<<Check Point>> LSN=(38,32892942), minRecoveryLSN=(38,32892942), farthestBackLSN=(38,32892942), elapsed time=3 ms, by: reorg misc
<<[LockFilesForReorg]: Database reorg quietpoint achieved>>
File C:/Jade/system/_userxrf.dat locked for reorg
Reorg start LSN(38,32893088)
ReorgJob=1 queued: readonly reindex of file: _userxrf [32]
ReorgJob=1 commenced: readonly reindex of file: _userxrf [32], map changes, updates disallowed

    commencing reorg of C:/Jade/system/_userxrf.dat, operation: reindex - logical eof = 53837824
    217093 objects processed
    29508 subobjects processed
    0 errors
    old file size 54001664(53878784 used), new file size 51642368(51585024 used)
ReorgJob=1 completed: readonly reindex of file: _userxrf [32], elapsed time = 00:00:00.812
renamed C:/Jade/system/_userxrf.dat to C:/Jade/system/_userxrf.bak
renamed C:/Jade/system/_userxrf.reo to C:/Jade/system/_userxrf.dat
_userxrf [32] instantiated
Reorg validation in transaction 782, BT=(38,32893614)
```

Note As there is one log file only for each operation, the current log is appended if you do not delete the log file between the same types of operation.

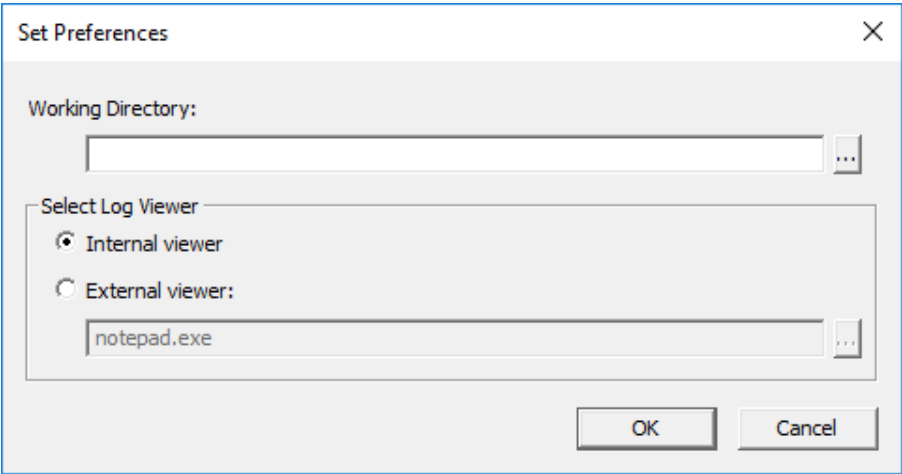
Setting Your JADE Database Utility Preferences

The **Preferences** command from the JADE Database utility View menu enables you to specify an alternative work directory and the type of viewer with which you want to display your log files. You can also specify a different font for your running display. (For details, see "[Selecting the Display Font](#)", earlier in this chapter.)

» To set your JADE Database utility preferences

1. Select the **Preferences** command from the View menu.

The Set Preferences dialog, shown in the following image, is then displayed.



2. Enter an alternative working directory in the **Working Directory** text box, if required.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the work directory that you require.

The JADE Database utility uses the working directory for the temporary files it created during the JADE Database utility operations; for example, during the compaction operation. If you do not specify a directory in this text box, it defaults to the database directory.

3. In the Select Log Viewer group box, select the **External viewer** option button if you do not want the log files displayed in an internal viewer. By default, log files are displayed in the internal viewer; that is, the **Internal viewer** option button is selected.
4. In the **External viewer** text box, specify the utility (for example, **Notepad.exe**) with which to display your JADE Database utility log files if you do not want to use the internal popup window.

Alternatively, click the adjacent browse button (indicated by the ... points of ellipsis symbol) to display the common Browse for Folder dialog that enables you to select the external viewer utility with which you want to view your log files.

5. Click the **OK** button to confirm your selections, or the **Cancel** button to discard your selections.

The JADE Database utility window is then displayed.

Specifying Your JADE Database Utility Edit Options

The Edit menu is displayed in the JADE Database utility only in the Log File View, to enable you to edit the log file that is currently displayed.

The Edit menu commands, described in the following subsections, are listed in the following table.

Command	Shortcut Key	Description
Undo	Ctrl+Z	Undoes the last action

Command	Shortcut Key	Description
Cut	Ctrl+X	Cuts (logically deletes) selected portion of text and copies it to the clipboard
Copy	Ctrl+C	Copies selected portion of text to the clipboard
Paste	Ctrl+V	Pastes portion of text from the clipboard to the current caret position
Delete	Del	Deletes the portion of text that is currently selected in the viewer
Find	Ctrl+F	Locates the specified text
Find Next	F3	Locates the next object in the current window that matches your search criteria
Replace		Locates the specified text in the current window and optionally replaces the text occurrence
Select All		Selects the entire log file displayed in the viewer
Word Wrap		Toggles the wrapping of text in the JADE Database utility Log File View

Edit menu commands that are not available for selection are disabled. For example, if you have not selected any text in the log file viewer, the **Copy**, **Cut**, and **Paste** commands are disabled, and cannot be selected.

Undo Command

Use the Edit menu **Undo** command to undo your last action in the viewer.

You can undo the following types of action.

- Last single keystroke
- Cut
- Paste

This command is disabled when a command cannot be undone; that is, you have not previously performed an action in the viewer that can be undone.

» To undo your last action, perform one of the following actions

- Select the **Undo** command from the Edit menu
- Press Ctrl+Z

Your last single action is then undone.

Note You can use this command to undo only the last action that you performed. You cannot undo a sequence of previous actions.

Cut Command

Use the Edit menu **Cut** command to logically delete a selected portion of text in the current viewer. This command is disabled when there is no selected text to be cut.

Select the portion of the text that you want to delete. The selected portion is then highlighted.

» To cut a selected portion of text, perform one of the following actions

- Select the **Cut** command from the Edit menu
- Press Ctrl+X

The selected portion is then removed from the current log file viewer.

When you use the **Cut** command, the selected portion of the text is copied to the clipboard. The clipboard is not saved between JADE Database utility sessions.

Copy Command

Use the Edit menu **Copy** command to copy a selected portion of text to another position in the current log file or to another log file. This command is disabled when there is no selected portion of the current log file to copy.

Select the portion of the text that you want to copy. The selected portion is then highlighted.

» To copy a selected portion of text, perform one of the following actions

- Select the **Copy** command from the Edit menu
- Press Ctrl+C

The selected portion is then copied to the clipboard. The clipboard is not saved between JADE Database utility sessions.

Paste Command

Use the Edit menu **Paste** command to paste a selected portion of text from the clipboard into the current log file. This command is disabled when there is no selected portion of text in the clipboard.

Note You can paste a portion of text to the same or to a different log file.

Before you can paste a portion or a complete log file to the same or a different log file, you must first have copied or pasted the required text to the clipboard.

» To paste text from the clipboard

1. Position the caret in the current log file at the position in which the paste operation is to begin.
2. Paste the text from the clipboard, by performing one of the following actions.
 - Select the **Paste** command from the Edit menu
 - Press Ctrl+V

The text from the clipboard is then pasted into the current log file, with the first part of the text located at the caret position.

When you use the **Paste** command, the text is copied from the clipboard. If you cut any text, any subsequent paste operations use the text from that cut. The clipboard is not saved between JADE Database utility sessions.

Delete Command

Use the Edit menu **Delete** command to delete a selected portion of text from the current log file.

This command is disabled when there is no selected portion of the current log file to delete.

Select the portion of the text that you want to delete. The selected portion is then highlighted. (You can delete the complete log file, if required, by first selecting the **Select All** command from the log file viewer Edit menu.)

» **To delete selected text, perform one of the following actions**

- Select the **Delete** command from the Edit menu
- Press DEL

The selected portion is then deleted.

Find Command

Use the Edit menu **Find** command to locate text in a log file.

» **To limit your search, perform one of the following actions**

- Select the part of the log file viewer contents in which the search is to be performed
- Specify that an exact case match must be performed
- Click the appropriate directional option button (that is, up or down)

» **To locate and optionally replace text in the log file viewer**

1. Select the **Find** command from the Edit menu. Alternatively, press Ctrl+F.
The Find dialog is then displayed.
2. Specify your search or replace options in the **Find What** text box, as appropriate.
3. If you want the exact match by case (where uppercase or lowercase is significant), check the **Match Case** check box.

A search is then performed for text with the same capitalization as the text in the **Find What** text box. By default, this option is not selected.
4. If you want to search backwards through the contents of the log file from the current caret position up to the beginning of the file, select the **Up** option button.

By default, the **Down** option button is selected (that is, the search is performed from the current position to the end of the log file.)
5. Click on the **OK** button.

Alternatively, click the **Cancel** button to abandon your selection, or click the **Help** button to obtain online help about this dialog.

To find the next occurrence of the text specified in the **Find What** text box, click the **Find Next** button.

If JADE finds the text string that matches your specified options, the located text is then highlighted, and focus is returned to the log file viewer.

If JADE cannot find the text string that matches your specified options, a message dialog informs you that the search text was not found, and waits for you to click the **OK** button in the message dialog, before returning focus to the log file viewer.

Find Next Command

Use the Edit menu **Find Next** command to locate the next occurrence of text that you specified in the Find dialog (by using the **Find** command from the Edit menu).

» To find the next occurrence of specified text, perform one of the following actions

- Select the **Find Next** command from the Edit menu.
- Press F3.

If JADE finds the text string that matches the value that you specified in the **Find What** text box of the Find dialog, the located text is then highlighted in the log file viewer.

Replace Command

Use the Edit menu **Replace** command to locate text in a log file, and replace the located text with a specified value.

» To limit your search, perform one of the following actions

- Select the part of the log file viewer contents in which the search is to be performed
- Check the **Match Case** check box

» To replace text in the editor pane

1. Select the **Replace** command from the Edit menu. The Replace dialog is then displayed.
2. Specify your search and replace options, as appropriate.
3. Check the **Match Case** check box if you want an exact match performed.
4. Click on the **OK** button.

Alternatively, click the **Cancel** button to abandon your selection, or click the **Help** button to obtain online help about this dialog.

» To find the next occurrence of the text specified in the Find What text box

- Click the **Find Next** button

If JADE finds the text string that matches your specified options, the located text is then highlighted, and focus is returned to the log file viewer.

If JADE cannot find the text string that matches your specified options, a message dialog informs you that the search text was not found, and waits for you to click the **OK** button in the message dialog, before returning focus to the log file viewer.

» To replace the occurrences of the specified text that is currently highlighted in the log file viewer

- Click the **Replace** button

» To replace all occurrences of the text specified in the Find What text box with the text specified in the Replace With text box

- Click the **Replace All** button

If JADE finds the text string that matches all of your specified options, all occurrences of the located text are then replaced with the specified replacement text, and focus is returned to the log file viewer.

Select All Command

Use the Edit menu **Select All** command to select the complete contents of the current log file viewer. For example, you can select a complete log file to copy it to the clipboard before pasting it into another log file or an editor.

» **To select the contents of the log file viewer**

- Select the **Select All** command from the Edit menu

The contents of the current log file viewer are then selected; that is, highlighted.

Word Wrap Command

Use the Edit menu **Word Wrap** command to automatically wrap text to the beginning of the next line when the text reaches the right edge of the log file viewer.

This does not affect your text file, which retains the full line of text.

» **To set word wrapping**

- Select the **Word Wrap** command from the Edit menu when the log file viewer is displayed

A check mark symbol is displayed to the left of the **Word Wrap** command in the Edit menu when this feature is set. Word wrapping remains in effect until you select this command again.

If word wrapping is not set (and no check mark is displayed), you may be required to use the horizontal scroll bar to view all text that is displayed in the JADE Database utility log file viewer.

Specifying Your Window Display

The JADE Database utility Window menu provides standard facilities to select tiling or cascading of windows. It also displays a list of the currently enabled windows.

The lower portion of the Window menu lists all currently open windows in the order in which they were opened. The current, or active, window is indicated by a check mark symbol to the left of the window number and name.


Tip To access an open window, select the appropriate window from the list in the Window menu. Alternatively, you can use the Ctrl+F6 shortcut keys to cycle through all open MDI child windows.

Use the Window menu commands in the JADE Database utility to perform the actions listed in the following table.

Command	Description
New Window	Opens a new JADE Database utility window
Cascade	Arranges open windows in an overlapping pattern
Tile Horizontally	Resizes and arranges windows horizontally without overlap
Tile Vertically	Resizes and arranges windows vertically without overlap
Arrange Icons	Arranges all minimized icons in an orderly manner

Obtaining JADE Database Utility Online Help

Use the Help menu in the JADE Database utility window to perform the actions listed in the following table.

Command	Toolbar Button	Description
Contents		Accesses the <i>JADE Database Administration Guide</i> in the Adobe Reader window
About jdbutil		Displays version information for the JADE Database utility

For details, see "[Accessing the Online Help Index](#)" and "[Accessing JADE Database Utility Release Information](#)", in the following subsections.

Accessing the Online Help Index

Use the JADE Database utility Help menu **Contents** command to access the *JADE Database Administration Guide*, which provides access to the topics available in online help.

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

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

The JADE online help is then displayed; for example, the **DbAdmin.pdf** document is displayed in Adobe Reader.

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 JADE Database Utility Release Information

Use the JADE Database utility **About jdbutil** command to access information (for example, version information) about the current release of the JADE Database utility.

» **To access the JADE Database utility version information**

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

Running the JADE Database Utility in Batch Mode

The **jdbutilb** program, installed with your JADE software, enables you to automate the JADE Database utility by running it in batch mode. You can use this batch mode utility, for example, to automate the process of resetting timestamps when replacing database files in a runtime deployment environment, or to initiate an offline backup of selected database files.

Run the batch JADE Database utility (for example, from a command script), specifying the following.

```
jdbutilb path=database-path
         ini=initialization-file-name
         [backup [file-list] backupDir=directory [compress] [overwrite]
           [excludeFrozenFiles] [excludeFrozenPartitions] [verify]
           [nostatus]]
         [backupJournal journal=number backupDir=directory
           journalDir=directory [lastJournal=number] [compress]
```

```

[overwrite] [verify] [nostatus]]
[certify [file-list] [nostatus]]
[changeFilePaths backupinfoPath=directory file-name
[part=partition-id]
[newBackupPath=directory|<default>]|
[newDatabasePath=directory|<default>] [nostatus]]
[clearDeltaMode]
[clearFilePath [file-list]]
[clearFilePathAudited [file-list]]
[clearRole]
[commandFile=command-file-name]
[compact [workers=number-of-worker-threads] [file-list] [nostatus]]
[convertToBackup [workers=number-of-worker-threads]]
[delete [file-list]]
[dumpJournal journal=number [journalDir=directory] [nostatus]
[lastJournal=number]]
[fileAttributes [file-list] initSize=Kbytes extentSize=Kbytes|
initSize=Kbytes|extentSize=Kbytes]
[freeSpace [file-list] [nostatus]]
[freezeSchemaFiles]
[generateEnvironmentIdentity]
[generateServerIdentity]
[generateUUID]
[journalingRates journal=number [journalDir=directory] [nostatus]
[lastJournal=number] [samplingInterval=minutes]]
[listBackupinfoFileNames]
[listFiles]
[listPartitions file-name]
[makeBackupinfo baseDir=directory deltaDir=directory
outputDir=directory]
[mapFile [file-list]]
[markOffline [file-list]]
[markOnline [file-list]]
[productionMode=true|false]
[recover [recoverTo=date time|lastJournal=journal-file-number
|lastSerialNumber=audit-record-number] [clearSDSRole]
[nostatus]]
[reindex [file-list] [nostatus]]
[resetTS [file-list]]
[restore backupDir=directory [noRecovery] [verify]
[recoverTo=date time|lastJournal=number
|lastSerialNumber=number] [clearSDSRole]
[clearFilePaths] [excludeFrozenFiles] [excludeFrozenPartitions]
[nostatus]]
[restoreFile backupDir=directory file-list [verify] [clearFilePaths]
[excludeFrozenFiles] [excludeFrozenPartitions] [nostatus]]
[restoreJournal backupDir=directory journal=number [nostatus]
[lastJournal=number]]
[restorePartitionFile backupDir=directory file-name
part=partition-id [verify] [clearFilePaths] [nostatus]]
[setFilePath fileDir=directory [file-list]]
[setFilePathAudited fileDir=directory [file-list]]
[setUserFileVersions version=version-number [file-list]]
[showInfo]
[thawSchemaFiles]
[touchDB]

```

```
[unmapFile [file-list]]
[verifyChecksums [workers=number-of-worker-threads]]
[verifyJournal journal=number [journalDir=directory] [nostatus]
[lastJournal=number]]
[help]
```

You can specify only one of the **certify**, **compact**, **delete**, **freeSpace**, **reindex**, or **resetTS** database operations at a time, but you can specify a file list for the operation that is executed if you do not want all database files processed. (Each file in a list is space-separated, and is specified with the data file prefix only.)

Tip If you run the batch JADE Database utility regularly and you use the same or similar commands, you can use the **commandFile** operation to specify a text file (for example, a **.txt** or a **.cmd** file) that contains a list of the commands and their arguments, with each command on a separate line in the command file.

You then need only to specify the program name, the operation, and the name of the command file. As JADE does not look for any other command when it detects the **commandFile** operation, ensure that you specify this as the last command in the command line.

Simple file masks are supported; for example, **_user*** processes all file names containing the **_user** prefix.

The following is an example of the command for the **certify** operation of the **_rootdef.dat**, **_userscm.dat**, **_userxrf.dat**, **_usergui.dat**, **_userint.dat**, **_userdev.dat**, **mydata.dat**, and **shrdemo.dat** files, with the status message display suppressed.

```
jdbutilb path=d:\salesdb ini=d:\salesdb\jade.ini clearDeltaMode certify _user*
mydata shrdemo nostatus
```

In the following examples, the production (primary) offline backup is loaded to a test machine with a different configuration. The database is restored using default locations and a large file is then relocated to another drive.

```
jdbutilb path=e:\test\system restore ini=e:\test\jade.ini
backupDir=g:\overnightOffline verify noRecovery clearSDSRole clearFilePaths

jdbutilb path=e:\test\system ini=e:\test\jade.ini setFilePath
fileDir=f:\test\system nameOfaBigFile
```

In the following example, a secondary system has had more disk added so some files of employee data are moved to a new drive.

```
jdbutilb path=e:\ProdB\system ini=e:\ProdB\jade.ini setFilePath
fileDir=f:\ProdB\system emp*
```

In the following example, a production system file is moved to a new drive, using the unaudited command so that it is not re-done in roll-forward recovery.

```
jdbutilb path=e:\ProdA\system ini=e:\ProdA\jade.ini setFilePath
fileDir=f:\ProdA\system AcctHist
```

Batch database utility program operations that consist of a number of steps or that take longer than a few seconds (for example, **backup**, **certify**, or **restore**) can optionally display a progress report of each specified operation while that operation is running, unless status output is suppressed. (For details, see "**nostatus**", later in this chapter.)

For details about the parameters that provide advanced diagnostic capabilities when performing certify and freespace evaluation operations, see "**Database Diagnostics Section [DBUtil]**", in the *JADE Initialization File Reference*.

Standard database operation information is output to **stdout** and error information is output to **stderr**. For details about displaying and redirecting the output from JADE batch utilities, see the [DisplayApplicationMessages](#), [LogServer](#), and [UseLogServer](#) parameters under "JADE Log Section [[JadeLog](#)]", in the *JADE Initialization File Reference*.

At the end of the operation, the running display provides a completion report, highlighting the number of files processed and the number of files found with error.

Detailed results of the certification, compaction, free-space evaluation, and checksum verification operations are written to the appropriate log file in your specified database directory.

If the **jdbutlib** executable program fails, a non-zero exit code is returned and an error message is displayed; for example, if the database directory was invalid.

Batch Database Utility Arguments

The batch JADE database utility command line argument values that apply to one or more batch JADE Database utility commands are described in the following subsections. (For details about the commands with which these arguments can be used, see ["Batch Database Utility Commands"](#), later in this chapter.)

backupDir

The **backupDir** argument specifies the destination directory for backup operations specified using the [backup](#) or [backupJournal](#) command, or the source directory for restore operations specified using the [restore](#), [restoreFile](#), [restoreJournal](#), or [restorePartitionFile](#) command.

For restore operations, you must specify an existing directory that is valid on the server. For backup operations, the directory is created if it is not present.

This argument must be specified if the **jdbutlib** command line contains the [backup](#), [backupJournal](#), [restore](#), [restoreFile](#), [restoreJournal](#), or [restorePartitionFile](#) command.

backupinfoPath

The **backupinfoPath** argument specifies the full path of the restore source (that is, backup) directory location or the **backupinfo** file that is to be changed by using the [changeFilePaths](#) command, following a backup operation and prior to a restore operation.

This argument must be specified if the **jdbutlib** command line contains the [changeFilePaths](#) command.

baseDir

The **baseDir** argument specifies the directory in which the base (full) **backupinfo** database file that is to be merged with a partial **backupinfo** file is located.

You must specify an existing directory that is valid on the server; the directory is not created if it is not present.

This argument must be specified if the **jdbutlib** command line contains the [makeBackupInfo](#) command.

clearFilePaths

The optional **clearFilePaths** argument enables you to specify that explicit map file settings (that is, hard-coded map paths) are overridden for restore operations when using the [restore](#), [restoreFile](#), or [restorePartitionFile](#) command.

When you specify the **clearFilePaths** argument, all user file paths in the control file and in the partition control files are set to **<null>** during database initialization.

When you specify the **clearFilePaths** argument for the **restoreFile** or **restorePartitionFile** command, it has an effect only when that restore file is processing the **_control.dat** file.

clearSDSRole

The optional **clearSDSRole** argument enables you to specify that the database recovery mechanism that performs a roll-forward recovery of a backed up SDS system, clears the SDS role of that database during the recovery process when using the **recover** or **restore** command on the **jdbutlib** command line.

When you specify the **clearSDSRole** argument, at the end of roll-forward recovery, a new database identifier is generated and the database role is set to undefined, without regard to the prior value of the database role.

Note Attempts to perform a roll-forward recovery of a backup of an SDS secondary system with termination conditions without this option enabled are disallowed. If you do *not* specify the **clearSDSRole** argument in the recovery from backup, the recovery operation is recovering a secondary database as a secondary. This means that:

1. No undo will be performed
2. A partial final journal will not be processed (when the secondary initializes, it re-fetches the journal from the primary)

The expectation is that the database will connect to the primary and then synchronize. When the last journal is not applied when recovering from a backup, the backup recovery is incomplete and file integrity is not restored. The undo action is suppressed, so there are further complications if there are incomplete transactions.

compress

The optional **compress** argument enables you to specify that data compression is invoked to compress data as it is copied to the backup when using the **backup** or **backupJournal** commands. (When you specify this argument, data is compressed on the fly as it is copied.)

deltaDir

The **deltaDir** argument specifies the directory in which the latest partial (delta) **backupinfo** database file that is to be merged with the base **backupinfo** file is located. You must specify an existing directory that is valid on the server; the directory is not created if it is not present.

This argument must be specified if the **jdbutlib** command line contains the **makeBackupInfo** command.

excludeFrozenFiles

The optional **excludeFrozenFiles** argument enables you to specify that frozen (read-only) files are excluded from the backup or restore operation when using the **backup**, **restore**, or **restoreFile** command.

When you specify the **excludeFrozenFiles** argument for the **restoreFile** command, it has an effect only when that restore file is processing the **_control.dat** file.

excludeFrozenPartitions

The optional **excludeFrozenPartitions** argument enables you to specify that frozen (read-only) partitions are excluded from the backup or restore operation when using the **backup**, **restore**, or **restoreFile** command.

extentSize

The optional **extentSize** argument of the **fileAttributes** command enables you to specify the number of bytes that you require for the growth increment extent of database files; for example, explicitly (**1048576**) or by using a prefix multiplier (**1M**).

The default value is **128K** and the minimum value is **64K**.

You can specify both this argument and the **initSize** argument (separated by a space), the **extentSize** argument only, or the **initSize** argument only.

fileDir

The **fileDir** argument specifies the directory to which files are moved using the **setFilePath** command or **setFilePathAudited** command.

This argument must be specified if the **jdbutlib** command line contains the **setFilePath** command or **setFilePathAudited** command.

file-list

The optional **file-list** argument specifies a list of selected files on which to perform the database action; for example, when using the **mapFile**, **markOffline**, **markOnline**, **setFilePath**, **setFilePathAudited**, or **setUserFileVersions** command.

When specifying a file list:

- Each file in a list is space-separated.
- Specify only the file prefix; that is, do not include the file suffix.
- Simple file masks are not supported; for example, a file list of **_user*** is invalid.

file-name

The **file-name** argument specifies the name of the database file that is to be processed by the **changeFilePaths** or **restorePartitionFile** command.

This argument must be specified if the **jdbutlib** command line contains the **changeFilePaths** or **restorePartitionFile** command.

ini

The **ini** argument specifies the fully qualified name of your JADE initialization file if it is not located in the database directory or it has a file name other than the default **jade.ini**.

Tip Specify the fully qualified name of an existing JADE initialization file in the command line.

initSize

The optional **initSize** argument of the **fileAttributes** command enables you to specify the number of bytes that you require for the initial size of database files; for example, explicitly (**524288**) or by using a prefix multiplier (**512K**).

The default value is **128** and the minimum value is **64K**.

You can specify both this argument and the **extentSize** argument (separated by a space), the **initSize** argument only, or the **extentSize** argument only.

journalDir

The optional **journalDir** argument specifies the directory in which the recovery log file is located if it is not in the current log directory for the JADE database. You must specify an existing directory that is valid on the server. Specify this argument when using the **dumpJournal**, **journalingRates**, or **verifyJournal** command.

journal

The **journal** argument specifies the transaction journal file that is backed up, restored, or verified. This argument must be specified if the **jdbutlib** command line contains the **backupJournal**, **dumpJournal**, **journalingRates**, **restoreJournal**, or **verifyJournal** command.

lastJournal

The optional **lastJournal** argument specifies the number of the last transaction journal file on which the operation is to be performed; for example, **9** (if you want to perform an operation on transaction journals up to and including **db0000000009.log**).

You can specify this argument if the **jdbutlib** command line contains the **backupJournal**, **dumpJournal**, **journalingRates**, **restoreJournal**, or **verifyJournal** command or instead of the optional **recoverTo** or **lastSerialNumber** argument if the **jdbutlib** command line contains the **recover** or **restore** command.

lastSerialNumber

The optional **lastSerialNumber** argument specifies the serial number of the last audit record to which committed transactions are recovered or restored.

You can specify this argument instead of the optional **lastJournal** or **recoverTo** argument if the **jdbutlib** command line contains the **recover** or **restore** command.

newBackupPath

The **newBackupPath** argument specifies the new restore source (that is, backup) location of a file or a file partition in the **backupinfo** file when you change the backup path prior to a restore operation.

When you specify this argument, its value is checked for validity (that is, it must exist).

This argument or the **newDatabasePath** argument must be specified if the **jdbutlib** command line contains the **changeFilePaths** command.

newDatabasePath

The **newDatabasePath** argument specifies the new restore destination (that is, database) location of a file or a file partition in the **backupinfo** file when you change the database path prior to a restore operation.

When you specify this argument, its value is checked for validity only if it is absolute, but it does not have to exist; that is, it is created if it does not exist.

This argument or the **newBackupPath** argument must be specified if the **jdbutlib** command line contains the **changeFilePaths** command.

noRecovery

The optional **noRecovery** argument specifies that no roll-forward recovery is to be performed when the database is restored by using the **restore** command.

nostatus

The optional **nostatus** argument enables you to suppress the status (progress) message display.

Note Full details of the operation are still output to the appropriate log file.

outputDir

The **outputDir** argument specifies the directory in which the base **backupinfo** database file and the latest partial **backupinfo** database file are to be merged. You must specify an existing directory that is valid on the server. The directory is created if it is not present.

This argument must be specified if the **jdbutlib** command line contains the **makeBackupInfo** command.

overwrite

The optional **overwrite** argument enables you to specify that any existing files in your destination directory are overwritten when backing up your transaction journals or database.

part

The **part** argument specifies the identifier of the database file partition that you want to process using the **changeFilePaths** or **restorePartitionFile** command.

This argument must be specified if the **jdbutlib** command line contains the **changeFilePaths** or **restorePartitionFile** command.

path

The **path** argument specifies the full path of your default JADE database directory in which the control file (**_control.dat**) is located; for example:

```
d:\jade\system
```

This argument must be specified if the **jdbutlib** command line contains any command other than the **changeFilePaths** command.

recoverTo

The optional **recoverTo** argument specifies the date and time to which committed transactions are recovered when using the **recover** or **restore** command. Specify the required end date and time in the following formats.

```
dd\mm\yyyy
```

```
HH:MM:SS
```

You can specify this argument instead of the optional **lastJournal** or **lastSerialNumber** argument if the **jdbutlib** command line contains the **recover** or **restore** command.

samplingInterval

The optional **samplingInterval** argument specifies the number of minutes at which journal rates are sampled. You can specify this argument if the **jdbutlib** command line contains the **journalingRates** command.

If you do not specify this argument, journaling rates are sampled every minute (that is, the default value is **1**, or 60,000 milliseconds).

verify

The optional **verify** argument enables you to specify that data is verified as it is copied to backup for database file or transaction journal backups when using the **backup** or **backupJournal** command.

Note that when using the **restore**, **restoreFile**, or **restorePartitionFile** command, the optional **verify** argument enables you to specify that a verify checksum operation is performed in-line during the restore operation. Each file restore operation additionally processes the file data written to the destination database directory to verify that the checksum matches the source data checksum.

Tip When you restore multiple map files concurrently for a large database and you specify the **verify** argument, you do not have to wait until all map files are restored and then perform a checksum verification by executing the **verifyChecksums** command.

version

The **version** argument of the **setUserFileVersions** command enables you to specify the version number to assign to all user data map files or to the user data map files specified in the optional *[file-list]* argument value.

The default value is zero (**0**); that is, no version number is assigned.

workers

The optional **workers** argument enables you to specify the number of multiple concurrent worker threads that are used to verify the database and generate a **backupinfo** file when using the **compact**, **convertToBackup**, or **verifyChecksums** command.

The default value of **2** is used if you do not specify the **workers** argument. If you specify a value greater than **16**, the maximum number of 16 worker threads is used.

Batch Database Utility Commands

The batch JADE Database utility commands are described in the following subsections. For details about the arguments that can apply to these commands, see "[Batch Database Utility Arguments](#)", earlier in this chapter. For a description of the effects of each of these commands, see the corresponding functional descriptions earlier in this chapter under "[Using Database Operational Commands](#)".

backup

The **backup** command initiates an offline backup of all database files or the selected database files specified in the optional *[file-list]* argument. Information or errors from the backup operation are recorded in the **backup.log** file located in the default database directory (specified by using the **path** argument).

Caution You cannot restore the backup if errors are encountered during the backup process.

The syntax of the **backup** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         backup [file-list]
         backupDir=backup-directory
         [compress]
         [excludeFrozenFiles]
         [excludeFrozenPartitions]
         [overwrite]
         [verify]
         [nostatus]
```

The following example shows the use of the **backup** command in the **jdbutilb** command line to backup all user schema files and the **userdata1.dat** and **userdata2.dat** files, with verification requested and the status message display suppressed.

```
jdbutilb path=d:\jade\db backup ini=d:\jade.ini _user* userdata1 userdata2
         backupDir=e:\jade\backup25032004 verify nostatus
```

In this example, the backup process expects to find the database control file (**_control.dat**) in the **d:\jade\db** database path specified in the **path** argument. The files will be backed up to the **e:\jade\backup25032004** directory specified in the **backupDir** argument.

backupJournal

The **backupJournal** command initiates an offline backup of the transaction journal file specified in the **journal** argument.

Information or errors from the backup operation are recorded in the **backup.log** file located in the default database directory (specified by using the **path** argument).

Caution If errors are encountered during the backup log process, the backup is removed.

The syntax of the **backupJournal** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         backupJournal
         backupDir=backup-directory
         journalDir=journal-directory
         journal=journal-file-number
         [lastJournal=journal-file-number]
         [compress]
         [overwrite]
         [verify]
         [nostatus]
```

The following example shows the use of the **backupJournal** command in the **jdbutilb** command line to backup a specific transaction journal file with verification and data compression requested.

```
jdbutilb path=d:\jade\db backupDir=\\backupserver\jade\jnlbackups\07aug2011
         ini=d:\jade\jade.ini backupJournal journalDir=s:\jade\system\logs\current
         journal=45 verify compress nostatus
```

In this example, the journal backup expects to find the control file in the **d:\jade\db** directory, and the selected journal file, located in the **s:\jade\system\logs\current** directory, will be backed up in the **\\backupserver\jade\jnlbackups\07feb2007** directory.

certify

The **certify** command enables you to perform a read-only check on the integrity of all database files or selected files specified in the optional [\[file-list\]](#) argument. Specify the optional **nostatus** argument if you do not want the status message displayed during the certify operation.

When you specify the **certify** command, you cannot also specify the **compact**, **freeSpace**, **reindex**, or **resetTS** command in the **jdbutilb** command line.

The syntax of the **certify** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        certify [file-list]
        [nostatus]
```

For details, see ["Database Certification"](#) and ["Using the Certify Files Command"](#), elsewhere in this document.

changeFilePaths

The **changeFilePaths** command changes the backup path attribute and the database path attribute of a file or a file partition in the **backupinfo** file. A JADE database backup is a disk backup and is described by a **backupinfo** file that contains information about all database files. By default:

- Database files are located in the same directory as the database control file (**_control.dat**)
- Database file partitions by default are located in the same directory as the partition control file (**file-name.dat**)
- Backup files are located in the same directory as the backup control file (**backupinfo**)
- Backup file partitions are located in the same directory as the backed-up partition control file

A file or file partition has a backup path attribute describing its location within the disk backup (used by **verifyChecksums** and **restore** operations), and a database path attribute containing its database path specification as copied from the database control file (used by the **restore** operation). For an example of changing file paths, see ["File Partition Location Modification Example"](#).

When backing up a database, it can be necessary to specify some non-default file or file partition backup paths; for example, when some files or file partitions are too big for the default location. When absolute paths are specified, that backup can be placed only on a machine that has devices matching the path specifications, and the files and file partitions with non-default backup paths have to be correctly placed to reconstitute the disk backup; otherwise the **verifyChecksums** and **restore** operations will fail.

Similarly, for operational reasons, a live database can have some non-default file or file partition database paths in use. When absolute paths are used, a backup of that database can be restored only on a machine that has devices matching the path specifications; otherwise the **restore** operation will fail.

The syntax of the **changeFilePaths** command in the **jdbutilb** command line is as follows.

```
jdbutilb changeFilePaths file-name
        backupinfoPath=directory
        ini=initialization-file-name
        [part=partition-id]
        [newBackupPath=directory|<default>]
        [newDatabasePath=directory|<default>]
        [nostatus]
```

Note The **path** argument is not used by this command, so it need not be specified. You can specify one or both of the **newBackupPath** and the **newDatabasePath** arguments.

The value of the **newBackupPath** argument is checked for validity and must exist (as it is where the file is located). The value of the **newDatabasePath** argument is checked for validity only if it is absolute, but it does not have to exist (the **restore** operation will create it, if necessary).

You can specify a **file-name** of * (the asterisk symbol), to cause the arguments to be applied to all files and partitions in the **backupinfo** file. For example, setting **newDatabasePath="<default>"** and then restoring the database would result in the same outcome for that restored database as specifying the **clearFilePaths** argument in the **restore** command.

File Partition Location Modification Example

In this example, the database is located in **F:\prod\server\c_system** but due to size constraints, partitions 1 through 4 of the partitioned file **Invoices** are located in **K:\prod\server\c_system**. The disk backup was made to **H:\backup**. Due to its size, the partitioned file **Invoices** was backed up to **G:\backup**.

You need to load this database backup onto another server environment (perhaps the system is being relocated or you are making a copy for testing).

The new server environment has an **F:** drive, does not have a **G:** drive or a **K:** drive, has room on its **H:** drive for the entire backup, and has a large **N:** drive to which some backup files could temporarily be loaded and where partitions 1 through 4 of the partitioned file **Invoices** of the live database should be placed.

In this scenario, after loading the backup (to **H:\backup** and perhaps **N:\temp**), the steps that you would take to change the backup file location and destination file partition locations are as follows.

1. Change the backup path for the file **Invoices** by running the **changeFilePaths** command and specifying the backup location of the file in the **newBackupPath** argument; for example:

```
jdbutilb ini=F:\prod\server\jade.ini changeFilePaths backupinfoPath=H:\backup
Invoices newBackupPath="<default>"
```

This action clears the non-default backup path attribute of the file **Invoices** in the **backupinfo** file.

The following example changes the backup location of this file in the **backupinfo** file to **N:\temp**.

```
jdbutilb ini=F:\prod\server\jade.ini changeFilePaths backupinfoPath=H:\backup
Invoices newBackupPath=N:\temp
```

Notes This backup path location must exist.

You must enclose the **<default>** value within double quote symbols.

2. When the backup paths in the **backupinfo** file have been made to reflect the actual file locations, perform the **verifyChecksums** operation to ensure that errors have not been injected during the file transfers.
3. Change the database location of partitions 1 through 4 of the file **Invoices** by running the **changeFilePaths** command and specifying the database location for each partition in the **newDatabasePath** argument.

The following example changes the database location of partition 1 in the **backupinfo** file to **N:\prod\server\c_system**. (Repeat the operation for partitions 2 through 4.)

```
jdbutilb ini=F:\prod\server\jade.ini changeFilePaths backupinfoPath=H:\backup
Invoices part=1 newDatabasePath=N:\prod\server\c_system
```

4. Perform the **restore** operation to **F:\prod\server\c_system**.

During database initialization, changed file and file partition path information in the **restoreinfo** file is adopted and the database control file and partition control files are updated as appropriate.

clearDeltaMode

The **clearDeltaMode** command enables you to clear the delta mode state preserved in the database control file. This may be necessary if, for any reason, it is not possible to deactivate delta mode online. A delta database allows non-permanent (or tentative) changes to be made to a database in read-only mode.

Note As the **clearDeltaMode** action is not audited, it is not replayable for archival recovery purposes.

The syntax of the **clearDeltaMode** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         clearDeltaMode
```

The following example shows the use of the **clearDeltaMode** command in the **jdbutilb** command line to remove any delta mode restrictions.

```
jdbutilb path=d:\salesdb ini=d:\salesdb\jade.ini clearDeltaMode
```

clearFilePath

The **clearFilePath** command enables you to clear the path of all database files or selected files specified in the optional *[file-list]* argument so that you can relocate files on secondary systems and in test environments.

Note As the **clearFilePath** action is not audited, it is not replayable for archival recovery purposes.

The syntax of the **clearFilePath** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         clearFilePath
         [file-list]
```

clearFilePathAudited

The **clearFilePathAudited** command enables you to clear the path of all database files or selected files specified in the optional *[file-list]* argument so that you can relocate files on secondary systems and in test environments.

Note As the **clearFilePathAudited** action is audited, it can be replayed for archival recovery purposes (if you want the operation to be redone in roll-forward recovery). SDS Secondary replay does not replay these audit records.

The syntax of the **clearFilePathAudited** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         clearFilePathAudited [file-list]
```

clearRole

The **clearRole** command enables you to clear the primary database role and create a non-SDS-enabled database. The syntax of the **clearRole** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         clearRole
```

Clearing the primary database role removes any SDS-enforced restrictions.

commandFile

The **commandFile** command enables you to specify the name of a text file (for example, a **.txt** or a **.cmd** file) that contains a line for each JADE Database utility command.

When you run the **jdbutilb** program and JADE detects the **commandFile** command, it opens the specified text file and reads each line, performing the defined actions on those lines.

Tip If you run the batch JADE Database utility regularly and you use the same or similar commands, this command means that you do not have to enter the same or similar information in the command line each time but can simply specify the program, the **commandFile** command, and the name of the existing command file (and its relative path name, if required).

As JADE does not look for any other command when it detects the **commandFile** operation, ensure that you specify this as the last command in the command line.

The syntax of the **commandFile** command in the **jdbutilb** command line is as follows.

```
jdbutilb [commandFile=command-file-name]
```

You can include the mandatory **path** argument in the command file, as shown in the following example of a text file specified as **jdbutilb commandFile=d:\jade\files\batchutil.txt**.

```
path=d:\jade\db
backupDir=\\backupserver\jade\jnlbackups\07feb2007 verify nostatus
backupJournal journalDir=s:\jade\system\logs\current journal=45 compress
backup _user* userdata1 userdata2
certify _user* mydata shrdemo nostatus
verifyChecksums
```

compact

The **compact** command enables you to run a compaction routine on all database files or selected files specified in the optional **[file-list]** argument.

If you want to compact multiple database files in parallel, specify the optional **workers** argument and the number of workers (greater than 1) that you require. If you do not specify the number of workers or you specify a number less than two workers, serial compact actions are performed.

Specify the optional **nostatus** argument if you do not want the status message displayed during the compaction operation.

When you specify the **compact** command, you cannot also specify the **certify**, **delete**, **freeSpace**, **reindex**, or **resetTS** command in the **jdbutilb** command line.

The syntax of the **compact** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         compact [workers=number-of-workers] [file-list]
         [nostatus]
```

For details, see "[Compacting Files](#)" and "[Using the Compact Files Command](#)", elsewhere in this document.

convertToBackup

The **convertToBackup** command, which has the following syntax, verifies the integrity of the database, computes file checksums, and stores these in **restoreinfo** and **backupinfo** files. The end result is a dual-purpose backup and "restored backup" image. The **restoreinfo** file can be used by the [verifyChecksums](#) command to verify that files are intact in the original position. The **backupinfo** file allows the backup image to be restored to a different location using existing database restore mechanisms.

```
jdbutilb path=database-path
         ini=initialization-file-name
         convertToBackup [workers=number-of-worker-threads]
```

The **convertToBackup** operation can be performed when the database image has been recovered to the end of a journal that contains in-progress transactions. Note that it is not valid to run the **convertToBackup** operation on a database image:

- That is in a crashed state requiring restart recovery
- Where the class file mappings are inconsistent, which could happen if a step-wise recovery terminated during a schema instantiation

If you have specified a value for the [LogDirectory](#) parameter in the [\[JadeLog\]](#) section of the JADE initialization file that you want to use, use the optional **ini** argument in the **jdbutilb** command to locate this setting.

Specify the optional **workers** argument if you want to use a single worker thread (1) or more than the default two (2) multiple concurrent worker threads. The **convertToBackup** operation can use a maximum of 16 multiple concurrent worker threads.

The following example shows the use of the **convertToBackup** command.

```
jdbutilb path=s:\jade\system ini=s:\jade\backups\jade.ini convertToBackup workers=3
```

This verifies the database and creates **restoreinfo** and **backupinfo** files that contain MD5 file checksums in the database directory specified in the **path** argument. Operational logging is output to the **checksum.log** file that is also located in the database directory specified in the **path** argument.

If the operation is successful, you can then perform the required action with the specified database directory (for example, you can copy the files to tape).

To verify loaded database files that have been copied back from another medium, execute the **jdbutilb** program as follows.

```
jdbutilb path=database-path ini=initialization-file-name verifyChecksums
```

The following example shows the use of the [verifyChecksums](#) command.

```
jdbutilb path=s:\jade\system ini=s:\jade\backups\jade.ini verifyChecksums
```

For more details, see "[Custom Backup Support](#)" under "[Backups](#)", in Chapter 3.

delete

The **delete** command enables you to delete test data or unwanted data in all user database files or selected files specified in the optional *[file-list]* argument.

If you delete a test database file and you subsequently start creating new instances of the data, the test database file is recreated.

Note You can delete user database files only; you cannot delete system database files.

When you specify the **delete** command, you cannot also specify the **certify**, **compact**, **freeSpace**, **reindex**, or **resetTS** command in the **jdbutilb** command line.

The syntax of the **delete** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         delete [file-list]
```

For details, see "[Using the Delete Files Command](#)", earlier in this chapter.

dumpJournal

The **dumpJournal** command produces a formatted dump of the transaction journal file specified using the **journal** argument.

The syntax of the **dumpJournal** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         dumpJournal
         journal=journal-file-number
         [lastJournal=journal-file-number]
         [journalDir=journal-directory]
         [nostatus]
```

The following example shows the use of the **dumpLog** command in the **jdbutilb** command line to dump the transaction journal number **0000076894**.

```
jdbutilb path=s:\testjade\db\logs\current ini=s:\testjade\jade.ini dumpJournal
journal=0000076894 journalDir=s:\testjade\logs\current
```

The following dump file is formatted using this example.

```
d:\testjade\db\logs\current\db0000076894.dump.txt
```

For more details, see "[Using the Dump Journals Command](#)", earlier in this chapter.

fileAttributes

The **fileAttributes** command enables you to set the initial file size and the extent size for selected database files. The attribute values are saved in the database control file and non-zero values override the **DefFileGrowthIncrement** and **DefInitialFileSize** parameter values configured in the **[PersistentDb]** section of the JADE initialization file.

If you want to clear the override for a specific file, set the value to zero (**0**), which means the default JADE initialization file parameter is used.

Specify the **extentSize** argument, the **initSize** argument, or both the **extentSize** argument and the **initSize** arguments. Set these values explicitly (for example, **262144**) or by using a prefix multiplier (for example, **256K**).

The minimum value is **64K** and the default value is **128K** for both file attributes.

The syntax of the **fileAttributes** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        [fileAttributes [file-list] initSize=bytes|suffix-multiplier
        extentSize=bytes|suffix-multiplier|
        initSize=bytes|suffix-multiplier|
        extentSize=bytes|suffix-multiplier]
```

freeSpace

The **freeSpace** command enables you to run a free-space evaluation routine on all database files or selected files specified in the optional *[file-list]* argument. Specify the optional **nostatus** argument if you do not want the status message displayed during the free-space evaluation operation.

When you specify the **freeSpace** command, you cannot also specify the **certify**, **compact**, **delete**, **reindex**, or **resetTS** command in the **jdbutilb** command line.

The syntax of the **freeSpace** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        freeSpace [file-list]
        [nostatus]
```

For details, see "[Evaluating Free-Space](#)" and "[Using the Evaluate Free Space Command](#)", elsewhere in this document.

freezeSchemaFiles

The **freezeSchemaFiles** command changes the volatility of the **_userint**, **_userscm**, and **_userxrf** schema files to frozen.

Setting the file volatility to **Frozen** overrides object volatility for objects stored in that file.

Freezing schema files provides performance improvements, by reducing the number of lock and unlock operations and fetched user schema objects. This benefit is a result of ignoring implicit lock requests on those objects stored in the frozen schemas. For details about object volatility, refer to "[Cache Concurrency](#)", in Chapter 6 of the *JADE Developer's Reference*.

Note Attempts to create or maintain any development objects (for example, all schemas, classes, methods, forms, and so on) are disallowed while the schema files are in a frozen state. An attempt to modify any such object raises exception 1106 (*Cannot update a frozen object*).

This also applies to maintenance activities outside the JADE development environment, including **jadloadb** batch JADE Database utility operations.

The syntax of the **freezeSchemaFiles** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        freezeSchemaFiles
```

generateEnvironmentIdentity

The **generateEnvironmentIdentity** command enables you to generate a new environment identity for an environment. The environment identity uniquely identifies an environment and it is shared by all databases in an SDS environment.

The syntax of the **generateEnvironmentIdentity** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         generateEnvironmentIdentity
```

Note When you change the environment identity of the primary, the change is written to the journal and is replayed automatically on SDS secondary databases. It comes into effect when the database is restarted.

generateServerIdentity

The **generateServerIdentity** command enables you to generate a new server identifier for a server.

The server identity uniquely identifies a secondary database in an SDS environment, as its database identity and environment identity must be the same as those of the primary database.

The syntax of the **generateServerIdentity** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         generateServerIdentity
```

generateUUID

The **generateUUID** command enables you to explicitly generate a new database unique identifier (uuid).

The syntax of the **generateUUID** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         generateUUID
```

Tip You should run this command in the batch JADE Database utility whenever you clone a database for a purpose other than to become an SDS secondary database (for example, when you take a copy of a development or production database for use in a test environment).

This is good practice, as it allows SDS to positively identify when a non-SDS clone of a database inadvertently attempts to join a Synchronized Secondary Environment (SDE). Although SDS detects this by other means, it is more obvious when non-SDS clones are assigned unique IDs.

help

The **help** command displays the required commands and arguments, and their values. The syntax of the **help** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         help
```

journalingRates

The **journalingRates** command produces a journaling rates text file report of the transaction journal file or files specified using the **journal** argument, to enable you to plan the capacity for an SDS connection.

The syntax of the **journalingRates** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        journalingRates
        journal=journal-file-number
        [journalDir=journal-directory]
        [lastJournal=journal-file-number]
        [samplingInterval=minutes]
        [nostatus]
```

Specify the **samplingInterval** argument if you want journal rates sampled at an interval other than one minute (that is, journal rates are sampled at one-minute intervals, or 60000 milliseconds, by default).

The following example shows the use of the **journalingRates** command in the **jdbutilb** command line to output the journaling rates of journal **db0000076894.log**.

```
jdbutilb path=s:\testjade\db\logs\current ini=s:\testjade\jade.ini journalingRates
        journal=76894 journalDir=s:\testjade\logs\current samplingInterval=15
```

Using this example, the rates of the following journal file are sampled and output every 15 minutes to the **journalingrates.csv** file in the same directory. (The 15-minute sampling interval is used to calculate the rate values in the **journalingrates.jra** text file.)

```
d:\testjade\db\logs\current\db0000076894.csv
```

For more details, see "Producing Journaling Rates Reports", in Chapter 3.

listBackupinfoFileNames

The **listBackupinfoFileNames** command enables you to display the names of all files (database files, partitions, Unstructured Data Resource (UDR), and single UDR files) in the **backupinfo** file located in the path specified in the **path** argument.

The name list is useful for managing clean-up at the full backup location; for example, detecting when a file has been removed from an environment.

The syntax of the **listBackupinfoFileNames** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        listBackupinfoFileNames
```

listFiles

The **listFiles** command enables you to display the internal file number, file information, name, mapped directory, initial file size, extent size, volatility (for example, **Transparent**, **Frozen**, **Volatile**, or **Unknown** if offline or not yet created), and the JADE version (for example, **NA** if not applicable, **Unknown** if offline or not yet created, or **7.1.04.036**) of all files in the JADE database; for example:

```
#21  a-  _system  x:/jade/system/  16384  65536  Transparent  NA
#44  o-  _rpstrans  <no dir> use c:/jade/system  16384  65536  Unknown
Unknown
```

The syntax of the **listFiles** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         listFiles
```

The column after the file number column contains the files partition status. Partitioned files have a **P** in this column. The two-character column following the file number column contains file status information.

- The values and meaning of the first character are:
 - **a**, indicating that the file status is assigned
 - **d**, indicating that the file status is deleted
 - **n**, indicating that the file status is new (that is, defined but not instantiated)
 - **u**, indicating that the file status is unmapped in an RPS database
 - **o**, indicating that the file status is offline
 - **?**, indicating that the file status is invalid or undefined
- The values and meaning of the second character are:
 - **-** (hyphen), indicating that there is nothing significant to note
 - **r**, for a repeated file name; that is, an entry with this name has already been seen

listPartitions

The **listPartitions** command displays the partition number, file status, name, label, volatility, and mapped directory of all partitions in the file; for example:

```
#1      a      part0000000001      <null>      Frozen      E:\jadexxdev\system
```

The second column containing the file status has the same meaning as the first character of the file status information column in the **listFiles** command.

The syntax of the **listPartitions** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         listPartitions file-name
```

makeBackupinfo

The **makeBackupinfo** command applies the latest partial backup (delta) **backupinfo** file located in the directory specified in the **deltaDir** argument to the last full (base) backup **backupinfo** file located in the directory specified in the **baseDir** argument, creating an output full backup **backupinfo** file in the directory specified in the **outputDir** argument, thus simplifying operational issues restoring the environment from backup. (Every backup creates a **backupinfo** file that contains information specific to the map files and partitions that were in this backup instance.)

File and partition backup information in the delta **backupinfo** file is copied to the output **backupinfo** file. File and partition information that is absent from the delta **backupinfo** file is copied from the base **backupinfo** file to the output **backupinfo** file.

Note It is the responsibility of the backup application to relocate necessary files.

You can use the [listBackupinfoFileNames](#) command to retrieve the names of database files, partitions, Unstructured Data Resource (UDR), and single UDR files from the **backupinfo** file. The name list is useful for managing clean-up at the full backup location; for example, detecting when a file has been removed from an environment.

The syntax of the **makeBackupinfo** command in the **jdbutilb** command line is as follows.

```
jdbutilb=path=database-path
        ini=initialization-file-name
        makeBackupinfo
        baseDir=directory
        deltaDir=directory
        outputDir=directory
```

In the following example, the **baseDir** argument specifies the last full backup, the **deltaDir** argument is the current partial backup, and the **outputDir** argument is where the new **backupinfo** file is to be placed.

```
jdbutilb ini=e:\jadetest\jade.ini makebackupinfo
        baseDir="e:\jadetest\backup\offline full"
        deltaDir="e:\jadetest\backup\offline partial"
        outputDir="e:\jadetest\backup\offline work"
```

mapFile

The **mapFile** command enables you to change the status of unmapped database files in an RPS database configured with the **Mapped Extent** storage mode to **Assigned**.

The syntax of the **mapFile** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        mapFile [file-list]
```

The following example shows the use of the **mapFile** command.

```
jdbutilb.exe path=s:\jade\system ini=s:\jade\backups\jade.ini mapFile _rootdef
        testschema locktest
```

For details about unmapping assigned database files, see the [unmapFile](#) command, later in this chapter.

markOffline

The **markOffline** command changes the state of specified database files to offline; for example, you could use this to mark the development-only **_sysdev**, **_jadeapp**, and **_jadedef** system files as offline in a non-development database. (For details about deployment and development JADE system files, see "[System Map Files](#)", in Chapter 3 of the *JADE Development Environment User's Guide*.)

The syntax of the **markOffline** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        markOffline [file-list]
```

The following example shows the **markOffline** command in the **jdbutilb** command line.

```
jdbutilb path=c:\jade\system ini=c:\jade\jade.ini markOffline _sysdev _jadeapp _
        jadedef
```

markOnline

The **markOnline** command changes the state of specified database files to online.

The syntax of the **markOnline** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        markOnline [file-list]
```

The following example shows the **markOnline** command in the **jdbutilb** command line.

```
jdbutilb path=c:\jade\system ini=c:\jade\jade.ini markOnline _sysdev _jadeapp _
jadedef
```

productionMode

The **productionMode** command enables you to specify that you want production mode set or unset for your JADE database.

Specify **true** if you want the production mode flag set on your JADE database or **false** to unset production mode. (You can use the **showInfo** command to display the current mode of your database files.)

The syntax of the **productionMode** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        productionMode=true|false
```

For more details, see "[Using the Production Mode Command](#)", earlier in this chapter.

recover

The **recover** command enables you to perform a roll-forward recovery, specifying termination conditions, of a database restored from backup to its final production location. Use this command to initiate the recovery of an online backup that has been restored separately; for example, direct from tape.

Roll-forward recovery reapplies transactions in the journals; that is, all database activity since a full backup was taken.

The syntax of the **recover** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        recover
        [recoverTo=date time|lastJournal=journal-file-number
        |lastSerialNumber=audit-record-number]
        [clearSDSRole]
        [nostatus]
```

The following example shows the use of the **recover** command in the **jdbutilb** command line to recover all committed transactions up to 8:41am on 14 July 2011.

```
jdbutilb path=d:\jade\newdb ini=d:\jade\jade.ini recover recoverTo=14\07\2011
08:41:09
```

By default, the recovery command applies all committed transactions contained in journals up to and including the latest current transaction journal, unless you specify the optional **recoverTo** argument described earlier in this chapter.

For more details, see "[Recovery Concepts and Strategies](#)" and "[Using the Initiate Recovery Command](#)", elsewhere in this document.

reindex

The **reindex** command enables you to reindex database files to repair damaged file indexes or to perform free-space garbage collection serially. You can perform this operation on all of your database files or only the files specified in the optional *[file-list]* argument, based on the object sequence. Specify the optional **nostatus** argument if you do not want the status message displayed during the reindex operation.

When you specify the **reindex** command, you cannot also specify the **certify**, **compact**, **delete**, **freeSpace**, or **resetTS** command in the **jdbutilb** command line.

The syntax of the **reindex** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        reindex
        [file-list]
        [nostatus]
```

For more details, see "[Reindexing Database Files](#)" and "[Using the Reindex Files Command](#)", elsewhere in this document.

resetTS

The **resetTS** command enables you to reset the timestamps in all of your database files or only the file or files specified in the optional *[file-list]* argument. You cannot reset timestamps if database recovery is required.

Caution Use this command with care, and only when it is necessary to override timestamps when files have been legitimately replaced.

When you specify the **resetTS** command, you cannot also specify the **certify**, **compact**, **delete**, **freeSpace**, or **reindex** command in the **jdbutilb** command line.

The syntax of the **resetTS** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        resetTS [file-list]
```

For more details, see "[Resetting Timestamps](#)" and "[Using the Reset Timestamps Command](#)", elsewhere in this document.

restore

The **restore** command initiates an offline restore of all database files from the backup directory specified by using the **backupDir** argument and performs a roll-forward recovery unless recovery is disabled by the optional **noRecovery** argument.

Information or errors from the restore operation are recorded in the **recovery.log** file located in the directory specified in the **ActivityLogDirectory** parameter in the **[PersistentDb]** section of the JADE initialization file.

Caution If errors are encountered during the restore and recover process, you will not be able to use the database.

The syntax of the **restore** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        restore
        backupDir=backup-directory
        [recoverTo=date time|lastJournal=journal-file-number
          |lastSerialNumber=audit-record-number]
        [noRecovery]
        [clearFilePaths]
        [clearSDSRole]
        [excludeFrozenFiles]
        [excludeFrozenPartitions]
        [verify]
        [nostatus]
```

The following example shows the use of the **restore** command in the **jdbutilb** command line to restore all files from the last backup made to the **c:\jade\backup03092001** directory and recover committed transactions up to 8:41am on 14 July 2011.

```
jdbutilb path=d:\jade\newdb ini=d:\jade\jade.ini restore
backupDir=e:\jade\backup03092001
recoverTo=14\07\2011 08:41:09 verify nostatus
```

In this example, the database control file (**_control.dat**) and all database files that do not have specified explicit directories are restored to the **d:\jade\newdb** directory, regardless of the default database directory from which the files were backed up. Any files that have explicit directories in the database control file are restored to those locations.

Note You must specify in the **path** argument a directory that is valid on the server.

When recovery of all committed transactions up to a specified date and time are specified, any transactions that completed after that date and time are not reapplied. For more details, see "[Restoring and Recovering the Database](#)" and "[Using the Restore Database Command](#)", elsewhere in this document.

restoreFile

The **restoreFile** command initiates an offline restore of all database files specified in the *file-list* argument from the backup directory specified by using the **backupDir** argument to the database location (which must be valid on the server) specified in the **path** argument.

The **restoreFile** command enables you to restore groups of database files concurrently by executing **jdbutilb** processes to exploit hardware parallelism, if available. Information or errors from the file restore operation are recorded in the **recovery.log** file located in the directory specified in the **ActivityLogDirectory** parameter in the **[PersistentDb]** section of the JADE initialization file.

The syntax of the **restoreFile** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        restoreFile
        backupDir=backup-directory
        file-list
        [clearFilePaths]
        [excludeFrozenFiles]
        [excludeFrozenPartitions]
```

```
[verify]
[nostatus]
```

When you restore a file:

- The first database file in the restored file list must be the control file (that is, **_control.dat**).
After you have restored the control file, you can initiate concurrent file restore operations by creating multiple **jdbutilb** processes.
- Each file in a list is space-separated.
- Specify only the file prefix; that is, do not include the **.dat** file suffix.
- Simple file masks are not supported; for example, a file list of **_user*** is invalid.

When restoring the database control file (**_control.dat**), the **restoreinfo** file in the database directory is created. If you do not want frozen (read-only) files or partitions carried over from the **backupinfo** file to the **restoreinfo** file, specify the optional **excludeFrozenFiles** or **excludeFrozenPartitions** argument, respectively. This will cause the excluded files and partitions to be marked offline, when the restored database is initialized.

Caution Owing to increased disk contention and disk head movement, concurrent restore operations run slower if the restore is sent to a single disk drive.

If errors are encountered during the file restore operation, you will not be able to use the database. Do not use the database until all files are restored.

The following example shows the use of the **restoreFile** command in the **jdbutilb** command line to restore the **userdata1.dat** and **userdata2.dat** files from the last backup made to the **e:\jade\backup3103** directory.

```
jdbutilb path=d:\jade\system ini=d:\jade\jade.ini restore
backupDir=e:\jade\backup3103
_control userdata1 userdata2 verify nostatus
```

restoreJournal

The **restoreJournal** command initiates a restore of the transaction journal file specified in the **journal** argument.

Information or errors from the backup operation are recorded in the **recovery.log** file located in the default database directory (specified by using the **path** argument).

The syntax of the **restoreJournal** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         restoreJournal
         backupDir=backup-directory
         journal=log-file-number
         [lastJournal=journal-file-number]
         [nostatus]
```

The following example shows the use of the **restoreJournal** command in the **jdbutilb** command line to restore transaction journal number **45** from the **e:\jade\backup230304** backup directory.

```
jdbutilb path=d:\testjade\db ini=d:\testjade\jade.ini restoreJournal
backupDir=e:\jade\backup230304 journal=45 nostatus
```

In this example, the restore operation expects to find the database control file in the **d:\testjade\db** directory and the transaction journal will be restored to the **current** log directory for the database.

For more details, see "[Using the Restore Journals Command](#)", earlier in this chapter.

restorePartitionFile

The **restorePartitionFile** command initiates an offline restore of the partitioned database file identified by the **part** argument in the file identified by the **file-name** argument from the backup directory specified by the **backupDir** argument to the database location (which must be valid on the server) specified in the **path** argument.

Information or errors from the partitioned file restore operation are recorded in the **recovery.log** file located in the directory specified in the **ActivityLogDirectory** parameter in the **[PersistentDb]** section of the JADE initialization file.

The syntax of the **restorePartitionFile** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         restorePartitionFile
         backupDir=backup-directory
         file-name
         part=partition-identifier
         [clearFilePaths]
         [verify]
         [nostatus]
```

When you restore a partitioned file, the specified value of the **file-name** argument must be the partition control file (that is, **file-name.dat**).

To restore only the partition control file and index before you restore individual partitions, specify:

```
restorePartitionFile file-name part=0
```

Caution If errors are encountered during the partitioned file restore operation, you will not be able to use the database. Do not use the database until the file partition is restored.

setFilePath

The **setFilePath** command enables you to set the path of all database files or selected files specified in the optional **[file-list]** argument so that you can relocate files.

When setting file paths, existing ***.dat** files and dependent files are moved to the specified path.

Note As the **setFilePath** action is not audited, it is not replayable.

The syntax of the **setFilePath** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         setFilePath
         fileDir=directory
         [file-list]
```

The following example shows the use of the **setFilePath** command in the **jdbutilb** command line when a secondary system has had more disk added so some files of employee data are moved to a new drive.

```
jdbutilb path=e:\ProdB\system ini=e:\ProdB\jade.ini setFilePath
fileDir=f:\ProdB\system emp*
```

In the following example, a production system file is moved to a new drive, using the `unaudited` command, so that it is not redone in roll-forward recovery.

```
jdbutilb path=e:\ProdA\system ini=e:\ProdA\jade.ini setFilePath  
fileDir=f:\ProdA\system AcctHist
```

setFilePathAudited

The **setFilePathAudited** command enables you to set the path of all database files or selected files specified in the optional *[file-list]* argument so that you can relocate files.

When setting file paths, existing *.dat files and dependent files are moved to the specified path.

Note As the **setFilePathAudited** action is audited, it can be replayed for archival recovery purposes (if you want the operation to be redone in roll-forward recovery). SDS Secondary replay does not replay these audit records.

The syntax of the **setFilePathAudited** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path  
ini=initialization-file-name  
setFilePathAudited  
fileDir=directory  
[file-list]
```

setUserFileVersions

The **setUserFileVersions** command sets the version number to the 32-bit unsigned integer value specified in the **version** argument of all user database map files or selected files specified in the optional *[file-list]* argument. This can be applied only to *user* data map files; that is, user schema files (for example, `_userdev.dat`, `_userscm.dat`, and so on) represented by the **DbFile** class **Kind_User_Schema** constant and user data files (for example, `_rootdef.dat`, `locktest.dat`, and `_indexdefs.dat`) represented by the **DbFile** class **Kind_User_Data** constant.

This enables system administrators to optionally assign a version number to user data map files. The JADE database preserves these values (for example, for reorganization, database compaction, application deployment, and so on) but ignores them.

Note As the **setUserFileVersions** action is not audited, the version number is not propagated to SDS secondaries.

The current value of the version number is available programmatically by calling the **DbFile** class **getUserPatchVersion** method or from the **jverinfo** JADE Version Information utility. (For details, see [Volume 1](#) of the *JADE Encyclopaedia of Classes* or "[Obtaining the Version Number of User Database Files](#)" under "[Using the JADE Version Information Utility](#)", in Chapter 2 of the *JADE Installation and Configuration Guide*.)

The syntax of the **setUserFileVersions** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path  
ini=initialization-file-name  
setUserFileVersions  
version=version-number  
[file-list]
```

showInfo

The **showInfo** command enables you to display the following information about the current database files.

- Database ID.
- Database Role, which displays **Primary** or **Secondary**.
- Last Database Mode, which displays **Archive**, **Default**, or **Shared**.
- Last Database Usage, which displays **NoAudit**, **ReadOnly**, or **Update**.
- Production Mode, which displays whether production mode is set to **Disabled** or **Enabled**. (For details about production mode, see "[productionMode](#)", earlier in this chapter.)
- Character Set, which can be ANSI or Unicode, depending on the type of JADE files that you have installed.
- Db Code Version, which displays the version of the database (for example, **7.1.03.005**).
- Highest transaction ID, which displays the latest identifier from the most recent transaction (for example, **14203**).
- Current journal (for example, **142**).
- Start recovery journal (for example, **142**).
- Start backup journal (for example, **0**).

The syntax of the **showInfo** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         showInfo
```

thawSchemaFiles

The **thawSchemaFiles** command changes the volatility of the **_userint**, **_userscm**, and **_userxrf** schema files to transparent. This removes the volatility override put in place by the **freezeSchemaFiles** command.

For details about object volatility, refer to "[Cache Concurrency](#)", in Chapter 6 of the *JADE Developer's Reference*.

The syntax of the **thawSchemaFiles** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         thawSchemaFiles
```

touchDB

The **touchDB** command enables you to open and close the database; for example, to ensure that the database is not in recovery state.

The syntax of the **touchDB** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         touchDB
```


unmapFile

The **unmapFile** command enables you to unmap assigned database files in an RPS database configured with the **Mapped Extent** storage.

The syntax of the **unmapFile** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        unmapFile [file-list]
```

The following example shows the use of the **unmapFile** command.

```
jdbutilb.exe path=s:\jade\system ini=s:\jade\backups\jade.ini unmapFile _rootdef
testschema locktest
```

For details about assigning unmapped database files, see the **mapFile** command, earlier in this chapter.

verifyChecksums

The **verifyChecksums** command enables you to perform a checksum analysis of your backed up database files or your restored system to verify that the files have not been corrupted by a hardware or environmental problem during the backup or restore process.

Information or errors from the checksum verification operation are recorded in the **checksum.log** file located in the directory specified by using the **path** argument if the **ActivityLogDirectory** parameter is not specified in the **[PersistentDb]** section of the JADE initialization file.

The checksums verification operation is driven from the **backupinfo** file in the specified backup directory or the **restoreinfo** file in the database system directory.

Note Perform a checksum analysis of any backup that has been moved across media, especially if transferred across a network.

Specify the optional **workers** argument if you want to use a single worker thread (1) or more than the default two (2) multiple concurrent worker threads. The **verifyChecksums** operation can use a maximum of 16 multiple concurrent worker threads.

The syntax of the **verifyChecksums** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
        ini=initialization-file-name
        verifyChecksums [workers=number-of-worker-threads]
```

This command performs a checksum analysis of files in the directory specified by the **path** argument or the **ActivityLogDirectory** parameter in the **[PersistentDb]** section of the JADE initialization file and returns a result for batch file **ERRORLEVEL** checking.

The following are examples of the checksum verification commands for backed up and restored files, respectively.

```
jdbutilb path=d:\jade\backup\offline\compressed ini\jade\jade.ini verifyChecksums
workers=3
```

```
jdbutilb path=d:\jade\system ini=d:\jade\jade.ini verifyChecksums
```

verifyJournal

The **verifyJournal** command initiates a verification of the transaction journal file specified by using the **journal** argument.

Details of the verify operation are recorded in the **journal-file-name.scan** file, located in the **current** logs directory in your default database directory (specified by using the **path** argument).

The syntax of the **verifyJournal** command in the **jdbutilb** command line is as follows.

```
jdbutilb path=database-path
         ini=initialization-file-name
         verifyJournal
         journal=journal-file-number
         [journalDir=journal-directory]
         [lastJournal=journal-file-number]
         [nostatus]
```

The following verification example shows the use of the **verifyJournal** command in the **jdbutilb** command line to verify the transaction journal number **0000065489**.

```
jdbutilb path=d:\testjade\db ini=d:\testjade\jade.ini verifyJournal journal=65489
```

For more details, see "[Using the Verify Journals Command](#)", earlier in this chapter.

This chapter covers the following topics.

- [Overview](#)
- [Running the JADE Database Administration Utility](#)
 - [Command Arguments](#)
 - [Database Actions](#)
 - [File or Partition Actions](#)
 - [File Actions](#)
 - [Partition Actions](#)

Overview

The JADE Database Administration utility (**jdbadmin**) is a command line interface similar to the single user offline batch database utility (**jdbutilb**), but is one that can execute database administrative operations, including file partition-related operations, online in multiuser mode or offline in single user mode.

The JADE Database Administration utility assumes the credentials of the user account executing the operating system process. This program must be executed on the same machine as the database server node.

The user account executing the **jdbadmin** process must be one of the following.

- Member of the local or domain administrators group
- The same account as the database server under which it is running

When running in single user mode, the account executing the **jdbadmin** process must have write access to files in the target database directory.

When you run **jdbadmin** and connect to a database server, there is an authentication process in which the server creates a key file that the client has to open. The **jdbadmin.template** client template file is created with special Access Control Lists (ACLs) and is used to create the JADE Database Administration utility per-session key files in the database directory. By default, template files are created in the JADE HOME directory. You can use the [ACLTemplateDirectory](#) parameter in the [\[JadeEnvironment\]](#) section of the JADE initialization file to control the location of the template files.

For details about the parameters that provide advanced diagnostic capabilities when performing certify and freespace evaluation operations, see "[Database Diagnostics Section \[DBUtil\]](#)", in the *JADE Initialization File Reference*.

Running the JADE Database Administration Utility

The **jdbadmin** program, installed with your JADE software, enables you to automate the JADE Database Administration utility by running it in batch mode. You can use this utility, for example, to enable the partitioning of a database file or to perform an online backup of selected files in multiuser mode. Run the JADE Database Administration utility (for example, from a command script), specifying the following.

```
jdbadmin action=Backup|ListDbFiles|FreezeSchemaFiles|ThawSchemaFiles|
Compact|ListParts|MakePartitioned|SetFilePath|
ActivateDeltaDb|DeactivateDeltaDb|FileOrPart-Command|
Partition-Command|StartSnapshot|EndSnapshot|Help
[path=database-path]
[ini=initialization-file-name]
[server=multiUser|singleUser]
```

Commands have the following syntax.

Backup	backupDir=file-directory [backup-arguments]
ListDbFiles	ListDbFiles
FreezeSchemaFiles	FreezeSchemaFiles
ThawSchemaFiles	ThawSchemaFiles
Compact	Compact [workers=number-of-workers [updatesAllowed=true false]] [files=file-name1,file-name2,... file=file-name]
ListParts	ListParts file=file-name
MakePartitioned	MakePartitioned file=file-name [method=method-name] [modulus=integer-value] [updatesAllowed=true false]
SetFilePath	SetFilePath file=file-name filePath=file-directory [audited=true false]
ActivateDeltaDb	ActivateDeltaDb [server=multiUser]
DeactivateDeltaDb	DeactivateDeltaDb [timeout=maximum-number-of-seconds] [server=multiUser]
FileOrPart-Command	Certify FreeSpace [files=file-name1,file-name2,... file=file-name [part=partition-id]] Freeze Thaw file=file-name [part=partition-id]
Partition-Command	MarkOffline MarkOnline PurgePartition RemovePartition file=file-name part=partition-id MovePartition SetPartitionLocation file=file-name part=partition-id location=file-directory SetPartitionLabel file=file-name part=partition-id Label=label-name
StartSnapshot	[server=multiUser]
EndSnapshot	[server=multiUser]

Backup command arguments have the following syntax.

```
[compress=true|false]
[excludeFrozenFiles=true|false]
[excludeFrozenPartitions=true|false]
[overwrite=true|false]
[quiesced=true|false]
[showProgress=true|false]
[verify=true|false]
```

Standard database operation information is output to **stdout** and error information is output to **stderr**. For details about displaying and redirecting the output from JADE batch utilities, see the [DisplayApplicationMessages](#), [LogServer](#), and [UseLogServer](#) parameters under "JADE Log Section [[JadeLog](#)]", in the *JADE Initialization File Reference*.

Database administration actions are logged in the activity log.

If the **jdbadmin** executable program fails, a non-zero exit code is returned and an error message is displayed; for example, if the database directory was invalid.

Command Arguments

The JADE Database Administration utility command line argument values that apply to one or more batch database administration utility actions or commands are described in the following subsections.

action

The **action** argument specifies the database administration function to initiate; for example:

```
jdbadmin action=Backup backupDir=c:\jade\temp path=c:\jade\system  
ini=c:\jade\system\jade.ini server=multiUser compress=true  
showProgress=true verify=
```

An action of **help** or no action outputs help information.

The commands that you can action are grouped by:

- Database actions
 - [ActivateDeltaDb](#)
 - [Backup](#)
 - [DeactivateDeltaDb](#)
 - [EndSnapshot](#)
 - [FreezeSchemaFiles](#)
 - [ListDbFiles](#)
 - [StartSnapshot](#)
 - [ThawSchemaFiles](#)
- File or partition actions
 - [Certify](#)
 - [FreeSpace](#)
 - [Freeze](#)
 - [Thaw](#)
- File actions
 - [Compact](#)
 - [ListParts](#)
 - [MakePartitioned](#)
 - [SetFilePath](#)
- Partition actions
 - [MarkOffline](#)
 - [MarkOnline](#)
 - [MovePartition](#)

- [PurgePartition](#)
- [RemovePartition](#)
- [SetPartitionLabel](#)
- [SetPartitionLocation](#)

path

The **path** argument specifies the database directory in which the control file (**_control.dat**) is located. This argument is required for single user execution and it is optional for multiuser execution.

ini

The **ini** argument specifies the fully qualified name of the JADE initialization file. The default value is **jade.ini**.

Tip Always specify this argument, to ensure that the correct JADE initialization file is used.

server

The optional **server** argument specifies the operational mode. When this argument is not specified, normal default evaluation of the mode occurs; that is, the value of the [Server](#) parameter in the [\[Jade\]](#) section in the JADE initialization file is evaluated and in its absence, the operational mode defaults to multiuser.

Database Actions

The database actions that you can perform are described in the following subsections.

ActivateDeltaDb

The **ActivateDeltaDb** command creates a new delta database in the **deltadb** subdirectory of the root database and then converts the root database to delta mode. A database in delta mode allows non-permanent (or tentative) changes to be made to a database in read-only mode.

Note The [DeltaDatabaseCapable](#) parameter in the [\[JadeServer\]](#) section of the JADE initialization file on the database server node must be set to **true** for an activation request to succeed.

Activating a delta database creates a new (empty) delta database and converts the current (root database) to delta mode. When the root database is in delta mode, it is in a read-only state and all object create, update, and delete operations on non-environmental persistent objects are redirected to the delta database. When an SDS native or RPS secondary database is in delta mode, database tracking is halted.

Current applications do not need to be idle when the delta database is being activated. However, allowing database transactions that started before delta mode was activated to commit after activation is not permitted. To prevent in-progress transactions from spanning an activation boundary, activating delta mode acquires a database quietpoint, which is acquired by blocking the start of new transactions in the database engine and waiting for existing transactions to complete.

If a database quietpoint cannot be achieved within the number of seconds specified by the [MaxWaitForQuietPoint](#) parameter in the [\[PersistentDb\]](#) section of the JADE initialization file, activation fails and error 3077 is returned.

The syntax of the **ActivateDeltaDb** action is as follows.

```
jdbadmin action=ActivateDeltaDb
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser]
```

The following is an example of the **ActivateDeltaDb** action.

```
jdbadmin path=d:\salesdb ini=d:\salesdb\jade.ini action=ActivateDeltaDb
```

Exception 1162 (*The system is not Delta Database capable*) is raised if you attempt to activate a delta database in single user mode.

Note You can also activate the delta database at run time, by calling the **System** class **activateDeltaDatabase** method with the **activate** parameter set to **true**.

Backup

The **Backup** action initiates a full backup that includes all online database files or a partial backup that excludes frozen files or frozen partitions, or both frozen files and partitions. Offline files and partitions are always excluded.

You can execute this command in multiuser mode to perform an online backup, with concurrent updating transactions or in quiesced (read-only) mode. In single user mode, the **path** argument is required.

The syntax of the **Backup** action is as follows.

```
jdbadmin action=Backup
        backupDir=backup-directory
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        [compress=true|false]
        [excludeFrozenFiles=true|false]
        [excludeFrozenPartitions=true|false]
        [overwrite=true|false]
        [quiesced=true|false]
        [showProgress=true|false]
        [verify=true|false]
```

In addition to the command line argument values that apply to one or more batch database administration utility actions or commands, documented under "**Command Arguments**", the **Backup** action arguments are listed in the following table.

Argument	Description
backupDir	Specifies the destination directory. The directory is created if it is not present.
path	Optional, in multiuser mode.
compress	Specifies that data compression is invoked to compress data as it is copied to the backup. The default value is false .
excludeFrozenFiles	Specifies that all frozen files are to be excluded from the backup. The default value is false .
excludeFrozenPartitions	Specifies that all frozen partitions are to be excluded from the backup. The default value is false .

Argument	Description
overwrite	Specifies that existing files in the backup location are to be overwritten. The default value is false .
quiesced	Specifies that the backup is to be a quiesced (read-only) backup. The default value is false .
showProgress	Specifies that progress information is to be sent to stdout . The default value is false .
verify	Specifies that file-level MD5 checksum verification is to be performed on each file after being copied. The default value is false .

DeactivateDeltaDb

The **DeativateDeltaDb** action deactivates delta mode and removes the associated delta database.

The syntax of the **DeactivateDeltaDb** action is as follows.

```
jdbadmin action=DeactivateDeltaDb
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser]
        [timeout=maximum-number-of-seconds]
```

The optional **timeout** argument is used to specify a maximum number of seconds to allow for deactivation to take place and is ignored when the **ActivateDeltaDb** action is set to **true**.

The delta database cannot be deactivated until all current processes are idle apart from JADE tools (for example, the JADE Monitor and the SDS Administration application) and the process making the request is idle with no persistent locks and not in transaction state.

The deactivation attempt is abandoned if this does not happen within the specified time, and the delta database remains activated. If a timeout or a value of zero (**0**) is not specified, a default timeout of 60 seconds is used.

When an SDS secondary database is taken out of delta mode, database tracking is resumed if it is not disabled. On an RPS secondary, the **Datapump** application is also restarted if it is not running and auto restart data pump is enabled.

If any of the following conditions are not satisfied, deactivation is abandoned and system error 1163 is returned.

- The request cannot be completed because not all applications have become idle within the specified timeout
- The process that requests the deactivation must not have any persistent objects locked
- The process that requests the deactivation must not be in transaction state
- The process that requests the deactivation must not be executing any methods that have a persistent user object as the receiver.

Before the delta database can be deactivated, all current applications (apart from the application making the request) must be idle, not in transaction state, and have no persistent locks held. To be idle, an application must not be currently executing any methods invoked via the JADE object manager; that is, its object manager call stack should be empty. The deactivation attempt is immediately abandoned if there are any idle processes that have persistent objects locked or that are in transaction state.

When a deactivation request is received, each application is allowed to proceed normally until its JADE call stack is empty. When a process is idle, any further attempt to invoke a method is held up until deactivation is completed or abandoned. (The process making the request and JADE tools such as the JADE Monitor and the SDS Administration application are exempt from this.)

Any attempts to start up new applications are also stalled until deactivation is completed or abandoned.

The following is an example of the **DeactivateDeltaDb** command.

```
jdbadmin path=d:\salesdb ini=d:\salesdb\jade.ini action=DeactivateDeltaDb
timeout=300
```

Exception 1162 (*The system is not Delta Database capable*) is raised if you attempt to deactivate a delta database in single user mode.

Note You can also deactivate the delta database at run time, by calling the **System** class **activateDeltaDatabase** method with the **activate** parameter set to **false**.

EndSnapshot

The **EndSnapshot** action takes the database out of snapshot mode by performing a **changeDbAccessMode (JadeDatabaseAdmin.Mode_Default, JadeDatabaseAdmin.Usage_Update)** method call. (For details about conditioning the database for recovery from a non-JADE backup, see "[StartSnapshot](#)", elsewhere in this document.)

The syntax of the **EndSnapshot** action is as follows.

```
jdbadmin action=EndSnapshot
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser]
```

The following is an example of the **EndSnapshot** command.

```
jdbadmin path=d:\salesdb ini=d:\salesdb\jade.ini action=EndSnapshot
```

Note For details about external third-party snapshot backups, see "[Non-JADE Backups](#)" in the *Developing a Backup Strategy White Paper*.

FreezeSchemaFiles

The **FreezeSchemaFiles** action changes the volatility of the **_userint**, **_userscm**, and **_userxrf** schema files to frozen.

Setting the file volatility to frozen overrides any specified object volatility for objects stored in that file. Freezing schema files provides performance improvements, by reducing the number of locks and unlocks and fetched user schema objects. This benefit is a result of ignoring implicit lock requests on those objects stored in the frozen schemas.

For details about object volatility, refer to "[Cache Concurrency](#)", in Chapter 6 of the *JADE Developer's Reference*.

Note Attempts to create or maintain any development objects (for example, schemas, classes, methods, forms, and so on) are disallowed while the schema files are in a frozen state. An attempt to modify any such object raises exception 1106 (Cannot update a frozen object). This also applies to maintenance activities outside the JADE development environment, including **jdbadmin** batch JADE Database Administration utility actions.

The syntax of the **freezeSchemaFiles** action is as follows.

```
jdbadmin action=FreezeSchemaFiles
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
```

The **path** arguments is optional in multiuser mode.

ListDbFiles

The **ListDbFiles** action displays all schema-defined **DbFile** entities and their attributes.

The syntax of the **ListDbFiles** action is as follows.

```
jdbadmin action=ListDbFiles
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
```

The **path** argument is optional in multiuser mode.

StartSnapshot

The **StartSnapshot** action conditions the database for recovery from a non-JADE backup, by performing a **changeDbAccessMode(JadeDatabaseAdmin.Mode_Snapshot, JadeDatabaseAdmin.Usage_Update);** method call. (For details about taking the database out of snapshot mode, see "[EndSnapshot](#)", elsewhere in this document.)

The syntax of the **StartSnapshot** action is as follows.

```
jdbadmin action=StartSnapshot
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser]
```

The following is an example of the **StartSnapshot** command.

```
jdbadmin path=d:\salesdb ini=d:\salesdb\jade.ini action=StartSnapshot
```

Note For details about external third-party snapshot backups, see "[Non-JADE Backups](#)" in the *Developing a Backup Strategy White Paper*.

ThawSchemaFiles

The **ThawSchemaFiles** action changes the volatility of frozen **_userint**, **_userscm**, and **_userxrf** schema files to transparent. This removes the volatility override put in place by the **FreezeSchemaFiles** action.

The syntax of the **ThawSchemaFiles** action is as follows.

```
jdbadmin action=ThawSchemaFiles
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
```

The **path** argument is optional in multiuser mode.

File or Partition Actions

The file or partition actions that you can perform are described in the following subsections.

Certify

The **Certify** action performs a check on the integrity of all database files, a specific file, a list of files, or a selected partition of a file. As each file is certified, its access mode is changed to read-only and then back to update mode after the certify action is complete.

The syntax of the **Certify** action is as follows.

```
jdbadmin action=Certify
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        [files=file-name1, file-name2,...|file=file-name [part=partition-ID]]
```

The **path** argument is optional in multiuser mode.

The **files** parameter can be used to specify a file list or the **file** argument can be used to identify a specific file. The **part** argument, when specified, limits the certification to the specified partition of the file specified in the **file** argument.

If you do not specify the **file** or **files** argument, a list of files that will not be processed for non-error reasons is output to the console (that is, to **stdout**).

For details, see "[Database Certification](#)", in Chapter 3.

FreeSpace

The **FreeSpace** action performs a free-space evaluation routine on all database files, a list of files, or a specific partition of a file.

The syntax of the **FreeSpace** action is as follows.

```
jdbadmin action=FreeSpace
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        [files=file-name1, file-name2,...|file=file-name [part=partition-ID]]
```

The **path** argument is optional in multiuser mode.

The **files** argument can be used to specify a file list or the **file** argument can be used to identify a specific file. The **part** argument, when specified, limits the certification to the specified partition of the file specified in the **file** argument.

If you do not specify the **file** or **files** argument, a list of files that will not be processed for non-error reasons is output to the log file.

For details, see "[Evaluating Free Space](#)", in Chapter 3.

Freeze

The **Freeze** action converts a specific file or partition to read-only mode, after which object update, delete, and create operations are not permitted.

The syntax of the **Freeze** action is as follows.

```
jdbadmin action=Freeze
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name [part=partition-ID]
```

The **path** argument is optional in multiuser mode.

The **file** argument specifies the target file. The **part** argument, when specified, targets the specified partition in the file specified in the **file** argument.

Thaw

The **Thaw** action changes the volatility of the specified file or partition to transparent. This removes the volatility override put in place by the **Freeze** action.

The syntax of the **Thaw** action is as follows.

```
jdbadmin action=Thaw
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name [part=partition-ID]
```

The **path** argument is optional in multiuser mode.

The **file** argument specifies the target file. The **part** argument, when specified, targets the specified partition in the file specified in the **file** argument.

File Actions

The file actions that you can perform are described in the following subsections.

Compact

The **Compact** action executes a compaction routine on all or selected database files.

When a partitioned file is compacted, all file partitions are compacted. You cannot execute this operation for a specific partition.

The syntax of the **Compact** action is as follows.

```
jdbadmin action=Compact
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        [workers=number-of-workers [updatesAllowed=true|false]]
        [files=file-name1, file-name2, ...|file=file-name]
```

The **path** argument is optional in multiuser mode.

The optional **workers** argument enables you to specify the number of multiple concurrent worker threads (greater than 1) that are used in parallel when using the **compact** action. If you do not specify the number of workers or you specify a number less than two workers, serial compact actions are performed. The default value of 2 workers is used if you do not specify the **workers** argument. If you specify a value greater than 16, the maximum number of 16 worker threads is used.

When you specify that multiple worker threads can compact the database in parallel, specify the optional **updatesAllowed** argument with a value of **true** if you want to allow updates when multiple workers are in action during a database file compaction. If the **updatesAllowed** argument is not specified, the value defaults to **false**.

The **files** argument can be used to specify a file list or the **file** argument can be used to identify a specific file. If a file selection is not specified, all database files are compacted. If you do not specify the **file** or **files** argument, a list of files that will not be processed for non-error reasons is output to the console (that is, to **stdout**).

For details, see "Compacting Files", in Chapter 3.

ListParts

The **ListParts** action displays the database file partitions, with attributes, of the specified file.

The syntax of the **ListParts** action is as follows.

```
jdbadmin action=ListParts
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name
```

The **path** argument is optional in multiuser mode.

The **file** argument identifies the target file.

MakePartitioned

The **MakePartitioned** action converts the specified non-partitioned database file into a partitioned format.

The syntax of the **MakePartitioned** action is as follows.

```
jdbadmin action=MakePartitioned
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name
        [method=method-name]
        [modulus=integer-value]
        [updatesAllowed=true|false]
```

In addition to the command line argument values that apply to one or more batch database administration utility actions or commands, documented under "Command Arguments", the **MakePartitioned** action arguments are listed in the following table.

Argument	Description
path	Optional, in multiuser mode.
file	Specifies the target file.
method	Identifies the partition method that is to be called as each object is processed. A partition method returns a partition index value that determines the partition in which the receiver is located. Subobjects (for example, collections, blob or slob properties, and JADE bytes) are automatically located in the same partition as the parent object. If the method argument is not specified, the autoPartitionIndex method is called. The default Object -level implementation of this method returns the value zero (0), which causes objects to be located in the latest partition.

Argument	Description
modulus	<p>Specifies the creation window size for the operation. This also establishes the number of partitions that will be created. If specified, the modulus value must be an integer in the range 1 through 1024. The default value of 1 results in a single partition getting instantiated.</p> <p>The effects of a successful MakePartitioned operation are redone by roll-forward recovery and replayed on an SDS secondary database. To accomplish that, a copy of the partition index is inserted in the database journal when the converted file is instantiated.</p>
updatesAllowed	<p>If you want to partition a file while allowing applications to continue updating the file being partitioned, set the value of the optional updatesAllowed argument to true. If the argument is not specified, the value defaults to false; that is, updates of a partitioned file are not permitted.</p> <p>If you execute the MakePartitioned command with updatesAllowed=true, when the file conversion process has completed, updates audited in transaction journals are applied to the output file. This recovery phase is referred to as <i>file synchronization</i>.</p> <p>During the initial file recovery or synchronization phase, further updates to the file are permitted. However, when the file is ultimately instantiated, a database quietpoint is acquired. This quietpoint will momentarily block active transactions from committing while remaining updates are applied and the partitioned file is instantiated.</p>

The following is an example of the **MakePartitioned** command.

```
jdbadmin path=c:\JADE_JRF301 ini=c:\JADE_JRF301\jade.ini action=MakePartitioned
file=sales modulus=36 updatesAllowed=true
```

SetFilePath

The **SetFilePath** action changes the control file path attribute of the specified database file and moves the database file and component files to the specified path.

The syntax of the **SetFilePath** action is as follows.

```
jdbadmin action=SetFilePath
[path=database-path]
[ini=initialization-file-name]
[server=multiUser|singleUser]
file=file-name
filePath=file-directory
[audited=true|false]
```

In addition to the command line argument values that apply to one or more batch database administration utility actions or commands, documented under "[Command Arguments](#)", the **SetFilePath** action parameters are listed in the following table.

Argument	Description
path	Optional, in multiuser mode.
file	Specifies the target file.
filePath	Specifies the new path.
audited	If specified, causes the operation to be audited so that it is redone in roll-forward recovery and replayed on SDS secondary systems.

Partition Actions

The partition actions that you can perform are described in the following subsections.

MarkOffline

The **MarkOffline** action marks a specified file partition offline (that is, officially absent). Marking a partition offline allows attempted accesses to objects in the offline partition to be distinguished from accesses to objects in a file or partition that has been accidentally moved or removed. The syntax of the **MarkOffline** action is as follows.

```
jdbadmin action=MarkOffline
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name part=partition-ID
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

MarkOnline

The **MarkOnline** action marks a specified file partition as online; that is, as present and accessible after the file or partition has been restored. Operator action may be required to ensure that partitions stored on archival media are located in the correct file system path. If required, use the **SetPartitionLocation** command to change the control file-defined location of a partition before marking it online.

The **MarkOnline** command fails if the partition is not found in its control file-defined location.

The syntax of the **MarkOnline** action is as follows.

```
jdbadmin action=MarkOnline
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name part=partition-ID
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

MovePartition

The **MovePartition** action moves the specified partition to the specified location.

If the destination of the move is on the same physical device as the source, the move is accomplished via a file system move (or rename). In this scenario, access to the file is blocked for the duration of the rename and meta-data updates. If the destination is on a different physical device, the move is accomplished using a backup copy and verify followed by removal of the source file.

When a non-frozen partition is moved to a different physical device, the **MovePartition** action blocks new transactions, acquires a database quietpoint, and freezes the partition for the duration of the operation; transactions are unblocked after the partition has been transitioned to a frozen state. The quietpoint is required to ensure there are no updates to the partition in the pipeline that would require an undo action should a transaction abort. Moving a partition required for object creation (any partition in the creation window) is prohibited.

The syntax of the **MovePartition** action is as follows.

```
jdbadmin action=MovePartition
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name part=partition-ID
        location=file-directory
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

The required **location** argument specifies the target path.

PurgePartition

The **PurgePartition** action executes a bulk-removal operation that performs the required side-effects of deleting each object in the partition, and then deleting the partition. The deletion step removes the file from the file system and changes its state to deleted. Because the objects and subobjects in the partition are not physically deleted, file maintenance and auditing overheads are avoided.

You can purge a current partition. If a purge is in action, a create or an update operation attempt fails and exception 3186 (*Operation not valid: partition being purged*) is raised. If this exception is raised, coordinate partition administrative actions with application availability requirements.

The object deletion side-effects that are affected include:

- Inverse maintenance
- Destructor execution
- Deletion of component aggregates that can be reached through parent-child relationships

The syntax of the **PurgePartition** action is as follows.

```
jdbadmin action=PurgePartition
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name [part=partition-ID]
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

RemovePartition

The **RemovePartition** action removes the specified partition. Before a partition can be removed, it must be empty. If required, use the **PurgePartition** command to delete all objects in the partition before it is removed.

The syntax of the **RemovePartition** action is as follows.

```
jdbadmin action=RemovePartition
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name [part=partition-ID]
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

SetPartitionLabel

The **SetPartitionLabel** action changes the label of the specified partition. The value of the label attribute can be changed at any time and does not affect the external file name.

The syntax of the **SetPartitionLabel** action is as follows.

```
jdbadmin action=SetPartitionLabel
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name [part=partition-ID]
        label=label-string
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

The required **label** argument specifies the new value to be assigned to the label attribute of the specified partition. If the label-string contains embedded spaces, you must enclose it within single quotes (') or double quotes (").

SetPartitionLocation

The **SetPartitionLocation** action changes the default or designated physical location of the specified offline file partition. The **location** argument for this action specifies the file system directory in which the offline partition will be located when it is marked online. The value of the **location** attribute is stored in the parent partition control file and is similar in function to the **path** attribute for map files, which is stored in the global database control file. If the location of a partition has never been set, it defaults to **null**; a value of **null** means that it is located in the directory of the parent partition control file. Relative paths are assumed to be relative to the database path.

The purpose of the **SetPartitionLocation** argument is to cause the database to open the file in the specified location. When the location is changed, the actual partition file must be in that location before it is marked online. A typical usage would be to allow offline partitions that have been burned to optical media to be mounted in a different file system directory (for example, a different drive).

The syntax of the **SetPartitionLocation** action is as follows.

```
jdbadmin action=SetPartitionLocation
        [path=database-path]
        [ini=initialization-file-name]
        [server=multiUser|singleUser]
        file=file-name [part=partition-ID]
        location=file-directory
```

The **path** argument is optional in multiuser mode.

The required **file** and **part** arguments specify the target file partition.

The required **location** argument specifies the file system directory in which the offline partition will be located when it is marked online.

This chapter covers the following topics.

- [What Is a Backup?](#)
 - [Why Are Backups Important?](#)
 - [When to Take Backups](#)
 - [Types of Failure](#)
- [Physical Database Structures](#)
 - [Database Files](#)
 - [Journal Files](#)
 - [Database Control File](#)
- [Database Identities](#)
- [Transaction Journal Files](#)
 - [Journal Switches and Journal Numbers](#)
 - [Quietpoints and Long Transactions](#)
 - [Checkpoints](#)
 - [Archived Transaction Journals](#)
 - [Database Archiving Modes](#)
 - [Automated Journal Close Actions](#)
 - [Unaudited Database File and Partition Operations](#)
- [Backups](#)
 - [Offline Full Backup](#)
 - [Online Quiesced Backup](#)
 - [Online Full Backup](#)
 - [Verification during Backup](#)
 - [Choosing a Backup Strategy](#)
 - [The Danger in Backing Up the Online Transaction Journal](#)
 - [Backing Up Your JADE Development Environment](#)
 - [Custom Backup Support](#)
- [Recovery Concepts and Strategies](#)
 - [Automatic Restart Recovery](#)
 - [Restoring and Recovering the Database](#)

- [Roll-Forward Recovery Restrictions](#)
- [Roll-Forward Recovery of a Standby Database](#)
- [Data Corruption](#)
 - [What Causes Data Corruption?](#)
 - [Detecting Data Corruption Problems](#)
 - [Performance Considerations](#)
- [Checking the Integrity of the Database Control File](#)
- [Database Certification](#)
- [Reindexing Database Files](#)
- [Evaluating Free-Space](#)
- [Compacting Files](#)
- [Resetting Timestamps](#)
- [Transient Database File Analysis](#)

See also Chapter 1, "[Administering a JADE Synchronized Database Service \(SDS\) Environment](#)", in the *JADE Synchronized Database Service (SDS) Administration Guide*.

What Is a Backup?

A database backup is simply a representative copy of data. When the original data is lost, you can use the backup to reconstruct lost information (that is, the physical files that constitute your JADE database). This copy includes important parts of your database such as the control file, transaction journals, and data files.

In the event of a catastrophic failure, your database backup is the key to successfully recovering your data. (For details, see "[Types of Failure](#)", later in this chapter.) Additionally, restoring and recovering a database from a backup can be operationally useful; for example, when moving or copying a database from one server to another.

By backing up a database from one computer and restoring and recovering the database from the backup to another computer, a copy of a database can be made quickly and easily.

As backing up and restoring a large database is a lengthy process, and backups of very large databases become impractical to perform on a regular basis for systems with high availability requirements, the Synchronized Database Service (SDS) provides an alternative recovery strategy. For details about using an SDS environment to keep secondary copies of a database synchronized with a primary updating copy and to provide read-only access to secondary copies of the database (for example, for the design and deployment of high workloads), see "[Administering a JADE Synchronized Database Service \(SDS\) Environment](#)", in Chapter 1 of the *JADE Synchronized Database Service (SDS) Administration Guide*.

Why Are Backups Important?

To establish that backups are important, consider the impact to revenue and customer satisfaction if your production database suddenly became unavailable, even for just five or ten minutes, or if data files were lost due to media failure and you could not restore or recover them because you did not have a backup. From the perspective of your enterprise, the results could be grim.

You must be able to restore and recover your data quickly to resume operations. The key to your success in this situation is a well-defined backup and recovery strategy.

When to Take Backups

Tailor your backup strategy to the needs of your business. For example, if it is acceptable to lose data in the event of a disk failure, you may not need to perform frequent backups.

What will happen if your database must be available twenty-four hours a day, seven days a week? In this case, you need to backup your database frequently.

The needs of your business primarily determine the frequency of your backups and the types of backups performed.

Types of Failure

An unfortunate aspect of every database system is the possibility of a system or hardware failure. The most common types of failure are as follows.

- Media failure, when one or more of the disk drives holding a database fails and you are faced with a complete loss of data unless you can restore and recover from a backup of the database.
- Application errors, when an application makes a large number of invalid modifications to data and the best way to deal with the problem may be to restore and recover the database from a backup to a specific date and time before the modifications were made.
- Software failure, when the operating system, device drivers, environmental software, or JADE itself has a problem that leads to file system or database corruption. For example, the File Allocation Table (FAT) file system is susceptible to various forms of corruption.
- Permanent loss of a server, when a server is disabled permanently or a site is lost due to a natural disaster and you may need to activate a warm standby server or restore and recover from a backup of the database on another server. For details about standby servers, see "[Roll-Forward Recovery of a Standby Database](#)", later in this chapter.

Physical Database Structures

The physical database structures are database files, transaction journal files, and the database control file. For details, see the following subsections.

Database Files

The logical entity representing a database file provides a convenient way of separating the user's view of data from some of the practical considerations associated with storing that data on disk.

The data in a database file is read during normal database operation and stored in memory cache, as required. Modified or new data is not necessarily written to a database file immediately or even when a transaction is committed.

Journal Files

The JADE database maintains details of updates to databases in a journal (or audit log) file, which is used to:

- Recover the database to a consistent state after a crash (crash or restart recovery)
- Undo the effects of failed or incomplete transactions (abort recovery)

A set of active, or *online*, journal files is maintained, as well as multiple offline (or archived) journal files. The active journal files record current database activity. As journal files become no longer required for abort recovery, they are released. When released, a journal is considered *offline*, and can then be moved to another location. For more details, see "[Journal Switches and Journal Numbers](#)" under "[Transaction Journal Files](#)", later in this chapter. JADE also supports the use of transaction journals to perform a roll-forward recovery. Roll-forward recovery takes advantage of the fact that transaction journals hold details of all changes that have been made to the database. You therefore do not necessarily need to undo changes but instead can reapply changes. See also "[Implementing a Database Recovery Strategy](#)", in Chapter 1 of the *JADE Synchronized Database Service (SDS) Administration Guide*.

JADE database recovery and the JADE Database utility journal dump and verify operations accept compressed journals as input. If an uncompressed journal is not found at the target location, a check for a compressed journal is performed. If a compressed journal is found, it is uncompressed for use as input to the operation. The temporary uncompressed file is removed when the operation completes.

Database Control File

The JADE database retains information about the physical structure of the database; for example:

- Timestamp and version information for each physical file
- Information about transaction journal files that may be used when recovering a database

This information is referred to as *control* data, and is stored in a special database file called the database control file (that is, the `_control.dat` file).

Database Identities

A JADE database has three Universally Unique Identifiers (UUID) values, representing the:

- Database identity
- Environment identity
- Server identity

In a default installation, all identity values are the same.

You can use these identities on the command line to specify the server Universal Resource Identifier (URI) target database and client-server transport. You can use this as an alternative to the **server** and **path** arguments in the command line and the **ServerNodeSpecifications** parameter in the [JadeClient] or [ConnectionParams] section of the JADE initialization file. The UUIDs are displayed in the:

- Messages in the JADE messages log file (**jommsg.log**) when the database server is initialized, as shown in the following example.

```
PDB: Database: {45cec8ae-2501-e111-8667-ca1120524153} successfully opened for
update, last update jomdb version: 1.0.0.3
PDB: Database Environment/Server identity: f28791cf-4d00-e111-9858-
ca1120524153/f2c369d4-4d00-e111-9703-ca1120524153
```

- Output of the **showInfo** command in the **jdbutilb** batch JADE Database utility, as shown in the following example.

```
JADE Database utility Version 7.1.0.3, Copyright (c) Jade Software Corporation
Limited 2014
Opening database ...
Database ID: 24389438-15d7-e011-82f0-5ae520524153
```

Attribute	Value
-----	-----
Last Database Mode:	Shared
Last Database Usage:	Update
Production Mode:	Disabled
Character Set:	UTF16 Little Endian
DB Code Version:	1.0.00.003
Highest transaction ID:	1329
Current journal:	4
Offline backup recovery LSN:	(4,3657620)
Environment Identity:	24389438-15d7-e011-82f0-5ae520524153
Server Identity:	24389438-15d7-e011-82f0-5ae520524153

- Output of the `System` class `getEnvironmentServerIdentity` method, which programmatically obtains environment and server identities (for example, 24389438-15d7-e011-82f0-5ae520524153/24389438-15d7-e011-82f0-5ae520524153).

Database Identity

The database identity is generated when the database environment is created and registered. Each database should have a unique value. If you clone a database from an existing one, you should give the new database a new unique identity, by using the `generateUUID` command in the `jdbutilb` batch JADE Database utility.

The database is given a new identity automatically when:

- The database role in an SDS environment changes from *undefined* to *primary* and back again,
- A hostile takeover is performed by a secondary database. This forces a re-clone of all copies.

Note All secondary databases in an SDS environment must have the same database identity as the primary. If you change the database identity of the primary, you must re-clone all of the secondaries.

Environment Identity

The environment identity uniquely identifies an environment and it is shared by all databases in an SDS environment. This value remains the same throughout the entire lifetime of the environment, unless you specifically change it.

When a database environment is first created, its environment identity is null. If the database server finds that the environment identity is null during initialization, it copies the database identity to the environment identity. If you clone a database from an existing one, give the new database a new unique environment identity, by using the `generateEnvironmentIdentity` command in the `jdbutilb` batch JADE Database utility.

Note All secondary databases in an SDS environment must have the same environment identity as the primary. When you change the environment identity of the primary, the change is automatically replayed on secondaries and it comes into effect when the database is restarted.

Server Identity

The server identity uniquely identifies a secondary database in an SDS environment, as its database identity and environment identity must be the same as the primary database.

When a database is first created, its server identity is null. If the database server finds that the server identity is null during initialization, it copies the environment identity to the server identity. If you clone a database from an existing one, give the new database a new unique server identity, by using the [generateServerIdentity](#) command in the **jdbutilb** batch JADE Database utility.

In an SDS environment when a secondary database is cloned from a primary database, the new database is automatically given a new server identity. If you clone a secondary from another secondary, the new database inherits the server identity of the source database, so you should generate a new server identity for the new database.

Transaction Journal Files

The transaction journal files are named **dbnnnnnnnnnn.log**, with the *nnnnnnnnnn* value being a number in the range **1** through **999999999**. If you have not specified the root directory for a recovery journal (by using the [JournalRootDirectory](#) parameter in the [\[PersistentDb\]](#) section of the JADE initialization file), the journal directory root defaults to **database-path\journals**.

The root journal directory contains **\current** and **\archive** subdirectories, as follows.

- The current directory contains those journals that are required to recover from a system failure.
- The archive directory contains offline journals and it exists if the [JournalArchiveDirectory](#) parameter is not specified in the [\[PersistentDb\]](#) section of the JADE initialization file.

Every JADE database has associated online transaction journals to protect the database in case the database experiences a failure.

The online transaction journals contain before and after images for database updates and control records indicating transaction and database state transitions.

Journal Switches and Journal Numbers

The point at which the database finishes writing to one online transaction journal and begins writing to another is called a *journal switch*. The JADE database engine supports the switching of auditing to a new journal file regardless of whether there are active transactions (that is, a quietpoint is not required for the switch to occur).

A journal switch is scheduled when the online transaction journal is completely filled and writing should continue to the next online transaction journal. When a transaction journal reaches its configured size, a journal switch is scheduled and processed by a worker thread. You can modify this behavior by using the [JournalSwitchInterval](#) parameter in the [\[PersistentDb\]](#) section of the JADE initialization file.

The database engine controls all required prior journals that may be needed should a transaction abort. (The undo processing of a transaction abort may require reverse traversal of multiple journal files.) As there is no forced checkpoint after switching to a new journal, the scope of restart recovery may encompass prior journal content. The [JournalRootDirectory](#) parameter in the [\[PersistentDb\]](#) section of the JADE initialization file contains the set of current journal files. As the database engine retains control over a journal file until there are no active transactions traversing it, a journal file is not closed when a switch to a new journal file occurs.

The database assigns the next *journal number* to the new online transaction journal when a journal switch occurs. All transaction journals (the online journals and all archived, or offline, journals) are uniquely identified by their journal number. During recovery, the database properly applies transaction journal files in ascending journal number order.

When archival recovery is enabled, the [\[PersistentDb\]](#) section [JournalCloseAction](#) parameter enables you to specify the action to be taken when a journal is closed and released.

You can support your own disaster recovery automation by using the **JournalSwitchInterval** parameter in the **[PersistentDb]** section of the JADE initialization file to specify that a transaction journal switches after a specified number of minutes, in conjunction with the size threshold specified in the **JournalMinSize** and **JournalMaxSize** parameters in that section. A time-based switch provides better control on how much processing is contained within a journal rather than only a size-based control mechanism.

The flexibility these parameters provide is shown in the following example.

```
JournalMinSize=0
JournalMaxSize=100
JournalSwitchInterval=60
```

The **JournalSwitchInterval** parameter value of **60** minutes in this example means that when the criterion specified by the **JournalMinSize** parameter has been met, the switch interval timer is honored. The interval timer is started during database initialization. As the **JournalMinSize** parameter is set to zero (**0**) in this example, the criterion is met every interval. If the application never generates 100M bytes of journal data within an hour, the journal will switch every hour.

If you find time-based switching useful but you do not want empty journals generated, set the **JournalMinSize** parameter to a value other than the zero (**0**) default. Setting the **JournalMinSize** parameter to **10** in this example would cause a journal switch attempt to occur every hour but to proceed only if the journal contained at least 10M bytes of journal data.

Quietpoints and Long Transactions

The database considers a user's transaction to be have two states. A kernel Begin Transaction instantiates a transaction for the user, marking the user as being in kernel transaction state. When the database processes the first activity for the transaction (for example, a put), the user is transitioned to being in database (DB) transaction state. It is when this transition from kernel transaction state to DB transaction state occurs that the Begin Transaction is audited, if auditing is enabled.

An active transaction is a transaction in which the user has transitioned to DB transaction state.

A *database quietpoint* is a point at which there are no active transactions. Database quietpoints are required when:

- Starting a non-updating reorganization
- Starting a reorganization on an SDS Primary system
- Relinquishing the Primary role in an SDE negotiated takeover
- Establishing a rollback point
- A **JadeDatabaseAdmin** class **commitCoherentBackup** method call
- Activating Delta mode
- Creating one or more file partitions
- Marking a file partition offline
- Moving a non-frozen file partition
- Starting the partitioning of a file

A *system quietpoint* is a point at which there are no users with transactions. System quietpoints are required when:

- Database mode is changed to archive by beginning a quiesced backup
- Database mode is changed to archive with **JadeDatabaseAdmin** class **changeDbAccessMode** method or a call to the **jomChangeAccessMode** JADE Application Programming Interface (API).
- Database usage is changed to no audit

The **MaxWaitForQuietPoint** parameter in the **[PersistentDb]** section of the JADE initialization file specifies the maximum time the database can wait for a database or a system quietpoint to be achieved. If a quietpoint is not achieved within the specified time, the database unblocks new transactions and exception 3077 (*Maximum time to wait for quiet point was exceeded*) is raised.

Checkpoints

Checkpointing is an activity that writes information to stable storage during normal database operations to reduce the amount of work required during restart recovery after a system failure.

The checkpoint procedure guarantees that at any time:

- The updates made by committed transactions that occurred prior to the checkpoint have been applied to the stable database during the last checkpoint, if not earlier

Use the **JadeDatabaseAdmin** class **doCheckpoint** and **getLastCheckpoint** methods to cause a database checkpoint operation to be queued to the database worker thread and retrieve the journal number and byte offset of the last database checkpoint, respectively.

Archived Transaction Journals

Automatic restart recovery after a system failure utilizes the online transaction journal or journals to recover the database to an operational state.

In a full recovery from a backup, all necessary transaction journal files are used. To recover the database to an operational state from the backup, all the archived (or offline) journal files created since the backup was performed are required, as well as the online transaction journal or journals.

Caution It is your responsibility to ensure that transaction journals are backed up. You may not be able to fully recover your database if you do not backup consecutively numbered journal files.

For protection against media failure in production systems, always locate the online and archived transaction journals on a different physical volume to the database files and mirror the transaction journal volume.

Database Archiving Modes

Use the **EnableArchivalRecovery** parameter of the **[PersistentDb]** section of the JADE initialization file to enable or disable archival recovery. By default, archival recovery is enabled. The default value of enabled (**true**) is the recommended value for productions systems. For non-critical systems, where transaction loss is acceptable, you can opt out of the ability to do archival recovery by setting the value of this parameter to **false**.

When you set this parameter to **false** and you perform an online backup, only the online transaction journals in the **!current** subdirectory are maintained. Any historical journal information is automatically deleted. This mode supports automatic restart recovery only, as only the most recent changes made to the database, stored in the online transaction journals, are available for recovery.

When archival recovery is enabled, archived (or offline) transaction journals are retained. For details, see the **JournalArchiveDirectory** and **JournalCloseAction** parameters under "Persistent Database Initialization Section **[PersistentDb]**", in the *JADE Initialization File Reference*.

Automated Journal Close Actions

You can automate the backing up of transaction journal files as they are closed, to ensure that this operation is not neglected. (The automating of the journal close action is ignored and unwanted journal files are removed when the **EnableArchivalRecovery** parameter in the **[PersistentDb]** section of the JADE initialization file is set to the default value of **false**.)

When archival recovery is enabled, use the **JournalCloseAction** parameter in the **[PersistentDb]** section of the JADE initialization file to specify one of the following close actions.

- **Copy**, which uses the operating system to copy the file to the archive directory
- **CopyAndCompress**, which programmatically creates a compressed copy of the file in the archive directory (using ZLIB compression mechanisms)
- **Move**, which uses the operating system to move the file to the archive directory
- **MoveAndCompress**, which programmatically creates a compressed copy of the file in the archive directory (using ZLIB compression mechanisms) and then removes it
- **None**, which takes no action

The journal close actions operate in conjunction with the **JournalArchiveDirectory** parameter, which specifies the destination directory for move and copy operations. The journal close action occurs when the database closes a journal. The database closes a journal when it releases it; that is, when it is no longer required for transaction abort processing or to restart recovery processing. (Switching journals, described under "**Journal Switches and Journal Numbers**", earlier in this chapter, is an independent activity to that of releasing journals.)

When you set the **JournalCloseAction** parameter to **Move**, the journals in the folder of active (live) journals are the current journal set and are held open by the database to prevent accidental removal. The current journal set comprises those journals required to back out the oldest transaction if it aborts or to perform recovery on restart after failure.

The number of the abort journal is not incremented. It changes based upon transaction activity, to reflect the oldest journal required for transaction abort or the restart recovery point established by the last checkpoint (whichever is farthest back in time). If there is no transaction activity, the abort journal number is not incremented.

Note This is a journal file *backup mechanism*, which is independent of the journal file shipping across a network connection used in conjunction with a hot standby server (documented in "Sample Usage Scenarios" in the *Synchronized Database Service (SDS)* white paper on the JADE Web site at <https://www.jadeworld.com/jade-platform/developer-centre/documentation/white-papers>).

Unaudited Database File and Partition Operations

When generating ad hoc indexes, for example, there is significant auditing overhead. Until the index build is complete, the database file or partition is known to be building (application preserved state) and if the build is restarted, the file or partition is first dropped.

Database files and partitions can be updated with auditing disabled, to eliminate journal disk space use and I/O overhead when loading data. When auditing is re-enabled for a file or partition, a copy of the file or partition is compressed by default, and inserted into the journal. During database roll-forward or replay, the file at the database location is replaced by the file reconstructed from the journal. Subsequent audited updates therefore replay correctly.

Caution Disable the auditing of database files and partitions only when restart recovery is not required.

The following table summarizes the methods used to manage unaudited operations.

Class	Method	Description
DbFile	disableAuditing	Disables the auditing associated with object operations performed against the file
DbFile	drop	Removes the file and marks it as deleted
DbFile	enableAuditing	Re-enables the auditing associated with object operations performed against the file
DbFile	isAuditing	Returns true if auditing associated with object operations performed against the file is enabled and returns false when auditing has been disabled
JadeDatabaseAdmin	doQuietpoint	Attempts to establish a database quietpoint
JadeDbFilePartition	disableAuditing	Disables auditing associated with object operations performed against the partition
JadeDbFilePartition	drop	Removes objects in the global partition index, removes the partition, and marks it as deleted
JadeDbFilePartition	enableAuditing	Re-enables the auditing associated with object operations performed against the partition
JadeDbFilePartition	isAuditing	Returns true if auditing associated with object operations performed against the partition is enabled and returns false when auditing has been disabled

In addition, the **DbFile** class provides the **EnableAuditNoCompress** class constant.

For details, see Volume 1 (that is, [EncycloSys1.pdf](#)) of the *JADE Encyclopaedia of Classes*.

Backups

A database backup can be a full offline backup, an online quiesced backup, or an online full backup. For details and a discussion about choosing a backup strategy, the danger of backing up the online transaction journal, and backing up your JADE development environment, see the following subsections.

For details about:

- The JADE initialization file parameters that enable you to configure your backups, see "Persistent Database Initialization Section [[PersistentDb](#)]", in the *JADE Initialization File Reference*.
- Using an SDS environment to keep secondary copies of a database synchronized with a primary updating copy and to provide read-only access to secondary copies of the database (for example, for the design and deployment of high workloads), see "[Administering a JADE Synchronized Database Service \(SDS\) Environment](#)", in Chapter 1 of the *JADE Synchronized Database Service (SDS) Administration Guide*.

The JADE Database utility backup compress operation uses temporary files (for example, ***.lo\$** and ***.da\$**) that can remain if the operation fails. You can safely delete these ***.lo\$** and ***.da\$** files. These temporary files are renamed if the operation succeeds (for example, to ***.lo_** and ***.da_**). The decompress operation also uses the dollar symbol (\$) file name variant before renaming the intermediate file to the actual target name when the operation succeeds. Overwrite and non-overwrite behavior applies to these temporary intermediate files.

Offline Full Backup

An *offline* backup involves shutting down the database server before starting the backup and then restarting the database after the backup is complete. As the database is shut down during the backup process, neither online users nor background processes can access the database for the duration of the backup. You must therefore schedule sufficient time to perform the backup, to ensure that the periods when the database is unavailable are acceptable to users.

You should use the JADE Database utility **Backup Database** command from the Operation menu to perform offline backups. This ensures that all necessary database files are backed up and it optionally permits verification and compression of files.

For details, see ["Using the Backup Database Command"](#), in Chapter 1.

Online Quiesced Backup

A backup that is performed while the database is in a read-only state is referred to as a *quiesced online*, or *warm*, backup.

The database is placed in a quiescent state by allowing current active transactions to complete and then flushing modified buffers from cache to the stable database.

In this mode, the physical database files contain all committed updates to the database and the database files are opened in read-only mode with shared read access, allowing external backup processes to safely copy the database files as a consistent database image.

During a quiesced backup, updating transactions are not permitted and attempts to execute database transactions raise a database exception.

When a backup is performed in a quiescent state, the physical database files are guaranteed to contain all database updates and a quiesced backup does not require backup recovery when the database files are restored.

Use the JADE Database Administration utility (**jdbadmin**) to perform online backups. For details, see Chapter 2, ["Using the JADE Database Administration Utility"](#).

You can also use the JADE database administration framework to integrate online backup services into your applications or to build standalone database administration applications. For details, see Chapter 7, ["Using the Database Administration Framework"](#), of the *JADE Developer's Reference*. Alternatively, the JADE **RootSchema** provides a database backup service that you can incorporate directly into any of your applications. For details, see ["Using the Online Database Backup Service"](#), in Chapter 7 of the *JADE Developer's Reference*.

Online Full Backup

A backup that is performed while the database is active for both read and write access is referred to as a *full online*, or *hot*, backup. As the database can be updated during the backup, the backed up data may be in an inconsistent state.

The transaction journal files must be retained so that the recovery process can recover the database to a fully consistent state.

Use the JADE Database Administration utility (**jdbadmin**) to perform online backups. For details, see Chapter 2, ["Using the JADE Database Administration Utility"](#).

You can also use the JADE database administration framework to integrate online backup services into your applications or to build standalone database administration applications. For details, see Chapter 7, "[Using the Database Administration Framework](#)", of the *JADE Developer's Reference*. Alternatively, the JADE Root Schema provides a database backup service that you can incorporate directly into any of your applications. For details, see "[Using the Online Database Backup Service](#)", in Chapter 7 of the *JADE Developer's Reference*.

Verification during Backup

When performing offline backups using the JADE Database utility (**jdbutil**), you can select or deselect the **Verify Files** and **Verify Checksums** check boxes on the Backup Database dialog, and your selections are retained across work sessions. The **Verify Files** option, when **true** (that is, checked), enables index block-to-target (object, subobject, and so on) verification, ensuring that index entries resolve to the correct records in the file. The **Verify Checksums** option, when **true** (that is, checked) means that file-level MD5 checksum verification is to be performed on each file after being copied.

The backup commands provided by the offline batch JADE Database utility (**jdbutilb**) and the online JADE Database Administration utility (**jdbadmin**) support the **verify** argument, which enables file-level checksum verification. Similarly, the **verifyFiles** parameter in the **JadeDatabaseAdmin** class backup methods enables file-level checksum verification.

By default, offline backups and online quiesced backups perform index block-to-target (object, subobject, and so on) verification. An online updating (non-quiesced) backup does not perform the same checks, as it cannot do so when files are being updated. As part of your recovery strategy, non-quiesced (that is, online updating) backups should be recovered to a temporary location where separate utility certification is performed to verify the backup contents.

Offline backup and online quiesced backup verification can be strengthened by setting the **BackupOrphanCheckDisabled** parameter in the [PersistentDb] section of the JADE initialization file to **false**. This causes target-to-index verification to occur.

Conversely, to make the verification weaker, set the value of the **BackupIndexCrosscheckDisabled** parameter in the [PersistentDb] section of the JADE initialization file to **true**, to disable the index-to-target verification. Where it is desirable to have an efficient backup process that fits comfortably in the processing window, you can set this parameter to **true** and then perform the JADE Database utility **certify** action after the backup, to certify the database using a temporary relocated (and checksum-verified) copy of the backup. Offline backup processing overheads are lower when no cross-checks are enabled, as data is not read into disk cache for use in verification.

Choosing a Backup Strategy

You must decide how you plan to protect your database against potential failures. Before developing your backup strategy, answer the following questions.

- Is it acceptable to lose any data if software or a disk failure damages some of the database files?

If it is not acceptable to lose any data, the database must be operated with archival recovery enabled, ideally with mirrored transaction journal volumes. Disk mirroring writes all information that is written to a primary disk to a secondary disk so that no data is lost if a disk failure occurs. For details about disk mirroring, see your operating system documentation.

If it is acceptable to lose a limited amount of data if there is a disk failure, you can operate the database with archival recovery disabled and avoid the work required to backup archive transaction journal files.

- Will you ever need to recover to arbitrary times in the past?

If you need to recover to a specific date and time in the past to correct an erroneous operational or programmatic change to the database, ensure that you run with archival recovery enabled.

- Does the database need to be available at all times (twenty-four hours a day, seven days a week)?

Do not operate the database without archival recovery enabled if the database must be available at all times, because the required whole database backups, taken while the database is shut down, cannot be made frequently, if at all. High-availability databases therefore always operate with archival recovery enabled and utilize online (or *hot*) backup capabilities.

Notes An online backup can be performed on a database that has archival recovery disabled. However, the usefulness of the backup is diminished, as it is not possible to recover any transactions completed since the backup was made following a restore from backup.

As backing up and restoring a large database is a lengthy process, and backups of very large databases become impractical to perform on a regular basis for systems with high availability requirements, the Synchronized Database Service (SDS) provides an alternative recovery strategy. For details, see "[Administering a JADE Synchronized Database Service \(SDS\) Environment](#)", in Chapter 1 of the *JADE Synchronized Database Service (SDS) Administration Guide*.

The Danger in Backing Up the Online Transaction Journal

There is no danger in backing up the online transaction journal. The danger is in accidentally restoring it without fully intending the consequences. Consider the following situation in which restoring a backed up online transaction journal severely compromises recovery.

When a crisis is in progress, it is easy to make a simple mistake. One of the dangers that system administrators frequently encounter is during a database restore. When restoring the whole database, it is possible to also accidentally restore the backed up online transaction journal, thus overwriting the current online transaction journal with the older (and less complete) version of the journal file. This forces the system administrator to perform an incomplete recovery when the intention was to perform a complete recovery, thus losing the ability to recover valuable transactions.

When writing is switched to a new journal, the journal *switched from* is marked read-only, to prevent it from being accidentally copied over. The **Restore Journals** command in the JADE Database utility (**jdbutil**) or the **restoreJournal** command in the batch JADE Database utility (**jdbutilb**) will not overwrite an existing file. (For details, see Chapter 1.)

Backing Up Your JADE Development Environment

In the JADE environment, you can backup your JADE development environment from a client workstation without having to sign off all users or bring down the server node.

You can perform online backups of your JADE development environment when the database is active for both read and write access.

When restoring a database backed up fully online (that is, the database was available for both read and write access at the time of the backup), the restore process requires the recovery of backed up transaction journals.

Alternatively, you can perform an online database backup with the database locked for write access so that records can still be read, although not updated. For details about the JADE initialization file parameters that enable you to configure your backups, see "Persistent Database Initialization Section [[PersistentDb](#)]", in the *JADE Initialization File Reference*.

» To backup your database

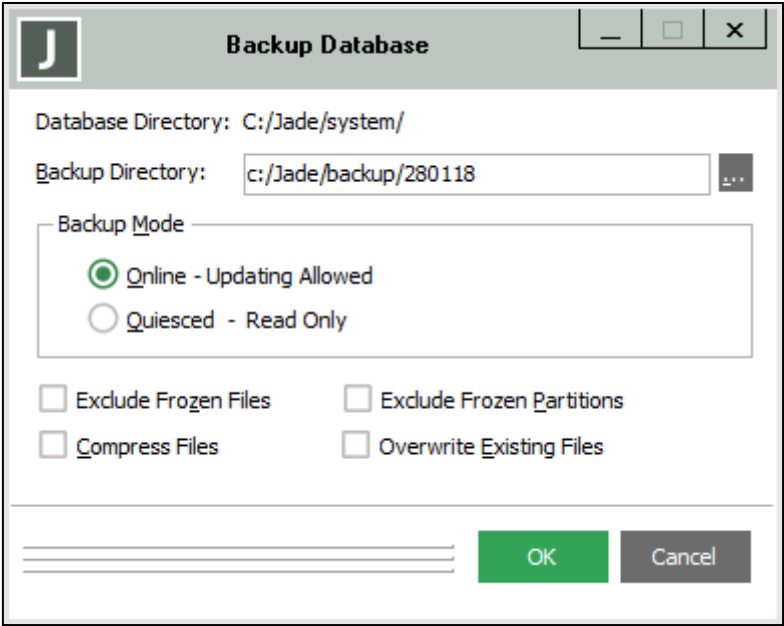
1. If you are currently running JADE, select the **Logoff** command from the File menu.

You are then logged off, and the JADE sign-on dialog is then displayed.

2. Select the **Administration** option button from the Select Options group box, to specify your installation preferences.

The JADE Installation Preferences dialog is then displayed, which enables you to specify the global options for your JADE development environment work session.
3. From the Admin menu, select the **Backup Database** command.

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



4. In the **Backup Directory** text box, specify the path of the directory to which your database files are to be backed up, as shown in the following example.

`d:/jade/system/backup`

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.
5. In the Backup Mode group box, select the **Quiesced – Read Only** option button if you want a quiesced read-only backup operation. By default, the **Online - Updating Allowed** option is selected, indicating that updates are allowed while the backup is performed. For details, see "[Online Full Backup](#)", earlier in this chapter.

When you select this option, the database is placed in a quiescent read-only state while the backup is performed. For details, see "[Online Quiesced Backup](#)", earlier in this chapter.
6. Check the **Exclude Frozen Files** check box, to exclude frozen (read-only) database files from the backup. By default, this check box is unchecked.
7. Check the **Exclude Frozen Partitions** check box, to exclude frozen (read-only) database partitions from the backup. By default, this check box is unchecked.
8. Check the **Compress Files** check box, to compress files as they are backed up. By default, database files are not compressed.

9. Check the **Overwrite Existing Files** check box, to overwrite any existing files in the specified destination backup directory as the database is backed up. By default, existing files are not overwritten.
10. 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 Installation Preferences dialog.

Custom Backup Support

The batch JADE Database utility (**jdbutilb**) enables you to perform database verification and checksum generation to support custom backup procedures. (For details, see ["The JADE Database Utility"](#), in Chapter 1.)

You can perform verification and checksum generation on a copy of your production database at a specific point so that you can then perform other actions to meet your requirements (for example, copying to tape).

The database content is verified (without opening the database) and a **restoreinfo** file is built containing calculated checksums. You could copy this **restoreinfo** file to tape with your database by using a mechanism of your choice, for example, and then restore the database with integrity checks, if required. For more details, see ["convertToBackup"](#) under ["Running the JADE Database Utility in Batch Mode"](#), in Chapter 1.

Recovery Concepts and Strategies

The JADE database backup and recovery mechanisms provide an important safeguard for protecting critical data stored in a JADE database. Restoring and recovering a database from a backup enables the complete restoration of data over a wide range of potential system problems.

It is important to distinguish between *restoring* a database and *recovering* a database. Restoring a database means bringing back the files that comprise the database from a backup or archive repository. The database at this point is not in an operational state.

Recovering a database means bringing the restored database to the point of being fully operational, which may involve restoring journal files and performing a roll-forward recovery, for example.

As backing up and restoring a large database is a lengthy process, and backups of very large databases become impractical to perform on a regular basis for systems with high availability requirements, the Synchronized Database Service (SDS) provides an alternative recovery strategy. For details, see ["Implementing a Database Recovery Strategy"](#), in Chapter 1 of the *JADE Synchronized Database Service (SDS) Administration Guide*.

When a database is restored, a **restoreinfo** file is created in the database directory. The **restoreinfo** file contains similar information to the **backupinfo** file and it includes a record for each file contained in the backup. When files or partitions are excluded from a database restore operation, they are marked as offline in the **restoreinfo** file. When the database is first opened following a restore, the **restoreinfo** file in the database directory is processed. Any files or partitions marked as offline that are not resident are marked as offline in their respective control files, and all control file latest backup timestamps, and latest full backup timestamps, are updated to reflect the backup that was restored.

During database restart recovery:

- When a recovery redo or undo operation encounters an error such as an index checksum error, the file is marked as having a broken index.
- Recovery continues to redo and undo object updates to files with broken indexes but stops applying most index updates (with the exception of index block creates).
- At the end of recovery, an automatic reindex operation is performed on all files with broken indexes.

See also ["Reindexing Database Files"](#), later in this chapter.

Automatic Restart Recovery

If your application terminates abnormally, the JADE object manager recovers automatically when it is next initiated.

Restoring and Recovering the Database

In the event of a media failure or an operational or software error destroying the database, you must bring back the files that comprise the database and perform recovery. To fully recover the database to a consistent operational state, the database files from the last backup, all archived transaction journals created since the backup, and the online transaction journal are required.

The database can be recovered to the end of the last journal or it can be recovered to a specific date and time to reverse the effect of unwanted modifications to data.

Notes Recovery to a specific date and time invalidates any existing later transaction journals. All transactions following that time are lost.

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.

The JADE Database utility Operation menu provides commands to facilitate database restoration and recovery, as follows.

- The **Restore Journals** command restores a numbered range of archived transaction journals from a specified directory.
- The **Restore Database** command provides a submenu that enables you to perform one of the following actions.
 - Restore your database with roll-forward recovery

Select the **Restore And Roll-Forward** command to restore a database from a JADE online or offline backup *and* perform a roll-forward recovery.
 - Restore your database with no recovery

Select the **Restore With No Recovery** command to restore a database from a JADE online or offline backup when you do not want a roll-forward recovery.
- The **Initiate Recovery** command performs a roll-forward recovery of previously restored offline backup files.

For operational details, see "[Using the Restore Journals Command](#)", "[Using the Restore Database Command](#)", and "[Using the Initiate Recovery Command](#)", in Chapter 1.

The starting journal file number, offset within the journal, and an internal creation timestamp are retained in the control file, which must be part of the backup so that a roll-forward recovery always starts from the correct journal. If the required transaction journal is not present, a dialog prompts you to restore the file from backup.

If the journal timestamp and required sequencing information are not correct, an error is reported and recovery does not proceed. These checks protect against operational errors; for example, the loading of the wrong journal files. Journal files contain the timestamps of the prior and next journal files. Each successive journal is checked to ensure that it has the correct timestamp and that it contains precisely the next required record in sequence (by date, time, and sequence number) before processing of the journal proceeds.

Database backup recovery and roll-forward recovery cannot be restarted. To keep the time taken to recover a database to a minimum, no checkpointing is performed during recovery. When the recovery operation is complete, a message in the following format is output to the **jommsg.log** file.

```
Database [secondary][archival|backup] recovery complete (elapsed time = nnn secs)
```

This message is logged after file data and headers are committed to disk. If the recovery is interrupted before this message is logged, the recovery is incomplete and the database must be restored before you restart the recovery process.

Roll-Forward Recovery Restrictions

Roll-forward recovery cannot be performed through a non-replayable reorganization. If a non-replayable database reorganization has occurred in a production database, take a full database backup *after* the reorganization. All database transaction journals created from the time of a backup are required to perform a roll-forward recovery.

Roll-Forward Recovery of a Standby Database

It may be desirable to maintain a copy of your database on another machine in standby mode. The standby copy of the database can be made quickly and easily, by backing up the database and then restoring and recovering it on another machine.

As transaction journals become archived (or offline) on your primary system, you can then copy them to the standby system and apply them to the database.

When a transaction journal switch occurs, a journal switch control record is written to the end of the journal. When this record is processed by roll-forward recovery and it is the final record in the transaction journal, the recovery process knows that later transaction journals can exist and the state is preserved so that a further roll-forward recovery from that point is possible.

When the batch JADE Database utility (**jdbutilb**) is run to recover the database, it processes the next journal and whatever later journals exist in the directory. However, on encountering a journal switch control record, the online JADE Database utility (**jdbutil**) waits for the next journal file to be provided or for the user to finish or abort the recovery processing. For more details, see "[Using the Restore Database Command](#)" and "[Using the Initiate Recovery Command](#)", in Chapter 1.

The online and batch versions of the JADE Database utility support continuing roll-forward recovery over multiple process executions. Any other open action of the database cancels the preserved state that enables continuous roll-forward recovery.

Data Corruption

This section contains the following topics that relate to the corruption of database files (for example, by the failure of hardware or the operating system).

- [What Causes Data Corruption?](#)
- [Detecting Data Corruption Problems](#)
- [Performance Considerations](#)

What Causes Data Corruption?

The JADE database engine depends on disk writes being atomic; that is, either all or none of a block is transferred to the disk media. This fundamental requirement can be jeopardized by hardware factors, particularly disk controller operational characteristics.

The disk controller may not complete in-progress writes when power loss occurs or the RST bus reset signal is raised. This may result in data corruption due to partially written and therefore incomplete, or torn, blocks.

Before JADE release 5.2, torn blocks were not detected during recovery and corrupted blocks remained undetected until used at some later time. As the JADE 5.2 release recovery algorithms introduced a high level of integrity checking, torn blocks affecting structural integrity can be detected and exhibit as **bTree** index block access and update errors. Torn blocks affecting object instances, however, can go undetected until used at some later time. In either case, the error handling and reporting mechanisms simply reflect the error state of the operation and the blocks that are parties to it, and can in no way determine that the root cause of the problem is a torn block on disk.

It has become evident that a significant number of JADE sites are not adhering to the hardware recommendations outlined in the *Environmental Considerations for Deploying JADE* white paper (available on the JADE Web site at <https://www.jadeworld.com/jade-platform/developer-centre/documentation/white-papers>). JADE systems deployed on servers with storage hardware that does not satisfy at least the minimum requirements documented under "Storage Requirements" in the white paper are susceptible to this form of disk corruption under system reset or power loss conditions.

Furthermore, we have determined that those requirements, while necessary, are insufficient to protect against incomplete disk write operations (that is, a "write through" caching policy by itself does not guarantee that write operations are atomic). Torn blocks are less likely to occur on disk drives attached to a controller that implements robust failure management mechanisms.

The controller must:

- Intercept and manage the RST bus reset signal and have on-board battery backup and mirrored or ECC memory
- Perform operational check-pointing so that when power is restored it can restart and complete the write operations that are in progress

Only high-end storage systems specifically designed to host mission-critical databases are likely to satisfy all of the requirements that guarantee write atomicity.

Despite the availability of robust storage subsystems, we have also observed that corruption can be injected by the I/O data path between main memory and the disk subsystem. When JADE issues a write I/O to disk, there are layers of software and hardware (for example, operating system, file system, I/O driver, data-bus, interface card, and cable) that handle the operation and data movement to the disk controller. Software bugs or faulty components in this I/O data path can introduce data corruption. Research has shown that this is an industry-wide issue with commodity hardware platforms that affects all database management systems.

For these reasons, we have implemented a software-level detection scheme that enables block corruption due to incomplete disk writes, bugs in data path software layers, or faulty data path components to be detected at the earliest opportunity.

Detecting Data Corruption Problems

JADE provides a block-level checksum mechanism so that corrupted blocks are detected when they are read from disk.

When an updated block is flushed from the disk cache, a block checksum is calculated and stored.

When a checksummed block is read into the database cache, the checksum is recalculated and compared. A mismatch raises a 3114 exception (*Checksum mismatch in fetched data*). When this exception is raised during normal processing, the affected data is inaccessible. While it is possible for you to continue operations with degraded access to your data, to reinstate data integrity you must restore your database from backup and perform a roll-forward recovery. (For details, see "[Using the Restore Database Command](#)", in Chapter 1.)

When a checksum error is detected during recovery processing, the recovery process is terminated. If you suspect that your database already contains corrupted blocks, you can verify structural integrity by performing a certification of your database using the JADE Database utility and verify referential integrity by running the JADE Logical Certifier. (For details, see "[The JADE Database Utility](#)" in Chapter 1 of this document, and Chapter 5, "[JADE Logical Certifier Diagnostic Utility](#)", of the *JADE Object Manager Guide*.)

As certifications are lengthy processes, we recommend that you perform these against your most-recent database backup. The database certification process verifies all checksummed blocks.

Performance Considerations

Maintaining checksums incurs a small performance overhead. Benchmark testing so far indicates that this overhead is minimal. However, as the impact of the checksum mechanism ultimately depends upon a variety of factors, including disk cache size, cache hits (working set), and the ratio of reads to writes, it varies for each environment.

Sites with tuned and appropriately sized disk caches will experience almost no overhead.

It is important to note that any overhead incurred due to this mechanism will be significantly less than having corrupted blocks going undetected for a long period. The longer the duration a corruption goes undetected, the more time it potentially takes to restore and recover the system and can, in turn, cause more-significant outages.

Checking the Integrity of the Database Control File

JADE provides the **dbcontrolcheck.exe** program, which you can use to check for duplicate and invalid entries in the database control file (**_control.dat**) and inconsistencies between the **DbFile** instances in the system and the database control file entries.

To check for errors, run the following program (for example, from a command script).

```
dbControlCheck path=database-path ini=initialization-file-name  
[server=singleuser|multiuser]
```

The default value for the **server** argument is **singleuser**.

Error 3035 (**DB_CERTIFY_ERROR**) is returned if errors are found, and these errors are reported to standard output and the **jommsg.log** file.

The program provides a command that enables you to fix the control file entries and the **DbFile** instances, where possible. To correct any errors that can be corrected, run the following program.

```
dbControlCheck path=database-path ini=initialization-file-name  
[server=singleuser|multiuser] applyFixes=true
```

Zero (**0**) is returned if the value of the **applyFixes** arguments is **true** and no unfixable errors have been found.

If any errors that cannot be corrected are found and your licenses include support, contact your local JADE support center or JADE Support with the log information for additional help.

Database Certification

The JADE Database utility Operation menu **Certify Files** command and the JADE Database Administration utility **certify** file or partition action enables you to perform a read-only check on the integrity of your database files. The database certification operation verifies the physical integrity of your database files. This process involves a series of scans that verify the contents of records and perform consistency cross-checks.

The certification of a database file involves three distinct phases or passes. A summary of the certification process, including statistics on the number of instances of objects and subobjects found in each scanned class and any errors that are detected, is reported to your **certify.log**. The log file output is prefixed with **process-id-thread-id**; for example:

```
04230-1850 2017/03/15 13:35:12.043 Certification of file: _userscm [31] commenced
```

You can perform certifications of multiple files of a database in parallel. This capability does not utilize worker threads within JADE; the approach taken supports multiple processes accessing the same database, where each process is certifying a different file or list of files. (A sharing access violation will occur if a certify process attempts to certify a file already being certified by another certify process.)

The JADE Database utility performs the following passes to certify your database files.

1. The file is read sequentially. The following table lists the verification performed for each block type during the sequential read.

Verify	Verifies...
Fixed Data Block	Block slot content and block meta data are valid and consistent
Head Block	Type and length are valid
Tail Block	Block content (types and lengths) are valid

As objects are encountered during block analysis, the corresponding index entry is retrieved and the file addresses matched.

2. The indexes are traversed in object identifier (oid) order. Consistency checks are performed on the indexes as each index block is read (for example, a check for illegal duplicates and that keys occur in the correct sequence).

For each object found in the indexes, the target object record is retrieved and the index and object record keys are compared for equality.

For details, see ["Using the Certify Files Command"](#), in Chapter 1, and ["Certify"](#), in Chapter 2.

Reindexing Database Files

The JADE Database utility Operation menu **Reindex Files** command enables you to repair damaged file indexes and perform free-space garbage collection serially.

The reindex file operation reads objects from a database (**.dat**) file serially and recreates them in a work (**.reo**) file, establishing and maintaining the indexes in the new file. If no errors are encountered, the reorganization work file is instantiated.

When a file is reindexed, any free-space in the file is consolidated, which provides an alternative mechanism to offline compaction for garbage collecting free-space in files.

In a post-recovery scenario, file reindexing is packaged and executed in the form of a replayable reorganization in which updates are disallowed. A journal switch is performed before starting a reindexing transaction, to provide a convenient journal boundary to facilitate a roll-forward recovery to splice out the reindexing transaction, if required, as a result of a mishap.

The value of the [ReorgWorkerThreads](#) parameter in the [\[JadeReorg\]](#) section of the JADE initialization file is observed, allowing multiple reindex operations to take place concurrently. The entire reorganization process, instantiation, validation, and committal of reindexed files is an atomic, idempotent (that is, remains unchanged if applied to itself), and recoverable process, which means that if the database server is terminated during the reindexing transaction before it completes and the database is restarted, the recovery process restores files to their pre-recovery state and the recovery process repeats the database file history including reindexing, as required.

As post-recovery automatic file reindex operations are journalled for replay:

- Roll-forward recovery replays any completed automatic reindex operations encountered during forward redo processing. Attempts to terminate recovery within an incomplete reindexing transaction are not permitted (the recovery fails with an exception).

Roll-forward recovery does not attempt to perform automatic reindexing as a post-recovery action, because any index corruption detected during roll-forward recovery is an indication of a potentially more-widespread issue that should not be masked. Such corruption would not be a result of a torn index block, as roll-forward recovery is supposed to be applied to a stable checked backup.

If roll-forward recovery encounters an error updating an index, recovery terminates with a fatal exception.

- The SDS secondary tracking process replays all automatic reindex transactions that completed successfully on the primary database. This also applies to any database files in a full or mapped extent RPS database that were reindexed on the primary database.

Note Initiating a file reindex operation on an SDS secondary database is not permitted.

To disable the automatic reindexing of database files, set the value of the [DisableAutoReindex](#) parameter in the [\[PersistentDb\]](#) section of the JADE initialization file to **true**. When you set this parameter to **true**, recovery is terminated and exception 3113 (*Database recovery not possible - fatal error encountered*) is raised when restart recovery encounters an error redoing or undoing an index update operation. The database must then be restored from backup and a roll-forward recovery performed. For details about performing a database file reindex operation, see ["Using the Reindex Files Command"](#), in Chapter 1.

Evaluating Free-Space

You can use the JADE Database utility Operation menu **Evaluate Free Space** command to analyze and report on the amount of free-space remaining in database files; for example, to check whether you need to compact your database files.

For details, see ["Using the Evaluate Free Space Command"](#), in Chapter 1.

Compacting Files

To optimize the use of storage and reduce internal file fragmentation, you should periodically compact your database files. Compacting a database file creates a copy of the file, rearranging how the file is stored on disk.

Compacted files are usually smaller than the original file but no smaller than configured in the [DefInitialFileSize](#) parameter of the [\[PersistentDb\]](#) section of the JADE initialization file.

A file compaction is a reorganization of a single database file in which no structural changes to objects are applied. Compacting the entire database compacts each file serially, so calculate the amount of free disk space for a full database compaction if the backup file (**.bak**) is to be on the same volume by doubling the size of the largest file that needs to be compacted in the database and add a little more for a few records in the journal.

If the compaction is non-updating, the space required for the compaction is the same regardless of whether the compaction is run offline or online. The following is an example of a non-updating compaction carried out in the code of a JADE method.

```
dbFile.changeAccessMode(DbFile.Mode_ReadOnly);  
dbFile.compactFile;  
dbFile.changeAccessMode(DbFile.Mode_Update);
```

If the online compaction is updating, the space requirement is the same as an offline compaction plus the overheads of create operations, because during the operation, creates in the original database file (**.dat**) have to be allocated from new space at the end of the file, therefore extending it. This new space cannot be calculated.

The file compaction process has the secondary effect of **bTree** index blocks being loaded more efficiently. The more-efficient loading of **bTree** index blocks potentially reduces the file space occupied by index blocks.

File compaction is an atomic process, in the same way that a normal database transaction is atomic. The file compaction process compacts to a working copy of the original file. If a file compaction is interrupted by a system or program failure, the original file is left unchanged.

When file compaction is complete, the original file is replaced with the compacted version. If an interrupted file compaction is restarted, the previous work file is removed and compaction restarts from the beginning.

For details, see ["Using the Compact Files Command"](#), in Chapter 1.

Resetting Timestamps

Use the JADE Database utility Operation menu **Reset Timestamps** command to override file timestamps after installing new system or user-defined schema files. (For details, see ["Using the Reset Timestamps Command"](#), in Chapter 1.)

When you upgrade a runtime-only installation of JADE at a site (when only methods or form definitions but no structures have been changed), if you do not reset the database file timestamps first, the database reports an error on start up as the file timestamps do not match those in the control file.

Caution Use the **Reset Timestamps** command with care and only when it is necessary to override timestamps when files have been replaced (for example, when deploying a new or patched release of a JADE application).

You cannot reset timestamps if database recovery is required.

To automate the process of resetting timestamps when replacing database files in a runtime deployment environment, you can execute the **jdbutilb** batch JADE Database utility executable program. (For details, see ["Running the JADE Database Utility in Batch Mode"](#), in Chapter 1.)

Transient Database File Analysis

Each transient database file is named using the following convention.

```
tdb_<host-name>_<pid>_<node-number><designator>.tmp
```

In this format:

- The `<pid>` value is the operating system process identifier and the `<designator>` value is one of the following.
 - `[kbp_nnnnnn]`, for transient files associated with a kernel background process
 - `[process-oid]`, for transient files associated with a user process
- The `<node-number>` value is zero (**0**) in normal circumstances.

You can identify the application process to which a transient database (TDB) file belongs by locating the process identifier in the `jommsg.log`; for example:

```
2014/02/22 14:23:26.897 000b8-0af4 Jom: Local process sign-on:oid=[187.2],
process=0x58f7fe8, no=2, id=2804, type=1 (non-prodn),scm=JadeSchema, app=Jade,
2016.0.0
```

In this log entry example, **[187.2]** is the oid for the process object.

Note The transient database files reside in the directory specified in the `TransientDbPath` parameter in the `[JadeClient]` section or `[JadeServer]` section of the JADE initialization file; for example, `c:\jadelsystem\tdb_WILBUR_00848_0[187.2].tmp`. (For details, see the *JADE Initialization File Reference*.)

The JADE Monitor displays information about transient files. You can also retrieve the information about the transient database file of a process by using the `Process` class methods summarized in the following table.

Method	Returns...
<code>analyzeTransientFileUsage</code>	A string containing a detailed analysis of the transient database file, including counts of objects by class number plus other useful information
<code>getTransientFileLength</code>	The physical size of the transient database file in use by the executing process
<code>getTransientFileName</code>	The name of the transient database file in use by the executing process

Alternatively, you can use the `Process` class methods listed in the following table to retrieve information about the transient database file of any process.

Method	Requests a process to send notifications containing...
<code>sendTransientFileAnalysis</code>	A detailed analysis of the contents of the transient database file
<code>sendTransientFileInfo</code>	The oid of the process, and the name and length of the transient database file

This chapter contains the following topics.

- [JADE Database Encryption Overview](#)
 - [Pre-requisite](#)
 - [Restrictions](#)
 - [Encryption Activity Audit Trail](#)
 - [Downgraded \(or Non-Existent\) Security](#)
 - [Strong Security](#)
- [Service Principal Names](#)
- [Running the JADE Database Encryption Utility](#)
 - [Encrypting a Database](#)
 - [Decrypting a Database](#)
 - [Command Arguments](#)
 - [Database Encryption Actions](#)

JADE Database Encryption Overview

Database encryption provides improved security of information held in the database. It makes it difficult to access the stored data except by means of approved access through the database engine.

An unauthorized user can examine the objects stored in an unencrypted database and obtain meaningful information using various file-editing tools, from as simple as Notepad through to a hex editor such as WinHex.

Backups of encrypted databases contain the encrypted data, so you do not require additional encryption to be reasonably confident that a misappropriated backup cannot be harvested for sensitive information without considerable effort and processing power.

The database has a mixture of encrypted and unencrypted files. The database administrator can choose which database map files to encrypt.

The default encryption algorithm used (that is, Advanced Encryption Standard (AES) with a 256-bit key) is provided by the Microsoft software stack (module CNG) and meets the requirements of United States Federal Information Processing Standards (FIPS) 140-2 level 2 certification.

Individual objects are decrypted when they arrive in the JADE Object Manager (JOM) from the database and are encrypted as they leave the JADE Object Manager for the database. The CPU time required for encryption and decryption is provided by the client node where the object is being used.

The database works solely with the encrypted copies of the objects. This saves database CPU time and also means that object images written to the journal and passed to and from client nodes do not require additional encryption.

Only the user-defined portion of each object is encrypted. The object record header, which contains fields such as the OID, is stored as plain text for use by the database engine. The encrypted database files also contain some plain-text records such as object index and freespace index blocks. These blocks have no user data.

Encryption is a two-step process.

1. An object-specific obfuscator is applied to the plain text, to break up repeated occurrences of the same character.
2. The encryption algorithm is applied, giving the cipher text to be stored.

Decryption reverses these two steps.

Each database map file has its own AES symmetric key, which is stored in the map file encrypted with the database master key. The master key:

- Is a 4096-bit RSA public/private key pair.
- Stored in the Microsoft persistent keystore independently from the database.
- Has a unique name generated from information stored in the database control file.

The database cannot be started if the correct master key cannot be read from the keystore.

Note You can determine whether the database encryption master key is present and correct, by calling the **System** class **verifyDbEncryptionMasterKey** method.

The master key is exported as readable encrypted text to disk file. It is encrypted using a separately entered passphrase. The file is designed to be printed so that if the electronic version is lost, the text can be typed into a text editor and saved to disk. The exported master key is required if the database is run on another machine (for example, a Synchronized Database Service (SDS)) or when the persistent keystore of the database machine is lost and must be reconstructed.

Caution If the database master key and all exported copies are lost, there is no way to recreate the key or to decrypt any encrypted database files.

Run the batch mode JADE Database Encryption utility **jdbcrypt**, to perform all database encryption-related functions. Most functions cannot be fully scripted, as they require entry of the master key passphrase using the keyboard.

You must run **jdbcrypt** from the same account as the database server so that the master key is stored in the correct account keystore.

Note We recommend that you run the database server under a non-system account.

When you enable database encryption, client-node authentication and authorization is also enabled. This requires a value to be specified in the **SspiAuthServicePrincipalName** parameter in the **[JadeClient]** section of the JADE initialization file on the client node. The simple form value for this entry is the account name used to run the database server.

The client node terminates with error 3574 if this parameter is not present. For more details, see "**Service Principal Names**", later in this chapter.

Notes New files added to databases using the **jadload** or **jadloadb** Schema Load utility are unencrypted, by default. You can use the optional **MandatoryFullEncryption** argument in the **EnableDatabaseEncryption** action to prevent objects from being added to new files before they are encrypted.

Mandatory full encryption does *not* cause files to be automatically encrypted. No files are automatically encrypted.

New map files added to a database are always added as unencrypted, regardless of the setting of the optional **MandatoryFullEncryption** argument. Should these files require encryption or the **MandatoryFullEncryption** argument is set to **true**, the files must be manually encrypted using the JADE Database Encryption utility (**jdbcrypt.exe**).

When **MandatoryFullEncryption** is set to **true**, it requires that all map files are encrypted before any attempt is made to create new objects in any map file, or exception 3345 (*DbCrypt Cannot create user objects in unencrypted files*) is raised.

Situations in which a database may be partially encrypted and where **MandatoryFullEncryption** should be set to **false** are as follows.

- Only certain map files contain data sensitive enough to require encryption
- Map files that can be created by the user logic are unencrypted and as such, would be unable to create objects

For more details about database encryption, see the following subsections.

- [Pre-requisite](#)
- [Restrictions](#)
- [Encryption Activity Audit Trail](#)
- [Access Check Options](#)

Pre-requisite

The encryption pre-requisite is as follows.

- The database server authenticates client nodes using the Windows SSPI Kerberos package, which requires both the server and clients to run with Active Directory-managed accounts.

Restrictions

The following restrictions apply when encrypting a JADE database.

- Encryption of single file instances of **JadeBytes** (Unstructured Data Resource (UDR)) is not supported.
- The **Object** class **autoPartitionIndex** method cannot be used if the database file for that object is encrypted, as the database cannot invoke the **autoPartitionIndex** method using an encrypted buffer.

If the file is encrypted, use the **Object** class **setPartitionID** or **setPartitionIndex** method to explicitly set the partition in the created object.

Encryption Activity Audit Trail

Each time the **jdbcrypt** program is run, it displays and logs the serial number and timestamp of the last update action.

When **jdbcrypt** successfully performs an update action, it updates the serial number and timestamp, and then displays and logs the new values.

Tip This provides you with an audit trail of database encryption activity.

Access Check Options

When you enable database encryption, you can specify the following values for the **AccessCheck** argument:

- Default (for details, see "[Default Security](#)")
- None (for details, see "[Downgraded \(or Non-Existent\) Security](#)")
- Strong (for details, see "[Strong Security](#)")

The **Default** option is used if you do not include the **AccessCheck** argument in the [EnableDatabaseEncryption](#) command.

Default Security

At this security level, the master key passphrase must be specified for the following **jdbcrypt** commands.

- [DisableDatabaseEncryption](#)
- [EncryptFile](#)
- [DecryptFile](#)
- [ExportMasterKey](#)

The master key passphrase is *not* required when the database server is started.

Downgraded (or Non-Existent) Security

You can enable database encryption so that a master key passphrase is never required by the **jdbcrypt** utility. This behavior allows encrypted databases to be created and destroyed in a fully scripted environment for testing and benchmarking.

Caution Use of this feature allows anyone who has access to the database user account to decrypt and encrypt the database.

When you enable database encryption with downgraded security, include the following optional argument on the **jdbcrypt** command line.

```
AccessCheck=None
```

You can suppress master key export and its associated request for a passphrase, by including the **MultipleExport=true** argument and omitting the **ExportPath** argument in the [EnableDatabaseEncryption](#) action.

When you encrypt or decrypt a file (by using the [EncryptFile](#), [EncryptFiles](#), [DecryptFile](#), or [DecryptFiles](#) action) with downgraded security, include the following argument on the **jdbcrypt** command line.

```
NoPassPhrase=true
```

Strong Security

You can enable database encryption so that the master key passphrase is required on every database use, including starting the database server.

This level can be used when a database with sensitive content is stored on a portable device such as a laptop and you want to prevent any unauthorized access. For maximum benefit, you should stop the database server each time you have finished accessing it, especially before hibernating a laptop or putting it to sleep.

Tip As a Windows-generated dialog is shown each time the master key is used with strong security, you should run the database server as a desktop application rather than as a service.

With strong security, the master key passphrase is not included as part of the export file data. When you use the **ImportMasterKey** action, you must enter the export file passphrase and then also enter the master key passphrase twice. It is not possible to force the same master key passphrase to be used.

When you enable database encryption, use a **jdbcrypt** command with the following syntax.

```
jdbcrypt path=database-path ini=initialization-file action=EnableDatabaseEncryption
AccessCheck=Strong ExportPath=file-path
```

You can also specify the **MultipleExport=true** argument for the **EnableDatabaseEncryption** action, but this allows anyone who knows the master key passphrase to export additional copies of the master key for abuse elsewhere. This is *not* recommended.

Service Principal Names

The Windows Security Support Provider Interface (SSPI) implementation requires that a client process use a Service Principal Name (SPN) to identify the server process that it is trying to authenticate. A JADE client process obtains the SPN of its database server from the **SspiAuthServicePrincipalName** parameter in the [JadeClient] section of the JADE initialization file.

The simplest form of SPN is the server process account name, but this means that the server process account name must be disclosed to the client processes (that is, put in the JADE initialization file of the client process).

The standard structure for a Service Principal Name is as follows.

```
service-class/instance-name:port-number/service-name
```

The *instance-name* value is (usually) a fully qualified domain name. You can obtain further discussion of the SPN format in MSDN, by searching for the topic Service Principal Name.

A suggested layout for JADE database server SPNs is as follows.

```
"JadeDB/server-host-fully-qualified-domain-name/database-name"
```

The *port-number* value is not used.

The inconvenient problem with *proper* SPNs is that they must be predefined in the Active Directory by a domain administrator. The **setSPN** Windows command line tool performs the SPN updates.

The SPN is defined on the database server account. If there is more than one matching SPN in the directory (for example, multiple accounts have the same SPN defined), authentication fails. If the database server is run with another account, either the client nodes must use a different SPN, which is defined on the new account, or the SPN must be moved to the new account.

The account can have multiple SPNs defined, but some part of each SPN must be unique. For example, the account could have an SPN for a primary database and also one for a secondary (run on another host), so that in the case of a failover, no Active Directory (AD) changes are required.

Windows does not enforce the SPN structure. Authentication does not require that the server process is run on a machine that is registered for the host name *instance-name*. The SPN is just a search key to find the server process account.

Running the JADE Database Encryption Utility

The JADE Database Encryption utility (**jdbcrypt.exe**), installed with your JADE software, is a batch utility in which all input values except passphrases are specified on the command line.

Notes As the JADE Database Encryption utility can be run only in single user mode, you must first shut down the database server.

The JADE Database Encryption utility must be run under the same account as the database server.

The **jdbcrypt** command line format is as follows.

```
jdbcrypt path=database-path
        ini=initialization-file
        action=ListStatus|EnableDatabaseEncryption|
               DisableDatabaseEncryption|EncryptFile|EncryptFiles|
               DecryptFile|DecryptFiles|ApplyPendingChanges|
               ClearPendingChanges|ExportMasterKey|ImportMasterKey|
               ListStoredKeys|DeleteStoredKey|help
        [optional-arguments]
```

Command actions have the following syntax.

ListStatus	ListStatus [ListPartitions=true]
EnableDatabaseEncryption	EnableDatabaseEncryption ExportPath= <i>file-path</i> [AccessCheck=Default Strong None] [MultipleExport=true] [MandatoryFullEncryption=true]
DisableDatabaseEncryption	DisableDatabaseEncryption [RetainMasterKey=true]
EncryptFile EncryptFiles	File= <i>file-name</i> File1= <i>file-name</i> File2= <i>file-name</i> ... Files= <i>file-name</i> ; <i>file-name</i> ;... [NoPassPhrase=true]
EncryptFile	MandatoryFullEncryption=true false [NoPassPhrase=true]
DecryptFile DecryptFiles	File= <i>file-name</i> File1= <i>file-name</i> File2= <i>file-name</i> ... Files= <i>file-name</i> ; <i>file-name</i> ;... [NoPassPhrase=true]
ApplyPendingChanges	ApplyPendingChanges
ClearPendingChanges	ClearPendingChanges
ExportMasterKey	ExportMasterKey ExportPath= <i>file-path</i>
ImportMasterKey	ImportMasterKey ImportPath= <i>file-path</i> [SuppressDatabaseAccess=true KeyFileName= <i>file-name</i>]
ListStoredKeys	ListStoredKeys [ListKeyDetails=true]
DeleteStoredKey	DeleteStoredKey StoredKeyName= <i>key-name</i>
help	help

The following is an example of the command for marking files for encryption. The targeted files are `_userscm`, `_userxrf`, `_usergui`, `_userint`, `_userdev`, `mydata`, and `shrdemo`.

```
jdbcrypt path=d:\jade\system ini=d:\salesdb\jade.ini action=EncryptFiles Files=_user*;mydata;shrdemo
```

Detailed results of the action are written to the appropriate log file in your specified database directory.

If the `jdbcrypt` executable program fails, a non-zero exit code is returned and an error message is displayed; for example, if the database directory was invalid.

Encrypting a Database

» To encrypt a database

1. Enable database encryption (using the `EnableDatabaseEncryption` action)
2. Mark database files for encryption (using the `EncryptFile` or `EncryptFiles` action)
3. Encrypt the marked files (using the `ApplyPendingChanges` action)

Tip Take a full offline backup of your database when the encryption process is complete, to avoid reprocessing the encryption during recovery.

Decrypting a Database

» To decrypt a database

1. Mark database files for decryption (using the `DecryptFile` or `DecryptFiles` action)
2. Decrypt the marked database files (using the `ApplyPendingChanges` action)

3. Disable database encryption (using the **DisableDatabaseEncryption** action)

If you run a secondary database on the same machine as the primary and you want to avoid recloning the secondary, specify **RetainMasterKey=true** so that there is no chance that the secondary restarts and requires the deleted master key before it replays the **DisableDatabaseEncryption** action.

Tip Take a full offline backup of your database when the decryption process is complete, to avoid reprocessing the decryption during recovery.

Command Arguments

The JADE Database Encryption utility command line argument values that apply to one or more database encryption commands are described in the following subsections.

action

The **action** argument specifies the database encryption function to initiate; for example:

```
jdbcrypt path=c:\jade\system ini=c:\jade\system\jade.ini
        action=EnableDatabaseEncryption ExportPath=c:\jade\system\crypt
        AccessCheck=Strong
```

An action of **help** or no action outputs help information. The **jdbcrypt** program **action** argument commands are listed in the following table.

Action	Description	Arguments
ListStatus	Lists the database and file encryption states	[ListPartitions=true]
EnableDatabaseEncryption	Enables database encryption	ExportPath= <i>file-path</i> [AccessCheck=Default Strong None] [MultipleExport=true] [MandatoryFullEncryption=true]
DisableDatabaseEncryption	Disables database encryption	None
EncryptFile EncryptFiles	Marks files for encryption	File= <i>file-name</i> File1= <i>file-name</i> File2= <i>file-name</i> ... Files= <i>file-name</i> ; <i>file-name</i> ... [NoPassPhrase=true]
EncryptFile	Modifies mandatory full encryption	MandatoryFullEncryption=true false [NoPassPhrase=true]
DecryptFile DecryptFiles	Marks files for decryption	File= <i>file-name</i> File1= <i>file-name</i> File2= <i>file-name</i> ... Files= <i>file-name</i> ; <i>file-name</i> ... [NoPassPhrase=true]
ApplyPendingChanges	Applies all pending encryption or decryption changes	None

Action	Description	Arguments
ClearPendingChanges	Clears all pending encryption or decryption changes	None
ExportMasterKey	Exports the master key	ExportPath= <i>file-path</i>
ImportMasterKey	Imports the master key	ImportPath= <i>file-path</i> [SuppressDatabaseAccess=true KeyFileName= <i>file-name</i>]
ListStoredKeys	Lists the persistent keys in the keystore	[ListKeyDetails=true]
DeleteStoredKey	Deletes a persistent key from the keystore	StoredKeyName= <i>key-name</i>

For details, see "Database Encryption Actions", later in this chapter.

path

The **path** argument specifies the database directory in which the control file (**_control.dat**) is located.

ini

The **ini** argument specifies the fully qualified name of the JADE initialization file. The default value is **jade.ini** in the database path directory.

Tip Always specify this parameter, to ensure that the correct JADE initialization file is used.

Database Encryption Actions

The database encryption **action** operations that you can perform are described in the following subsections.

ApplyPendingChanges

The **ApplyPendingChanges** action performs all pending encryption and decryption actions, using an offline reorganization-style behavior.

The [EncryptFile](#), [EncryptFiles](#), [DecryptFile](#), and [DecryptFiles](#) actions only mark the specified files as encrypt-pending or decrypt-pending; the file contents are not encrypted or decrypted until the next **ApplyPendingChanges** action. The syntax of the **ApplyPendingChanges** action is as follows.

```
jdbcrypt path=database-path
        ini=initialization-file-name
        action=ApplyPendingChanges
```

As this action does not require the master key passphrase, it can be fully scripted.

The following is an example of the **ApplyPendingChanges** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=ApplyPendingChanges
```

The **ApplyPendingChanges** action writes a secondary log file called **bulkcrypt.log**.

The contents of each database map file are copied to a **.reo** file in encrypted form. The pending keyset is moved into the current keyset slot in the **.reo** file. When all **.reo** files have been created successfully, instantiation replaces the original files with the encrypted files.

You can specify that multiple files are encrypted in parallel, by specifying the number of worker threads you require in the **ReorgWorkerThreads** parameter in the **[JadeReorg]** section of the JADE initialization file.

Tips You should not set the number of workers to more than the number of cores in the machine, minus one (for example, on a four-core machine, specify three workers). Consider the effect of multiple workers on the disk subsystem.

Take a full offline backup of your database when the encryption or decryption process is complete, to avoid reprocessing the encryption or decryption during recovery.

ClearPendingChanges

The **ClearPendingChanges** action clears all pending encryption or decryption changes.

As the **EncryptFile**, **EncryptFiles**, **DecryptFile**, and **DecryptFiles** actions only mark the specified files as encrypt-pending or decrypt-pending; you can clear encryption or decryption changes before they are applied to the database.

The syntax of the **ClearPendingChanges** action is as follows.

```
jdbcrypt path=database-path
         ini=initialization-file-name
         action=ClearPendingChanges
```

This action requires the manual entry of the master key passphrase.

The following is an example of the **ClearPendingChanges** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=ClearPendingChanges
```

DecryptFile and DecryptFiles

The **DecryptFile** and **DecryptFiles** actions mark the specified file or files as decrypt-pending.

The syntax of the **DecryptFile** and **DecryptFiles** actions are as follows.

```
jdbcrypt path=database-path
         ini=initialization-file-name
         action=DecryptFile|DecryptFiles File=file-name|
         File1=file-name File2=file-name ...|
         Files=file-name;file-name;...
         [NoPassPhrase=true]
```

The **DecryptFile** and **DecryptFiles** actions require the manual entry of the master key passphrase.

The following is an example of the **DecryptFile** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=DecryptFile File=cust
```

The *file-name* value is a complete map file name or a partial map file name with a trailing asterisk; for example, **cust*** means all map files that begin with the letters **cust**. Specify the *file-name* mask ***** (a single asterisk) to indicate decryption of all encrypted user files, including **rootdef**. Specify the mask **_*** (underscore asterisk) to indicate the decryption of all encryptable system files such as **_userscm**.

The following are examples of the **DecryptFiles** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=DecryptFiles Files=*  
  
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=DecryptFiles Files=cust*;  
userscm;archive*;  
  
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=DecryptFiles File1=testdb  
File2=banking File3=_userdev
```

The action fails if a complete file name does not match a database map file name. A file name mask with no matches generates a warning.

You can repeat these actions multiple times and you can interleave them with [EncryptFile](#) or [EncryptFiles](#) actions. The **DecryptFile** and **DecryptFiles** actions only mark the specified files as decrypt-pending; the file contents are not decrypted until the next [ApplyPendingChanges](#) action.

DeleteStoredKey

The **DeleteStoredKey** action deletes a persistent key from the keystore. The syntax of the **DeleteStoredKey** action is as follows.

```
jdbcrypt path=database-path  
ini=initialization-file-name  
action=DeleteStoredKey StoredKeyName='key-name'
```

The following is an example of the **DeleteStoredKey** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=DeleteStoredKey  
StoredKeyName='Cornelius Cephas'
```

DisableDatabaseEncryption

The **DisableDatabaseEncryption** action disables database encryption. The syntax of the **DisableDatabaseEncryption** action is as follows.

```
jdbcrypt path=database-path  
ini=initialization-file-name  
action=DisableDatabaseEncryption [RetainMasterKey=true]
```

The following is an example of the **DisableDatabaseEncryption** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=DisableDatabaseEncryption  
RetainMasterKey=true
```

Notes This action requires that all map files are decrypted first.

When you are sure that you do not need to restore an encrypted version of the database, you should destroy any copies of the old exported master key. If you re-encrypt the database, a new master key is always created.

This action requires the manual entry of the master key passphrase.

When you call this action to disable database encryption, the master key is automatically deleted from the keystore by default. This means that a secondary database restarted on the same machine as the primary becomes unusable if it has not replayed the file decryptions and disabled database encryption, because it requires the master key to reopen the database.

The JADE Database Encryption utility displays a message when the master key has been deleted.

If you want to override the default deletion of the master key and retain it when encryption is disabled, specify the **RetainMasterKey=true** argument for the **DisableDatabaseEncryption** action.

Caution If you run a secondary database on the same machine as the primary and you want to avoid recloning the secondary, specify **RetainMasterKey=true** so that there is no chance that the secondary restarts and requires the deleted master key before it replays the **DisableDatabaseEncryption** action.

When the secondary no longer requires the old master key, you can use the [ListStoredKeys](#) and [DeletedStoredKey](#) actions to find and delete it.

EnableDatabaseEncryption

The **EnableDatabaseEncryption** action enables database encryption. The syntax of the **EnableDatabaseEncryption** action is as follows.

```
jdbcrypt path=database-path
        ini=initialization-file-name
        action=EnableDatabaseEncryption
        ExportPath=file-path
        [AccessCheck=Default|Strong|None]
        [MultipleExport=true]
        [MandatoryFullEncryption=true|false]
```

Note You can specify the optional argument **MultipleExport** with a value of **true**, but this allows anyone who knows the master key passphrase to export additional copies of the master key for abuse elsewhere. This is *not* recommended.

The following is an example of the **EnableDatabaseEncryption** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=EnableDatabaseEncryption
ExportPath=d:\temp
```

Enabling database encryption generates the encryption Universal Unique Identifier (UUID), encryption timestamp, and the master key, and then updates the database control file with these details, to mark the database as encryption-enabled. It also exports the encrypted master key to a text file for separate secure storage.

You are prompted twice to enter the master key passphrase, which must be a minimum of 15 characters containing at least four letters, four digits, and two other characters; for example, **abcdefg+=123456**.

The title of the database master key is generated from the database encryption UUID and timestamp; for example:

```
Jade Software Corporation Limited.JADEDB.Master.b422f147-879d-01cb-1027-
00001d93f502.20101119035612199
```

By default, the master key is marked so that it can be exported once only from the keystore at the time it is created. A less-secure option is to specify the **MultipleExport** argument with a value of **true**, in which case the master key can also be exported at a later date.

The value of the **ExportPath** argument is the name of the directory to which the file containing the exported key is written. The name of the file is defined by the database encryption UUID and timestamp; for example:

```
JADEDB.Master.25146be6-8aca-01cb-1027-0000e329f802.20101123045153031.txt
```

The content of the file is plain ASCII text that can be printed as a backup to a copy of the file stored on removable media such as a pen drive (USB flash drive). You can make multiple copies of the exported key file. The file can be reconstructed by typing the printed contents into a text editor and saving it with the file name specified in the first line of text.

You must manually enter an export file passphrase, which is used to encrypt the master key before it is written. The export file passphrase, which should be different from the master key passphrase, has the same strength requirements.

Store the exported key file in a secured location and a sealed printed copy of the export file passphrase in another secure location.

Caution If you lose the exported master key or its export file passphrase, you cannot move the database to another machine or restore the contents of the keystore if it is lost or corrupted. In this case, you will be unable to access or decrypt the contents of the database.

When a new database map file is added to a fully encrypted database, the new file will be unencrypted, which will result in your database being partially encrypted if you forget to specifically encrypt the new file.

Note Mandatory full encryption does *not* cause files to be automatically encrypted. No files are automatically encrypted.

If you want mandatory encryption of *all* files in your database, specify the **MandatoryFullEncryption=true** argument (the default value is **false**). When this argument is set to **true**, the database server rejects attempts to create user objects in unencrypted user data map files. However, objects can be read, updated, and deleted.

New map files added to a database are always added as unencrypted, regardless of the setting of the optional **MandatoryFullEncryption** argument. Should these files require encryption or the **MandatoryFullEncryption** argument is set to **true**, the files must be manually encrypted using the JADE Database Encryption utility (**jdbcrypt.exe**).

When **MandatoryFullEncryption** is set to **true**, it requires that all map files are encrypted before any attempt is made to create new objects in any map file, or exception 3345 (*DbCrypt Cannot create user objects in unencrypted files*) is raised.

Situations in which a database may be partially encrypted and where **MandatoryFullEncryption** should be set to **false** are as follows.

- Only certain map files contain data sensitive enough to require encryption
- Map files that can be created by the user logic are unencrypted and as such, would be unable to create objects

Tip If you want to verify the encrypted status of all files in your database, run the **ListStatus** action regularly.

For details about the optional **AccessCheck** argument, see "**Access Check Options**", earlier in this chapter.

EncryptFile and EncryptFiles

The **EncryptFile** and **EncryptFiles** actions mark the specified file or files as encrypt-pending by generating the symmetric key for each specified map file, encrypting it with the master key, and storing it in the map file in the pending keyset slot.

The syntax of these **EncryptFile** and **EncryptFiles** actions are as follows.

```
jdbcrypt path=database-path
        ini=initialization-file-name
        action=EncryptFile|EncryptFiles File=file-name |
                                           File1=file-name File2=file-name ...|
                                           Files=file-name;file-name;...

[NoPassPhrase=true]
```

Alternatively, you can use the **EncryptFile** action to change the mandatory full encryption state, as follows.

```
jdbcrypt path=database-path
ini=initialization-file-name
action=EncryptFile MandatoryFullEncryption=true|false
[NoPassPhrase=true]
```

The **EncryptFile** and **EncryptFiles** actions require the manual entry of the master key passphrase.

The following is an example of the **EncryptFile** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=EncryptFile
File=annualsales
```

The *file-name* value is a complete map file name or a partial map file name with a trailing asterisk; for example, **cust*** means all map files that begin with the letters **cust**.

Specify the *file-name* mask ***** (a single asterisk) to indicate encryption of all user files, including **rootdef**. Specify the mask **_*** (underscore asterisk) to indicate the encryption of all encryptable system files such as **_userscm**.

The following are examples of the **EncryptFiles** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=EncryptFiles Files=*
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=EncryptFiles Files=cust*;*
userscm;archive*;
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=EncryptFiles File1=testdb
File2=banking File3=_userdev
```

Map files matching a mask and which are already encrypted are ignored.

The action fails if a complete file name does not match a database map file name. A file name mask with no matches generates a warning.

You can repeat these actions multiple times and you can interleave them with **DecryptFile** or **DecryptFiles** actions.

The **EncryptFile** and **EncryptFiles** actions only mark the specified files as encrypt-pending; the file contents are not encrypted until the next **ApplyPendingChanges** action.

When you encrypt a file with downgraded security, specify the **NoPassPhrase=true** argument to suppress the request for the master key passphrase. By default, the value of this argument is **false**.

When a new database map file is added to a fully encrypted database, the new file will be unencrypted, which will result in your database being partially encrypted if you forget to specifically encrypt the new file.

Note Mandatory full encryption does *not* cause files to be automatically encrypted. No files are automatically encrypted.

If you want to change mandatory encryption of *all* files in your database, use the **EncryptFile** action, specifying the **MandatoryFullEncryption=true|false** argument. When this argument is set to **true**, the database server rejects attempts to create user objects in unencrypted user data map files. However, objects can be read, updated, and deleted.

New map files added to a database are always added as unencrypted, regardless of the setting of the optional **MandatoryFullEncryption** argument. Should these files require encryption or the **MandatoryFullEncryption** argument is set to **true**, the files must be manually encrypted using the JADE Database Encryption utility (**jdbcrypt.exe**).

When **MandatoryFullEncryption** is set to **true**, it requires that all map files are encrypted before any attempt is made to create new objects in any map file, or exception 3345 (*DbCrypt Cannot create user objects in unencrypted files*) is raised.

Situations in which a database may be partially encrypted and where **MandatoryFullEncryption** should be set to **false** are as follows.

- Only certain map files contain data sensitive enough to require encryption
- Map files that can be created by the user logic are unencrypted and as such, would be unable to create objects

Tip If you want to verify the encrypted status of all files in your database, run the [ListStatus](#) action regularly.

ExportMasterKey

The **ExportMasterKey** action exports the master key. The syntax of the **ExportMasterKey** action is as follows.

```
jdbcrypt path=database-path
         ini=initialization-file-name
         action=ExportMasterKey ExportPath=file-path
```

The **ExportMasterKey** action requires the manual entry of the master key passphrase. This action fails unless the [EnableDatabaseEncryption](#) action included the **MultipleExport=true** argument and value.

The **ExportPath** argument value is the name of the directory to which the file containing the exported key is written. The name of the file is defined by the database encryption UUID and timestamp; for example:

```
JADEDB.Master.25146be6-8aca-01cb-1027-0000e329f802.20101123045153031.txt
```

The following is an example of the **ExportMasterKey** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=ExportMasterKey
ExportPath=d:\jade\security\prodsales
```

ImportMasterKey

The **ImportMasterKey** action imports the master key into the Windows account keystore. The syntax of the **ImportMasterKey** action is as follows.

```
jdbcrypt path=database-path
         ini=initialization-file-name
         action=ImportMasterKey ImportPath=file-path
         [SuppressDatabaseAccess=true KeyFileName=file-name]
```

The **ImportMasterKey** action requires the manual entry of the master key passphrase.

The **ImportPath** argument value is the name of the directory from which the file containing the imported key is read. The name of the file is defined by the database encryption UUID and timestamp; for example:

```
JADEDB.Master.25146be6-8aca-01cb-1027-0000e329f802.20101123045153031.txt
```

If you need to import the master key when the database does not have encryption enabled (for example, during roll-forward recovery through the encryption action), include the optional **SuppressDatabaseAccess=true** and **KeyFileName=file-name** arguments.

The value of the **KeyFileName** argument is the name of the file without the path. The specified **ImportPath** directory is searched for the specified file.

The following is an example of the **ImportMasterKey** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=ImportMasterKey
ImportPath=z:\jade\security\sales
```

ListStatus

The **ListStatus** action lists the database and file encryption states.

The syntax of the **ListStatus** action is as follows.

```
jdbcrypt path=database-path
ini=initialization-file-name
action=ListStatus [ListPartitions=true]
```

The following is an example of the **ListStatus** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=ListStatus
```

The database and file encryption states are output to the console (that is, to **stdout**). For details about displaying and redirecting JADE batch utility output, see the [DisplayApplicationMessages](#), [LogServer](#), and [UseLogServer](#) parameters under "JADE Log Section [[JadeLog](#)]", in the *JADE Initialization File Reference*.

The following is an example of the output when database encryption is enabled but there are no encrypted files.

```
JADE Database Encryption Utility Version 7.1.0.0 Copyright (c) Jade Software
Corporation Limited 2014
```

```
#Database and File encryption status
Opening database directly ...
Database UUID: 48cf13de-b6fd-df11-87e2-2e5920524157
Last update was #1 at 2014/01/11 17:10:55.578
Encryption UUID: 81bca195-b145-01cb-1037-000056faf402
Encryption Timestamp: 2014/01/11 04:10:40.767 (UTC)
MasterKey is valid
```

```
FILE NAME
tdb _TDB Plaintext
11 _environ Plaintext
31 _userscm Plaintext CurrentCI=none PendingCI=none
32 _userxrf Plaintext CurrentCI=none PendingCI=none
33 _usergui Plaintext CurrentCI=none PendingCI=none
34 _userint Plaintext CurrentCI=none PendingCI=none
35 _userdev Plaintext CurrentCI=none PendingCI=none
40 _rootdef Plaintext CurrentCI=none PendingCI=none
51 dbcryptttesting002 Plaintext CurrentCI=none PendingCI=none
52 dbcryptttesting003 Plaintext CurrentCI=none PendingCI=none
53 par493 Plaintext CurrentCI=none PendingCI=none
88 dbcryptttesting Plaintext CurrentCI=none PendingCI=none
190 dbcryptttesting001 Plaintext CurrentCI=none PendingCI=none
```

The following is an example of the output when database encryption is enabled and all user data map files are encrypted.

```
JADE Database Encryption Utility Version 7.1.0.0 Copyright (c) Jade Software
Corporation Limited 2014
```

```
#Database and File encryption status
```



```
Opening database directly ...
Database UUID: 48cf13de-b6fd-df11-87e2-2e5920524157
Last update was #1 at 2014/01/11 17:10:55.578
Encryption UUID: 81bca195-b145-01cb-1037-000056faf402
Encryption Timestamp: 2014/01/11 04:10:40.767 (UTC)
MasterKey is valid
```

FILE NAME		
tdb _TDB	Plaintext	
11 _environ	Plaintext	
31 _userscm	Plaintext	CurrentCI=none PendingCI=none
32 _userxrf	Plaintext	CurrentCI=none PendingCI=none
33 _usergui	Plaintext	CurrentCI=none PendingCI=none
34 _userint	Plaintext	CurrentCI=none PendingCI=none
35 _userdev	Plaintext	CurrentCI=none PendingCI=none
40 _rootdef	Encrypted	CurrentCI=present PendingCI=none
51 dbcryptttesting002	Encrypted	CurrentCI=present PendingCI=none
52 dbcryptttesting003	Encrypted	CurrentCI=present PendingCI=none
53 par493	Encrypted	CurrentCI=present PendingCI=none
88 dbcryptttesting	Encrypted	CurrentCI=present PendingCI=none
190 dbcryptttesting001	Encrypted	CurrentCI=present PendingCI=none

Tip Run the **ListStatus** action regularly, to verify the encrypted status of all files in your database.

ListStoredKeys

The **ListStoredKeys** action lists the persistent keys in the keystore. The syntax of the **ListStoredKeys** action is as follows.

```
jdbcrypt path=database-path
         ini=initialization-file-name
         action=ListStoredKeys [ListKeyDetails=true]
```

The following is an example of the **ListStoredKeys** action.

```
jdbcrypt path=d:\dbcrypt ini=d:\salesdb\jade.ini action=ListStoredKeys
```

The stored-keys (and optionally the stored key details, when you specify **ListKeyDetails=true**) are output to the console (that is, to **stdout**).