

# Upgrading to JADE Release 7.0 Consolidated Release Information

VERSION 7.0.10

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 © 2014 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
<b>Upgrading to JADE Release 7.0.10 .....</b>	<b>8</b>
JADE Release Support .....	9
Deimplementations and Deprecations .....	9
ActiveX Exposure Deimplementation .....	9
Compact JADE Single User Mode .....	9
Database Rebuild Files Deimplementation .....	9
JADE Dump and Load Utility Deimplementation .....	9
JADE on Linux .....	9
JADE Portable Graphical User Interface (GUI) Client .....	10
Rational Rose Deimplementation .....	10
System::getDbObjectCacheStats Method .....	10
Web Service Framework Jade61 Application Type .....	10
Advanced Notice of JADE 7.1 Changes .....	10
32-bit Presentation Client Upgrade .....	10
Java Framework Deprecation .....	10
Server Read-Only User Deprecation .....	10
Silverlight XAML Deprecation .....	10
String Arrays and Dictionary Keys .....	10
Accessing Details about Faults Fixed in Releases .....	12
How to Locate PARs Fixed in a Specific Release .....	12
Upgrading from an Earlier JADE 7.0 Release or JADE 6.3.05 Release or Higher .....	13
Running the Upgrade .....	13
Running Two Releases of JADE on the Same Workstation .....	16
High-Availability Upgrade .....	16
JADE Thin Client Upgrade .....	16
Upgrading a 32-Bit Presentation Client Connecting to a 64-Bit Application Server .....	17
Upgrading a Synchronized Database Environment (SDE) .....	17
Upgrading an RPS Node from JADE 6.3 to 7.0 .....	17
Upgrade Validation .....	17
Hot Fix Releases .....	17
Changes in JADE Release 7.0.10 .....	19
.NET Changes .....	19
.NET Class Library (PAR 60740) .....	19
JADE .NET Developer's Reference .....	19
Adding User Classes at Run Time .....	20
AutoComplete Functionality .....	20
AutoComplete Selection from a List (PAR 61330) .....	20
Pseudotype Variables (PAR 61647) .....	20
Binary and String Length (NFS 4822) .....	20
Class Persistent Instances Methods (PAR 61167) .....	21
Extract Sort Exception (PAR 61403) .....	21
JADE Initialization File Parameter Handling (PAR 61704) .....	21
JADE Logical Certifier (PAR 61172) .....	21
OpenSSL Library Version Upgrade (PAR 61368) .....	21
Thin Client Automatic Download (PAR 61337) .....	22
Changes in JADE Release 7.0.09 .....	23
AutoComplete Feature (PAR 60436) .....	23
C# Exposure Wizard .....	23
ExternalCollection Subclasses (PAR 60285) .....	23
Disk Write-Cache Messages (PAR 60863) .....	23
Dynamic Property Cluster Class Map Change (PAR 60706) .....	24
Generated JADE .NET Names (NFS 60673) .....	24
Inherited Control Event Method Display (PAR 60398) .....	24
Interface Stub Methods (NFS 60359) .....	24
JADE Initialization File .....	24
AcceptZeroEnvironmentUUID Parameter .....	24
BackupCompressedFileGrowthIncrement Parameter (PAR 60653) .....	25

DiskCacheWriteOnlyDbSegments Parameter (PAR 60707)	25
JADE Server Section Parameter Values (PAR 60842)	25
JADE Logical Certifier (PAR 60649, 60505, 60506)	25
JADE Painter	26
Integer Precision (PAR 60470)	26
Menu Design (NFS 2495)	26
JADE Version Information Utility (PAR 60988)	26
Java Exposure Wizard (PAR 59510)	26
Methods View Browser	27
JADE Scripts and Test Cases in the Methods View Browser (PAR 60442)	27
Methods View Browser Context Menu (PAR 60947)	27
ODBC Driver Query Execution Timeout (NFS 60759)	27
Replayable Reorganization (NFS 60428)	27
RPS Database Creation (NFS 53778)	28
Save Method Message Box (NFS 60438)	28
StringUtf8Array Elements (PAR 60605)	28
Changes in JADE Release 7.0.08	29
.NET Requirements (PAR 60253)	29
App Object in .NET Applications (PAR 59732)	29
Application Browser (NFS 59947)	29
Assigning an Object to a Partition (PAR 59567)	29
AutoComplete Functionality	30
Hiding the Display of RootSchema Entities (NFS 60118)	30
Translatable Strings (NFS 59442, PAR 59673)	30
Cleaning up Virtual Directories (NFS 60054)	30
Compiling Method Constants (PAR 60173)	31
Compressing Database Files (PAR 59324)	31
Database Control File Integrity Checking (PAR 49652)	31
DisableRPSRedintegrateOnRestart Parameter Default Value	32
Disabling Database Encryption (PAR 59650)	32
drawFillStyle Property Background (PAR 60265)	32
Enabling or Disabling the read Instruction (NFS 59963)	32
Getting System Statistics (PAR 60021)	33
JADE Command Files Creation (NFS 60055)	33
JADE Painter Menu Design (NFS 48839)	34
JadeAuditAccess::loadDescriptionByName Method (PAR 55770)	34
Merge Iterator Functionality in the .NET JADE API (NFS 60352)	34
Moving Object Instances in an Encrypted Database (PAR 59648)	34
Parameters Specified in the Command Line (PAR 59207)	34
Process Usages	35
Including Subclasses in the Process Usages Display (PAR 60146)	35
Multiple Processes Running under the Same User Code (PAR 60290)	35
Removing a Class from a Schema View (NFS 59950)	35
Silverlight Painter (PAR 54312)	36
Support for HTML-based Online Help	36
Thin Client Upgrade (NFS 59839)	37
TimeStampInterval::set Method (PAR 58981)	37
Transient Leak Analysis	37
Displaying Transient Leaks in a Methods Browser (NFS 59913)	37
Excluding Code from Transient Leak Analysis (NFS 59911)	37
Locating Possible Transient Leaks (NFS 59910)	38
Unit Tests	38
Failure Count (PAR 59549)	38
Unit Test Method Options in an Abstract Class (PAR 59914)	38
Visibility of Loaded Schemas (PAR 58840)	39
Changes in JADE Release 7.0.07	40
About Form of an Application (PAR 57199)	40
Application Browser (NFS 44762)	40
Binary Length (PAR 59558)	41
Date and Time Logged Formats	41
Date Format (PAR 58201)	42
Extracting all Exposures (PAR 58995)	42
Function Keys (PAR 59439)	42
Interface Implementation (PAR 59129)	42
JADE Debugger (NFS 59455)	43

JADE Logical Certifier Detection of Abstract Class Instances (PAR 59203)	43
JADE Monitor	43
Additional Information in the Users View (NFS 21502)	43
JADE Monitor Default View (PAR 51595)	43
JADE Serial Port (PAR 58961)	43
JadeBytes Instances (PAR 58990)	44
Loading Forms Definition Files (PAR 56094)	44
Local Implementors (PAR 53510)	44
MDI Window List Order (NFS 59101)	44
Printing	44
Opening a Printer (PAR 58242)	44
Printing Draw Fill Styles (PAR 59082)	44
Purging a Partition (PAR 59558)	45
Schema Inspector	45
Inspecting Collections (NFS 59009)	45
Inspecting Dynamic Objects (PAR 59423)	45
Inspector Forms (NFS 59331)	45
Search Functionality (PAR 54465)	46
Web Service Consumer Generate Test Case Dialog (PAR 53623)	46
Relationship View Diagram Printing (PAR 61264)	46
Changes in JADE Release 7.0.06	47
C# Exposure and .NET	47
.NET Exposure Generation (NFS 58838)	47
Format of the Generated Source (NFS 58847)	47
Converting a Partitioned Database (PAR 58782)	48
Database Administration	48
Multiple Workers for Database Compaction (NFS 48609)	48
JadeDatabaseAdmin::compactDbFiles Method	48
Jade Database Utility Compact Operation	48
JADE Database Administration Utility Compact Action	48
JADE Monitor (PAR 57802)	49
Transient File Compaction (PAR 57802)	49
Dynamic Property Methods	49
compactDynamicPropertyClusters Method	49
deleteDynamicPropertyCluster Method	50
Dynamic Property Clusters in Partitioned Database Files (PAR 58866)	50
Database Production Mode (PAR 58555)	50
Development Environment MDI Styles	51
Disk Cache Maximum Segments (PAR 58521)	52
Extracting a Method (PAR 58472)	52
Hierarchy Browser History (PAR 58906)	52
JADE Logical Certifier Utility (NFS 51392)	53
JADE Version Information	53
Schema Inspector (PAR 54665)	53
Silverlight XAP File Generation using jadclient (PAR 58614)	54
Synchronized Database Service (PAR 56965, 57901)	54
Thin Client Automatic Download (PAR 57876)	54
Windows	55
Draw Fill Styles (NFS 58716)	55
Drawing Form and Control Elements	58
Skins	58
Menus	58
Button Control	58
CheckBox Control	58
OptionButton Control	59
Changes in JADE Release 7.0.05	60
Database Identities	60
Database Identity	60
Environment Identity	61
Server Identity	61
Format of the Server Uniform Resource Identifier (URI) String	61
File Scheme (Single User)	62
TcpIp, TcpIpv4, and TcpIpv6 Schemes	62
Hybrid Pipe Shared Memory (HPSM) Scheme	63
JadeLocal Scheme (Shared Memory)	63

Debugger Initialization (PAR 57923)	63
Development Environment	63
AutoComplete	63
Colon Character Recognition (PAR 58199)	63
Default AutoComplete Setting (PAR 58351, NFS 57721)	64
Recognition of Parameter Usage Types (PAR 57358)	64
JADE Painter	64
Loading a Form into the JADE Painter (NFS 47177)	64
Menu Design Form (NFS 46595)	64
Refactoring	64
Declaring an Unknown Local Identifier as a Local Variable (NFS 58186)	65
Extracting Existing Logic and Creating a New Method (NFS 58231, PAR 58351)	65
Generating a Method Stub from the Editor Pane (NFS 58158, PAR 58389, PAR 58388)	68
Promoting a Local Constant to be a Type Constant (NFS 58179)	69
Promoting a Local Variable to be a Parameter of the Current Method (NFS 58190, PAR 58390)	69
Promoting a Local Variable to be a Property on the Current Class (NFS 58174)	70
Promoting a Method Value to be a Local Method Constant (NFS 58182)	70
Renaming or Changing an Entity from the Editor Pane (NFS 58129)	70
Search Functionality	72
Finding an Entity in a Schema View (NFS 43327, 43794)	72
Finding an Entity in the Schema in which the Class is Defined (NFS 49367)	72
Locating Possible Transient Object Leaks (NFS 58062)	73
Locating Unused Local Variables and Parameters (NFS 58066)	74
Search and Replace Dialogs (NFS 38838, 58078)	75
Unit Tests (NFS 58045)	76
Disk Cache Memory (PAR 57865, 58354)	76
Dynamic Properties	76
Benefits of Dynamic Properties	76
Dynamic Property Classes, Properties, and Methods	76
Restrictions	76
Limitations in the Current Release	77
Object::getPropertyValue Method Change	77
Object::setPropertyValue Method Change	77
Object::tryGetPropertyValue Method	78
Object::deletePropertyValue Method	78
Class::addDynamicPropertyCluster Method	78
Class::addDynamicProperty Method	78
Class::findDynamicPropertyCluster Method	79
JadeDynamicPropertyCluster Class	79
JadeDynamicPropertyCluster::addDynamicProperty Method	79
JadeDynamicPropertyCluster::addExclusiveDynamicProperty Method	80
JadeDynamicPropertyCluster::deleteDynamicProperty Method	80
JadeDynamicPropertyCluster::findDynamicProperty Method	80
Property Class Methods	80
Property::addDynamicInverse Method	80
Property::changeDynamicProperty Method	81
Property::changeExclusiveDynamicProperty Method	81
Property::removeDynamicInverse Method	81
Example Usages	82
Exception Handling (NFS 52373)	82
Installed Thin Client Download Files	83
JADE Initialization File (PAR 57781)	83
Logical Certifier (PAR 57908)	83
Monitor Display of the Node Edition (NFS 56397)	83
Multiple Database Access for .NET	84
Restrictions	84
Using the Multiple Database Access Support Feature	84
Supporting Features	85
Database Identity	85
Connection String Extensions	85
JADE Assembly Public Keys	86
SDS Exception (PAR 57818)	86
Secure Sockets Layer (SSL) Protocol	86
OpenSSL Version Upgrade (PAR 58040)	87
Secure Socket Layer Security (PAR 58011)	87

Silverlight Stateless (NFS 57307) .....	87
System::getDbDiskCacheStats Method .....	88
Thin Client Errors (PAR 58279) .....	90
Windows Ghosting (PAR 57699) .....	91
Changes in JADE Release 7.0.04 .....	92
Behavioral Change of Web Services (PAR 56921) .....	92
Certificate Authentication (PARs 56904, 57638) .....	92
Converting a User Database (PAR 56876) .....	93
Date Parsing Routines for Two-Digit Years (PAR 56832) .....	93
Deleting and Renaming Entities .....	93
Deleting Packages (NFS 45928) .....	93
Deleting a Schema (NFS 57540) .....	94
Renaming a Schema (NFS 57075) .....	94
Dock Controls (PAR 56859) .....	94
Initiating an Application (PAR 57541) .....	95
Iterating Backwards through a Virtual Collection (PAR 57322) .....	95
JADE Unit Testing Framework .....	95
Debugging the Execution of a JADE Unit Test (NFS 52101) .....	95
Initializing and Finalizing the JADE Unit Test Framework (PAR 56498) .....	95
JadeLocal Transport (PAR 57602) .....	96
JadeWebServiceProvder::getServerVariable (PAR 57163) .....	96
Java Framework Support for HugeStringArray Objects (NFS 54330) .....	96
Journal Rate Analysis Sampling (PAR 57125) .....	96
Monitoring Node Locks (NFS 54406) .....	97
Reblocking Collection Class Maps .....	97
Relational Population Service (RPS) .....	97
Adding an Existing Property or Method to an RPS Table (NFS 57061) .....	98
Converting OID Columns (PAR 57473) .....	98
Datapump Application Execution (PAR 53009) .....	98
Extracting an RPS Table (PAR 55803, PAR 56829) .....	98
Silverlight .....	98
Accessing Third-Party Controls (PAR 56871) .....	98
Extracting a Schema to Generate a XAP File (PAR 57458) .....	99
JADE Properties for XAML Controls inside DataTemplates (PAR 57187) .....	99
XAML Browser (PAR 57192) .....	100
Synchronized Database Service (SDS) (PAR 57217) .....	100
JadeDatabaseAdmin::getCurrentJournalOffset Method .....	100
Changed JadeDatabaseAdmin::sdsGetSecondaryProxy Method .....	100
[ConnectionParams] Section SocketBufferSize Parameter .....	101
System::verifyDbEncryptionMasterKey Method (NFS 57338) .....	101

# Upgrading to JADE Release 7.0.10

---

This document covers the following topics.

- [JADE Release Support](#)
  - [Deimplementations and Deprecations](#)
  - [Advanced Notice of JADE 7.1 Changes](#)
- [Accessing Details about Faults Fixed in Releases](#)
- [Upgrading from an Earlier JADE 7.0 Release or JADE 6.3.05 Release or Higher](#)
  - [Running the Upgrade](#)
  - [High-Availability Upgrade](#)
  - [JADE Thin Client Upgrade](#)
  - [Upgrading a Synchronized Database Environment \(SDE\)](#)
  - [Upgrading an RPS Node from JADE 6.3 to 7.0](#)
  - [Upgrade Validation](#)
  - [Hot Fix Releases](#)
- [Changes in JADE Release 7.0.10](#)
- [Changes in JADE Release 7.0.09](#)
- [Changes in JADE Release 7.0.08](#)
- [Changes in JADE Release 7.0.07](#)
- [Changes in JADE Release 7.0.06](#)
- [Changes in JADE Release 7.0.05](#)
- [Changes in JADE Release 7.0.04](#)

Refer to *Version 7.0.03 Release Information (RelInfo7003)* for details about JADE release 7.0.03 changes that may affect your JADE 6.3 existing schemas and changes in JADE release 7.0.03.

---

**Tip** For details about using Acrobat Reader to view JADE documents, see "JADE Product Information Library in Portable Document Format", in Chapter 2 of the *JADE Development Environment User's Guide*.

The *JADE Product Information Library* document ([JADE](#)) provides a summary of contents of documents in the JADE product information library and navigation to the documents.

---

If you want to develop your own installation process for Windows, the JADE install and upgrade steps are documented in the **ReadmeInstallSteps.doc** file in the **documentation** directory. To customize the deployment upgrade on Windows, see "Customizing the Deployment Upgrade Process", in Appendix A of the *JADE Runtime Application Guide*.

---

**Note** This release includes full updated JADE documentation, including the search index catalog, as all documents in the JADE product information library have been revised and reissued for this release.

---



# JADE Release Support

Support for JADE 6.3.12 continues until February 2015.

For details about the JADE release policy, go to:

[http://www.jade.co.nz/downloads/jade/JADE\\_ReleasePolicy.pdf](http://www.jade.co.nz/downloads/jade/JADE_ReleasePolicy.pdf)

For details about the JADE release schedule, go to:

<http://www.jade.co.nz/jade/updates.htm#releasesched>

JADE 7.0 requires Windows NT 6.0 or later (that is, Vista or Windows Server 2008, or later).

JADE 7.0 is supported on Windows 8 and Windows Server 2012.

Microsoft SQL Server 2012 is supported as a Relational Population Service (RPS) relational target database.

From JADE 7.0.09, JADE is not supported on Windows XP.

## Deimplementations and Deprecations

This section contains the deimplementations and deprecations in JADE 7.0.

### ActiveX Exposure Deimplementation

As notified in JADE 6.3, JADE's ActiveX exposure feature is no longer available. In recent years, ActiveX technologies have been replaced by .NET. From JADE 6.3, you could generate exposures using these .NET technologies, providing a more modern, flexible, and easier to develop mechanism than that provided by ActiveX.

If you have not already done so, where required, we recommend that you re-write ActiveX exposures using the JADE .NET exposure. For details, see Chapter 17 of the *JADE Development Environment User's Guide*.

### Compact JADE Single User Mode

As notified in JADE 6.3, Compact JADE single user mode is no longer available in this product release; that is, support for JADE databases running on Windows Mobile devices is no longer available.

The Compact JADE thin client continues to be available on these devices in this release.

### Database Rebuild Files Deimplementation

The operation to rebuild files provided by the JADE Database utility **jdbutil** and **jdbutilb** programs is no longer available.

Use the reindex files operation to re-establish corrupted indexes.

### JADE Dump and Load Utility Deimplementation

As notified in JADE 6.3.08, JADE's Dump and Load utility (**jddlutl**) is no longer available.

To convert a user database, use the **JadeConvertDb** application in the **jadclient** program command line. For details, see Chapter 4 of the *JADE Runtime Application Guide*.

### JADE on Linux

As notified in JADE 6.3, JADE 7.0 supports Windows only, which means that in this product release, Linux distributions (for both Red Hat and SUSE) are no longer available.

JADE will continue to support customers running JADE 6.3 on Linux until April 2014.

## JADE Portable Graphical User Interface (GUI) Client

As notified in JADE 6.3, JADE portable GUI client is no longer available with this release.

## Rational Rose Deimplementation

As notified in JADE 6.3, the JADE interface for the Rational Rose modeling tool is no longer available.

If you want to use third-party design tools (for example, Enterprise Architect), you can use the JADE XMI interface to import the model into your database. (For details, see Chapter 5, "XML Metadata Interchange (XMI) Support", of the *JADE External Interface Developer's Reference*.)

## System::getDbObjectCacheStats Method

In JADE 7.0, the database engine no longer utilizes an object cache. The **System** class **getDbObjectCacheStats** method is now deimplemented in JADE 7.0.

## Web Service Framework Jade61 Application Type

As notified in JADE 6.3, the **Jade61** value for the **JadeHttp.ini** file **ApplicationType** parameter is no longer valid.

## Advanced Notice of JADE 7.1 Changes

This section contains advanced notice of changes in JADE 7.1.

### 32-bit Presentation Client Upgrade

Any system that is to run JADE 7.1 must have Microsoft Visual C++ Redistributable Package installed. JADE 7.1 32-bit presentation clients require Visual C++ 2013 Redistributable Package (x86), version 12.0.30501. In JADE 7.1, by default a 32-bit presentation client will be upgraded to a version requiring the Microsoft Windows Visual Studio 2013 C++ runtime binaries.

---

**Note** JADE 7.1 does not support 32-bit presentation clients using the Visual Studio 2005 C++ runtime binaries that were required by JADE 6.3 releases.

---

### Java Framework Deprecation

The JADE Java framework is to be deprecated in JADE 7.1 and the Java exposure feature will no longer be available.

### Server Read-Only User Deprecation

The **ReadOnlyUser** value of the **server** parameter in the **jadclient** executable is to be deprecated in JADE 7.1, as there is no longer a requirement to run JADE from read-only media.

### Silverlight XAML Deprecation

The JADE Silverlight Extensible Application Markup Language (XAML) implementation is to be deprecated in JADE 7.1 and the XAML classes will no longer be available.

### String Arrays and Dictionary Keys

As the 30-character limit for schema entity names is increasing in JADE 7.1 to the maximum of 100 characters, the **Array** class will provide the **JadIdentifierArray** subclass in JADE 7.1, to enable you to store entities instead of in the **StringArray** subclass, which will no longer be sufficient for storing entities. (The membership string size for a **StringArray** is 62 and that of the new **JadIdentifierArray** is 100.)

If you are using entity names as dictionary keys, you must also consider the size of your dictionary key in JADE 7.1, as the total size of the key cannot exceed 512 key units. (A key unit is a byte for any non-character data type, or one character for any character data type; that is, key sizes are string-encoding agnostic. Key sizes also must allow for a null character to terminate any strings; characters are not null-terminated.)

## Accessing Details about Faults Fixed in Releases

To access the complete documentation about the Product Anomaly Reports (PARs) fixed in this release, run **Parsys**, our Fault Management and Customer Contact system. This system also enables you to view the progress of your own contacts.

If you have any queries about **Parsys**, please direct them to JADE Parsys Support in the first instance, at [parsysupport@jadeworld.com](mailto:parsysupport@jadeworld.com).

You can download the install shield for **Parsys** from the following URL.

[http://www.jade.co.nz/jade/parsys\\_support.htm](http://www.jade.co.nz/jade/parsys_support.htm)

When you first run the **Parsys** application, it downloads an update via the automatic thin client download feature. When this has completed and you have the log-on form ready and waiting, please contact JADE Parsys Support, who will then send you an e-mail message with your user code and password details. **Parsys** requires you to change your password when you first log on.

---

**Note** Because the encryption of passwords is a one-way algorithm, we cannot advise you of your password should you forget it, but we can reset it to a known value again.

---

## How to Locate PARs Fixed in a Specific Release

This section describes the actions that enable you to locate Product Anomaly Reports (PARs) fixed in a specific release.

### » To locate the PARs fixed in a specific release

1. Select the **Advanced Search** command from the Search menu with the following settings on the **Basic Search Criteria** sheet.
  - a. The **Latest** item is selected in the **Mode** combo box.
  - b. **All** is selected in the **Priority** list box.
  - c. The **PAR** check box is checked in the Phase group box.
  - d. The **Fault** check box is checked in the Type group box.
  - e. The **Closed** and **Patched** check boxes are checked in the Status group box.

---

**Note** If you want to restrict the search to the hot fixes that were produced, check the **Hot fix created** check box on the **Advanced Search Criteria II (Optional)** sheet.

---

2. On the **Advanced Search Criteria III (Optional)** sheet:
  - In the **Closed** list box of the Releases group box, select the release whose fixed PARs you want to locate (for example, the **7.0.3** list item).
3. Click the **Search** button.

# Upgrading from an Earlier JADE 7.0 Release or JADE 6.3.05 Release or Higher

This section covers the following topics.

- [Running the Upgrade](#)
  - [Running Two Releases of JADE on the Same Workstation](#)
- [High-Availability Upgrade](#)
- [JADE Thin Client Upgrade](#)
  - [Upgrading a 32-Bit Presentation Client Connecting to a 64-Bit Application Server](#)
- [Upgrading a Synchronized Database Environment \(SDE\)](#)
- [Upgrading an RPS Node from JADE 6.3 to 7.0](#)
- [Upgrade Validation](#)
- [Hot Fix Releases](#)

---

**Caution** Before you upgrade to this JADE 7.0 release, refer to *Version 7.0.03 Release Information (RelInfo7003)* for details about JADE release 7.0.03 changes that may affect your JADE 6.3 existing schemas.

---

## Running the Upgrade

Server nodes (database upgrades) require the 64-bit edition. Client nodes can upgrade the 64-bit edition or the 32-bit edition.

The JADE 6.3 to 7.0 database upgrade differs from previous JADE upgrades in that it involves a migration of the data, due to the nature of the structural changes of the objects in the database.

---

**Notes** This database upgrade will require sufficient disk space to accommodate an additional full copy of the database plus head-room of about 10 percent of the database size.

If you want to continue processing update transactions on a production JADE 6.3 database while a copy of that database is upgraded to the JADE 7.0 structure, see "[High-Availability Upgrade](#)", later in this document.

The JADE 6.3 to 7.0 upgrade process cannot include an ANSI to Unicode conversion (or the reverse). Such a conversion can be made before the upgrade or after it.

---

If you want to develop your own installation process, refer to the JADE install and upgrade steps documented in the **ReadmeInstallSteps** document file in the **\documentation** directory.

The JADE Setup program enables you to upgrade your binary and database files to JADE 7.0, by performing the following actions.

1. Ensure that your JADE environment is an earlier JADE 7.0 release or release 6.3.05 or higher.

On the system to be upgraded, carry out the following certify operations. Proceed to the next certify operation only when any and all errors reported in the current operation are resolved.

- a. A physical certify using JADE Database utility (**jdbutil.exe** or **jdbutilb.exe**), to ensure that the system is structurally correct. (For details, see Chapter 1 of the *JADE Database Administration Guide*.)
- b. A meta logical certify, to ensure that the meta model is clean. (For details, see "Running a Non-GUI JADE Logical Certifier", in Chapter 5 of the *JADE Object Manager Guide*.)

- c. A logical certify, to ensure that the user data is referentially correct. (For details, see "Running the Diagnostic Tool", in Chapter 5 of the *JADE Object Manager Guide*.)

---

**Note** If you are unsure how to interpret the information output by the certify process, contact JADE Support ([jadesupport@jadeworld.com](mailto:jadesupport@jadeworld.com)) for advice.

---

2. If your database has partitioned database files, ensure that any offline partitions are brought online.
3. Take a full backup copy of your existing JADE directories, first ensuring that your database is not in recovery mode.

---

**Caution** As roll-forward recovery of the installation and upgrade process is not supported, it is important that you backup your database before starting the JADE Setup process to install JADE 7.0 and upgrade your existing data.

---

4. Ensure that you have the appropriate privileges or capabilities to install applications.

---

**Note** Installing any Microsoft redistributable package requires administration privileges.

---

If you are upgrading the 64-bit edition, the **vc\_red.msi** Microsoft C++ 2010 SP1 Redistributable Package (x64) is required. If you are upgrading the 32-bit edition of JADE, the **vc\_red.msi** Microsoft Windows C++ 2010 SP1 Redistributable Package (x86) is required. This will be installed during the JADE installation and is supplied on the JADE distribution media.

5. To start the JADE Setup program, invoke the **setup.exe** program from the **Jade70** release medium or execute the executable program downloaded from the JADE Web site.
6. If not already installed, the required Microsoft Windows C++ Redistributable Packages are installed.
7. In the Welcome folder, click the **Next>** button to continue the upgrade process.
8. Read the entire software license agreement in the Software License Agreement folder and then click the **Yes** button to continue the upgrade.
9. In the Installation Type folder, select the **Feature Upgrade** option button, to specify that you want to upgrade an existing JADE release. By default, the **Fresh Copy** option is selected.
10. In the Setup Type folder, select the type of installation that you are upgrading. By default, the **Development** option is selected for 64-bit installation or **JADE Client** for 32-bit installation.

---

**Note** The **Custom** type applies only to a **Fresh Copy** installation type, and is not relevant when upgrading.

The **SDS/RPS Database Server** option applies only to 64-bit **Feature Upgrade** installation type (it is not available for a JADE 6.3 to 7.0 upgrade).

---

11. In the Select Installation Folders folder, specify the locations of the JADE files that are to be upgraded. The upgrade process defaults to the most-recently used JADE files, and displays these values in the **Install Directory**, **Executable Directory**, and **Database Directory** text boxes.

The installation directory is most likely to be the root directory in which you installed JADE, unless you subsequently renamed the root directory or moved the files to another location.

If the locations are not as required, click the adjacent browse buttons (indicated by the ... ellipsis symbols) to display the common File Selection dialog that enables you to select the appropriate directories and files.

By default, the **jade.ini** file located in your specified database directory is used. If required, use the **JADE INI File** text box to specify a different valid fully qualified directory and name of the JADE initialization file; for example:

```
d:\mysys\jade\system\jade.ini
```

If Program Start folders are to be updated, specify the name of the folder in the **JADE Program Folder** text box. If you are unsure of the folder to be updated, click the adjacent browse button to display the common Folder selection dialog that enables you to select the folder.

The **Database Directory** text box enables you to explicitly specify the location in which the database (system) files are installed. When the destination folder is not **\Program Files**, the database destination defaults to **system** under the install folder (for example, if you specify **c:\jade70** in the **Install Directory** text box, the database directory defaults to **c:\jade70\system**).

If the installation directory is a subdirectory of the programmatically determined location of **\Program Files**, the **\Program Files** portion of the install directory is replaced with the programmatically discovered location for the common application data directory (for example, if you specify **c:\Program Files\Jade70** in the **Install Directory** text box, the default database location is **c:\ProgramData\Jade70\system**).

The process checks whether the specified database directory is a valid system and that it is the correct ANSI or Unicode type.

12. The Start Copying Files folder summarizing your upgrade options is displayed. If the selections displayed in the Start Copying Files folder are correct, click the **Next>** button.

Alternatively, click the **<Back** button to modify your selections.

13. The Question dialog is displayed, advising you to ensure that you have taken a full backup of that database before you proceed with the upgrade process.

When you are sure that you are upgrading the correct system (and that it has been backed up), click the **Yes** button to start the upgrade process.

14. A warning message box is then displayed, advising you that Dynamic Link Libraries (DLLs) may need to be recompiled.

Click the **OK** button, to recompile any required DLLs.

15. A warning message may be displayed if the upgrade validation process has not completed. If so, check the **jadeupgrade.log** file for information about what needs to be modified in your user schemas to pass the validation and enable application execution.

16. When the upgrade is complete, the JADE Setup program informs you that the JADE Setup was successfully completed and that you can now view the **ReadMe.txt** file. To view the **ReadMe.txt** file, ensure that the check box is checked (the default).

The **ReadMe.txt** file is then displayed in a text editor (for example, Notepad). The **ReadMe.txt** file is a read-only text file installed in your JADE root directory that you can print or delete, if required. This file contains a reference to other JADE-related documents.

17. Click the **Finish** button to end the JADE upgrade process.
18. Re-establish any control file paths set for database files and any required partition file attributes such as location, label, and frozen or offline states.

For a JADE 6.3 upgrade, after the upgrade has completed, the JADE 6.3 database files will be retained in the directory **database-directory\_63** (for example, **c:\jade\system\_63**). The *database-directory* (for example, **c:\jade\system**) will contain the upgraded JADE 7.0 database.

---

**Caution** As with any JADE release, you may need to recompile any external method Dynamic Link Libraries (DLLs) or external programs using the JADE Object Manager Application Programming Interfaces (APIs) with the new JADE `\Include` and `\Library` files before you attempt to run your upgraded JADE systems. (For details about the JADE Object Manager APIs, see Chapter 3 of the *JADE Object Manager Guide*.)

Some obsolete files are deleted from the JADE directories when upgrading from JADE 6.3. If you require these files for your JADE system, you must save them before you upgrade or restore them from the original JADE 6.3 release medium.

Documentation and example files are not part of the installation and must be downloaded and installed from the JADE Web site or the release medium separately, if required, into the **documentation** folder and **examples** folder, respectively, of your JADE installation directory.

---

## Running Two Releases of JADE on the Same Workstation

You can have two releases of JADE installed on the same workstation, if the files are in different directories. If ODBC is installed, only the last installation of the JADE ODBC driver is available from the ODBC Data Source Administrator.

If you install multiple copies of JADE 7.0 on a machine, an initial dialog during the installation asks if the installation should install a new instance or maintain or update the installed instance. You should select the default **Install a new instance of this application** option.

## High-Availability Upgrade

When converting a JADE 6.3 database to a JADE 7.0 release, the database conversion step must be performed in-place on a production database with the database down for the duration of the conversion process.

The high-availability upgrade feature enables you to reduce the downtime required to upgrade a JADE 6.3 database to run with a JADE 7.0 release, by continuing to process update transactions on a production 6.3 version database while a copy of that database is upgraded to the 7.0 structure.

The software is installed, directories set up, files copied, and your user schema and data files converted to the new JADE 7.0 structure in a copy of the production database while the source database remains available for online transaction processes.

The required downtime is reduced to the steps executed in an offline transition phase, which are as follows.

1. Shut down the production database.
2. Execute an upgrade replay process to apply committed transactions from JADE 6.3 journals.
3. Execute standard release upgrade steps.
4. Copy or move the converted database to the production directory structure.

For details, see the **HighAvailabilityUpgradeSteps.doc** file in the `\documentation` directory.

## JADE Thin Client Upgrade

Ensure that you have the appropriate Windows privileges or capabilities to install applications.

If JADE is installed in the `\Program Files` directory (or `\Program Files (86)` directory on a 64-bit machine with 32-bit JADE binaries):

- If User Account Control (UAC) is disabled, the thin client upgrade will fail because of lack of permissions for standard users.

For administrative users, the necessary privileges are automatically granted so the upgrade will succeed.

- If UAC is enabled, administrative users are prompted with an **Allow** or a **Cancel** choice but standard users



must know and supply the user name and password of a user with administrative privileges to enable the upgrade to succeed.

For more details, see Appendix B, "Upgrading Software on Presentation Clients", in the *JADE Thin Client Guide*.

## Upgrading a 32-Bit Presentation Client Connecting to a 64-Bit Application Server

When a 32-bit presentation client connects to a 64-bit application server, the application server upgrades the version of the presentation client but it does not change the 32-bit to 64-bit type of the presentation client, because:

- The presentation client does not check to see if the operating system on which it is running is 64-bit-capable (and it would have to inform the application server about this).
- Any support libraries needed by the presentation client (for example, ActiveX control and automation libraries) would also have to be downloaded or already installed in the presentation client.

By default, a 32-bit presentation client will be upgraded to a version requiring the Microsoft Windows Visual Studio 2010 C++ runtime binaries. If it is not possible for operational reasons to install the Visual Studio 2010 runtime binaries, you can configure the application server to use the presentation clients built with Visual Studio 2005 (that is, compatible with JADE 6.3 C++ runtime binaries). For details, see Appendix B, "Upgrading Software on Presentation Clients", in the *JADE Thin Client Guide*.

## Upgrading a Synchronized Database Environment (SDE)

A JADE release 6.3 to 7.0 upgrade of a Synchronized Database Service (SDS) node is not supported.

When the upgrade to 7.0 has completed successfully, re-clone the secondary databases from the upgraded primary database.

## Upgrading an RPS Node from JADE 6.3 to 7.0

A JADE release 6.3 to 7.0 upgrade of an RPS node is not supported. To retain the Relational Database Management System (RDBMS) target database, see "[High-Availability Upgrade](#)", earlier in this document.

Conversely, if you do *not* want to retain the RDBMS target database, when the upgrade to 7.0 has completed successfully, re-clone the RPS node from the upgraded primary database and recreate the RDBMS target database from the upgraded RPS node.

## Upgrade Validation

During the upgrade process, a validation script is run to check the integrity of the upgraded system. Any user schema entities that conflict with system schema entities are logged as errors in the **jommsgn.log** file.

All errors must be corrected and validation re-run before user applications can be executed on the updated system. If the system is in the un-validated state, a message box is displayed when you log on to the JADE development environment, asking if validation should be re-run.

## Hot Fix Releases

Hot fixes for JADE system files are released as binary files.

To apply the hot fix:

1. Shut down the system.
2. Copy the hot fix system files into the appropriate directory.
3. Start up the system.

---

**Caution** You must apply all of the files contained in the hot fix at the same time.

It is important to ensure that versions of JADE system files do not diverge from dependent binaries. Doing this ensures that dependent code files (JADE system files and libraries) are backed up and restored together. The default location of the JADE system files is the installation directory (that is, the **bin** directory).

When it is necessary to restore a database from backup and perform recovery, you must avoid reverting to earlier JADE system file and binary versions. When restoring the binaries directory, ensure that it is from the latest backup.

---

# Changes in JADE Release 7.0.10

This section contains details about changes in JADE release 7.0.10.

---

**Caution** If your earlier JADE 7.0 system used dynamic properties, you must perform a manual fix before using the upgraded system. For details, see "[Dynamic Property Cluster Class Map Change \(PAR 60706\)](#)".

---

For details about changes in JADE:

- Releases 7.0.09, 7.0.08, 7.0.07, 7.0.06, 7.0.05, and 7.0.04, see "[Changes in JADE Release 7.0.09](#)", "[Changes in JADE Release 7.0.08](#)", "[Changes in JADE Release 7.0.07](#)", "[Changes in JADE Release 7.0.06](#)", "[Changes in JADE Release 7.0.05](#)", and "[Changes in JADE Release 7.0.04](#)", respectively, later in this document.
- Release 7.0.03 (the first general release of JADE 7.0), see the *JADE Version 7.0.03 Release Information (RelInfo7003)*.

## .NET Changes

This section describes the .NET changes in this release.

### .NET Class Library (PAR 60740)

Under certain circumstances in earlier releases, JADE notifications did not behave as expected when `app.doWindowEvents` was indirectly called from a .NET process.

---

**Note** The following applies only to .NET applications that use JADE methods exposed via the C# Exposure wizard in the JADE development environment.

---

If a .NET application calls exposed JADE methods and the JADE method contains a `beginNotification` or `beginClassNotification` instruction, the notification will be delivered to the thread that performed the .NET "new JobContext()".

A `beginNotification` or `beginClassNotification` instruction in a JADE method is regarded as subscribing to a different notification than one done via `RegisterNotificationHandler` or `UnregisterClassNotificationHandler` in .NET, even if all of the parameters are equivalent. An event that is explicitly (or implicitly) caused will be delivered to all subscribers, even if that language that caused the notification did not subscribe to the notification.

When a JobContext is Disposed, all subscriptions are cancelled.

As the JADE code is running within .NET, the JADE process will not have an idle state, so the `Application` class `doWindowEvents` method must be called. When doing so, review all of the documented cautions in Volume 1 of the *JADE Encyclopaedia of Classes* regarding the use of this method.

### JADE .NET Developer's Reference

The *JADE .NET Developer's Reference* has been enhanced to include the following information. (Chapters 1, 2, and 3 from earlier releases have been renumbered [Chapter 5](#), [Chapter 6](#), and [Chapter 7](#), respectively.)

---

<a href="#">Chapter 1</a>	.NET and JADE requirements
<a href="#">Chapter 2</a>	JADE .NET object management
<a href="#">Chapter 3</a>	Sample client-server application
<a href="#">Chapter 4</a>	Introduction to JADE .NET tutorial
<a href="#">Appendix A</a>	Mapping JADE primitive types to CLR data types
<a href="#">Appendix B</a>	Developing applications to store, edit, and query spatial information directly through the JADE .NET API

---

## Adding User Classes at Run Time

Your user applications can add user classes at run time. As these user-defined classes are visible only in the JADE development environment, they are not considered as part of the JADE model, you cannot reference them in JADE methods, they are not extracted, reorganized, and so on.

You cannot add properties to user classes in the JADE development environment. You can only define runtime dynamic properties on user classes. Instances of any lifetime can be created (that is, persistent, transient, and shared-transient lifetimes).

The following table summarizes the classes and methods that enable you to maintain user classes.

Class	Method	Description
Schema	<a href="#">addUserCollectionSubclass</a>	Creates a user collection class as a subclass of the specified superclass in the receiving schema
Schema	<a href="#">addUserSubclass</a>	Creates a user class as a subclass of the specified superclass in the receiving schema
Schema	<a href="#">deleteUserSubclass</a>	Deletes a user class from the specified superclass in the receiving schema
JadeUserCollClass	<a href="#">addExternalKey</a>	Adds an external key definition to a user class at run time
JadeUserCollClass	<a href="#">addMemberKey</a>	Adds a member key definition to a user class at run time
JadeUserCollClass	<a href="#">clearKeys</a>	Clears existing key definitions
JadeUserCollClass	<a href="#">endKeys</a>	Indicates the end of a single or multiple key definition
JadeUserCollClass	<a href="#">setLength</a>	Sets or changes the element length for an array
JadeUserCollClass	<a href="#">setMembership</a>	Sets or changes the membership of a user class at run time

For details, see volumes 1 and 2 of the *JADE Encyclopaedia of Classes*. See also "[Adding User Classes at Run Time](#)", in [Chapter 21](#) of the *JADE Developer's Reference*.

## AutoComplete Functionality

This section describes the AutoComplete changes in this release.

### AutoComplete Selection from a List (PAR 61330)

When you used the TAB key to select an item from the displayed **AutoComplete** list box in earlier releases, the tab character was appended to your selected option when it was inserted in the editor pane.

When you select an item in the list by using the TAB key, the tab character is no longer appended to the selected entry when it is inserted into the editor pane.

### Pseudotype Variables (PAR 61647)

Pseudotype variables are not handled by the AutoComplete feature.

## Binary and String Length (NFS 4822)

The behavior of embedded **Binary** and **String** properties (that is, non-binary large object (blob) or string large object (slob) properties) and **Binary** key values has changed. This also applies to the **BinaryArray** class and **StringArray** class and their subclasses.

In earlier releases, when a value was stored in a **Binary** or **String** property, the value was appended with null (0) bytes up to the declared length of the property. For example, if a property was defined as **Binary** length 60 and the property was set with a 20-byte value, 40 null bytes were appended. Similarly, if an external key dictionary had a **Binary** key, null bytes were also appended, up to the declared length of the key. After a **Binary** value had been stored, it was not possible to determine how many bytes were actually written and how many null bytes were appended. After a **String** value had been stored, the value returned was the value stored up to, but not including, the first null character. This meant that if the value stored contained an embedded null character, a truncated value was returned.

In this release, the actual **Binary** or **String** value stored in a property is maintained and the actual **Binary** value used as a key is maintained. With this change, the behavior of embedded **Binary** and **String** properties is now consistent with blobs and slobs; that is, the value stored in a property is the value that is returned when the property is retrieved. For example, if you declare a **Binary** property with length of 30 but the value stored is of length 20, the **Binary** primitive type **length** method returns 20 (with similar handling for a **String** property).

---

**Note** Assigning a string containing embedded null characters to a **String** attribute *may* result in truncation of the string at the first null character when the value is used; for example, if an external key dictionary has a **String** key, the value used for collation is the value up to, but not including, the first null character.

---

## Class Persistent Instances Methods (PAR 61167)

The **Class** class now provides the **countPersistentInstances64** and **countPersistentInstancesLim64** methods to return the number of non-exclusive 64-bit persistent instances in the receiver class. These methods have the following signatures.

```
countPersistentInstances64(): Integer64;  
  
countPersistentInstancesLim64(limit: Integer64): Integer64;
```

## Extract Sort Exception (PAR 61403)

In earlier releases, if the input file passed to the **File** class **extractSort** method contained a null character, because of the way embedded **Binary** and **String** properties were handled, a 1415 internal exception was raised and a process dump was generated.

The sort code now recognizes the null character and raises a new exception 5320 (*Unable to process all input in the file*).

## JADE Initialization File Parameter Handling (PAR 61704)

In earlier releases, the **<default>** value for a JADE initialization file section and parameter pair was case-sensitive.

The value is now case-insensitive (for example, **<default>** and **<DEFAULT>** are both valid).

## JADE Logical Certifier (PAR 61172)

In earlier releases, the JADE Logical Certifier did not detect if a collection contained an object multiple times, where the keys for one entry were valid and the keys for the other entries were not valid; that is, it was still possible to locate the entry using the **includes** statement.

The JADE Logical Certifier now detects if an object is in a **MemberKeyDictionary** multiple times, where the keys for one entry were valid and the keys for the other entries were not valid. The existing user data error 3 has been renamed 3A (*Object was found via foreach relatedObj in coll but not via coll.includes(relatedObj)*) and a new user data error 3B (*Object was found in the collection multiple times with valid and invalid keys*) is provided. The repair is **rebuild coll**; that is, if the object has an invalid key path, it is your responsibility to repair the error yourself.

## OpenSSL Library Version Upgrade (PAR 61368)

The OpenSSL library has been upgraded to version 1.0.1h.

## Thin Client Automatic Download (PAR 61337)

When a thin client installation was in progress in earlier releases, another initiation of JADE using the same binaries could cause the installation process to fail. JADE uses a file lock when determining the download requirements, but once the thin client installation was initiated, the file lock was lost and the second client could also attempt to perform the installation.

This has now been changed so that the file lock is retained by the JADE thin client install process for standard Windows clients. (This feature is not available in Windows for mobile clients). This prevents multiple clients attempting to perform the same download process. If another JADE thin client is initiated while the first client is processing the download requirements and the application server indicates that a download is required, the file lock is will be detected. If the value of the **AskToDownload** parameter in the [**JadeThinClient**] section of the JADE initialization file is set to **true**, a message box will be displayed (using a temporary copy of **jaddinstd.exe** to do the message box display), informing the client that another client is performing the download and asking him or her to wait. If the value of the **AskToDownload** parameter is **false**, the information is output only to the client **jommsg.log** file. In both cases, the JADE executable terminates with exit code 14165 (*The thin client software is currently being downloaded, please wait and try again shortly. The user performing the download is:*).

---

**Note** You should not use the same JADE binaries to access different application servers at the same time. If the application servers have different download files, a conflict of interest will occur when two JADE thin clients are initiated at the same time and the installation process will most likely fail with files being in use.

---

# Changes in JADE Release 7.0.09

This section contains details about changes in JADE release 7.0.09.

---

**Caution** If your earlier JADE 7.0 system used dynamic properties, you must perform a manual fix before using the upgraded system. For details, see "[Dynamic Property Cluster Class Map Change \(PAR 60706\)](#)".

---

For details about changes in JADE:

- Releases 7.0.08, 7.0.07, 7.0.06, 7.0.05, and 7.0.04, see "[Changes in JADE Release 7.0.08](#)", "[Changes in JADE Release 7.0.07](#)", "[Changes in JADE Release 7.0.06](#)", "[Changes in JADE Release 7.0.05](#)", and "[Changes in JADE Release 7.0.04](#)", respectively, later in this document.
- Release 7.0.03 (the first general release of JADE 7.0), see the *JADE Version 7.0.03 Release Information (RelInfo7003)*.

## AutoComplete Feature (PAR 60436)

The JADE AutoComplete feature selects a matching entry out of the most-recently used list for the current context in the displayed list box. However, in earlier releases the selection was case-sensitive. As a result, as more characters were typed, the currently selected item could change unexpectedly when the case of latest character did not match.

The selection is now case-insensitive, with the exception of the first character (the case of first character determines the types of entries that are displayed).

## C# Exposure Wizard

This section describes the C# Exposure Wizard changes in this release.

### ExternalCollection Subclasses (PAR 60285)

Uses of the **ExternalArray**, **ExternalDictionary**, and **ExternalSet** subclasses of the **ExternalCollection** class are not supported by the JADE C# exposure, and are excluded from the generated C#. In earlier releases, uses of these classes were generated and the C# compile failed.

The C# Exposure Wizard now ignores uses of these classes in the list of features that can be exposed (for example, a property reference).

---

**Note** You must manually alter any existing C# exposures that include such features, to exclude the exposed feature.

---

## Disk Write-Cache Messages (PAR 60863)

In earlier releases, no warning was issued during database file or journal file opening if device disk write-caching was enabled and Windows drive cache flush was disabled.

As each new volume is encountered when opening a writable file, the database now logs the device write-cache settings. If the device reports that write-cache is enabled and the device write-cache type is not write-through, warning messages are logged if either the device and the device does not support flush-cache operations or you have configured the device as power protected (suppressing flush-cache operations). Under these circumstances or if the device settings cannot be determined, a power outage could result in irrecoverable data loss or corruption.

## Dynamic Property Cluster Class Map Change (PAR 60706)

If your upgraded system uses dynamic properties from an earlier 7.0 releases, call the following method to move the dynamic property cluster definition instances to the correct class map file *before* you use your system.

```
rootSchema._fixDynamicPropertyClusters();
```

## Generated JADE .NET Names (NFS 60673)

The JADE .NET Import Wizard now uses a different mechanism to ensure that the generated JADE names are unique, resulting in more-workable JADE names. The results of this change are as follows.

- Fewer names need to be qualified
- The added qualifiers will mostly be simple one-level low-value numeric values (for example, `_1`, `_2`, and so on)
- Multiple-level qualifiers (for example, `_2_1`) will not often be needed

---

**Note** This change is applied only to new .NET assembly imports. Existing imports are unaffected, and reloading an assembly uses the mechanism under which the JADE names were generated so that existing interfaces are not affected.

---

The style of the generated names is saved in the database and in the forms definition (`.ddb`) file when extracted.

## Inherited Control Event Method Display (PAR 60398)

When the **Show Inherited** command in the View menu of the Class Browser was checked in earlier releases, the list of control and form event methods did not include events defined for superclasses included in the schema view. This command is unchecked, by default.

The display of form and control event methods now includes inherited superclass event methods when the **Show Inherited** command is selected.

---

**Note** Only defined event methods in superclasses are shown; not the unassigned event methods that are shown for the currently selected control or form item.

---

## Interface Stub Methods (NFS 60359)

When stub methods are created for an interface implemented on a class, the included text description now includes the name of the interface for which they were created; that is:

```
this method stub has been automatically generated by Jade to satisfy interface  
implementation requirements for the interface-name interface
```

## JADE Initialization File

This section describes the JADE initialization file changes in this release.

### AcceptZeroEnvironmentUUID Parameter

The [JadeServer] section of the JADE initialization file can now contain the **AcceptZeroEnvironmentUUID** parameter, which is set to **false** by default, indicating that the environment identity specified in the **server** command line parameter of a client shortcut must match the identity of the database server, as shown in the following example.

```
server=TcpIp://localhost:6005/48cf13df-bf6d-df11-87e2-2e5925024153
```



For more details, see "Format of the Server URI String", in Chapter 3 in the *JADE Installation and Configuration Guide*.

Set the **AcceptZeroEnvironmentUUID** parameter to **true** if zeroes can be specified for the environment identity in the **server** command line parameter of a client shortcut, as shown in the following example.

```
server=TcpIp://localhost:6005/00000000-0000-0000-0000-000000000000
```

## BackupCompressedFileGrowthIncrement Parameter (PAR 60653)

The [PersistentDb] section of the JADE initialization file can now contain the **BackupCompressedFileGrowthIncrement** parameter, which has a default value of **1G**. This parameter specifies the size of the increment by which a backup file increases during a compressed backup.

If the value supplied is not a multiple of the value of the **BackupBlockSize** parameter, it is rounded up to the next highest multiple.

The minimum value is **64M** bytes. If the file to be backed up is smaller than the value of the **BackupCompressedFileGrowthIncrement** parameter, the backup file is pre-allocated to that size. The maximum value is **64G** bytes.

## DiskCacheWriteOnlyDbSegments Parameter (PAR 60707)

The [PersistentDb] section of the JADE initialization file can now contain the **DiskCacheWriteOnlyDbSegments** parameter, which has an Integer default value of **16**, unless limited by available physical memory. This parameter specifies the minimum and maximum number of segments that the disk cache is to use when the database mode is write-only; for example, during upgrades.

For the minimum and maximum values, refer to the **DiskCacheMinSegments** and **DiskCacheMaxSegments** parameters, respectively, in the [PersistentDb] section.

---

**Note** The value of this parameter can be constrained by the actual maximum number of segments calculated during initialization.

---

The parameter is read when the database server node is initialized; for example, when you restart the database server.

## JADE Server Section Parameter Values (PAR 60842)

The default, minimum, and maximum values of the [JadeServer] section parameters listed in the following table are:

Parameter	Default	Minimum	Maximum
MaxLongThreads	100	1	4095
MaxShortThreads	100	2	4095
MinLongThreads	15	1	4095
MinShortThreads	20	2	4095
TransportIdlePollInterval	120000	5000	Max_Integer (#7FFFFFFF)

## JADE Logical Certifier (PAR 60649, 60505, 60506)

The JADE Logical Certifier now:

- Writes output of meta data certification to **\_metacert.\*** files, except for the fix file, which continues to be output as **\_logcert.fix**. (Repair files are unaffected by this change.)
- Appends output to the **\_logcert.log** file and the **\_repair.log** file, rather than overwriting them.

- Uses the **JadeLog** directory instance as the default value of the **Log File Directory** text box on the Jade Logical Certifier dialog.

## JADE Painter

This section describes the JADE Painter changes in this release.

### Integer Precision (PAR 60470)

In earlier release, the JADE Painter allowed the entry of **Integer** values beyond the maximum 32-bit integer possible for properties of that type (for example, the **tabIndex** property), which subsequently raised an exception.

The JADE Painter now restricts the entry of unsigned **Integer** value properties to a maximum of nine digits.

### Menu Design (NFS 2495)

The JADE Painter Menu Design form now displays:

- A list of accelerators that are available for use for the current selected menu item.  
This list contains any alphabetic characters in the menu captions that are not used as an accelerator in any other peer menu item.
- Top-level menu item accelerators of any superclass forms are excluded from the list if the menu item is a top-level item. Subclass forms are not considered.
- The accelerators used for any menu items in the same submenu list are excluded if the menu item is a submenu.
- A message, in red, if the currently selected menu item has an accelerator that is duplicated in any peer menu item.

This is a warning only. Duplicated accelerators are permitted and handled. The message includes the caption of the menu item that duplicates use of the accelerator key.

### JADE Version Information Utility (PAR 60988)

The JADE Version Information utility (**jverinfo**) now reports the total **commitLimit** and available **commitLimit**, rather than the total virtual memory and available virtual memory.

These **commitLimit** values are the combination of physical memory and page file size.

### Java Exposure Wizard (PAR 59510)

The Java Exposure Wizard had the following issues when generating the exposure files in earlier releases.

- When the output directory was browsed for, the directory was relative to the database server file system. However, when the files were created, the logic assumed that the file system was accessible from the running node; for example, the application server.
- When the **package-info.java** file was written, the value of the **FileNode** class **usePresentationFileSystem** property was **true**. As a result, the generate process failed if the client was not running on the same file system as the application server and did not have a similar file system structure.

As a result, the Java exposure generation has been changed so that it is now relative to the node executing the logic; that is, the application server. All files and directories are written using the value of **false** for the **usePresentationFileSystem** property.

This change should not have any impact unless your site had application servers on different machines from the database server (in which case the generate process would probably have failed).

---

**Note** This differs from the C# exposure, where the files and directories are accessed using the **FileNode** class **usePresentationFileSystem** property value of **true**.

---

## Methods View Browser

This section describes the Methods View Browser changes in this release.

### JADE Scripts and Test Cases in the Methods View Browser (PAR 60442)

The customized Methods View Browser now enables you to run and debug **JadeScript** and **JadeTestCase** subclass methods. (In earlier releases, you could do this from the Class Browser only.)

### Methods View Browser Context Menu (PAR 60947)

The JADE Methods View Browser now displays the Methods View menu as the context menu, and this menu now includes the **Open** command.

## ODBC Driver Query Execution Timeout (NFS 60759)

Open Database Connectivity (ODBC) queries executed in the JADE ODBC driver can now be timed out, by using the **QueryTimeout** parameter in the [JadeOdbc] section of the JADE initialization file to specify the number of seconds that a query executes before timing out. The default value of zero (**0**) indicates that there is no timeout. This parameter is valid for the server node in which the ODBC Server application is executing (it sets the default for all queries from all connections) or for the standard (fat) client ODBC driver.

The query timeout can also be set by the ODBC tool submitting the query. This setting overrides the JADE initialization file setting for that statement only.

A new exception 8360 (*Query timeout expired*) is raised when a query timeout has been set for an ODBC query and that time has expired before the query execution completes.

## Replayable Reorganization (NFS 60428)

In the batch JADE Schema Load utility (**jadloadb**), the handling of the optional **replayableReorg** parameter has changed.

The **replayableReorg** parameter now defaults to the value of the **EnableArchivalRecovery** parameter in the [PersistentDb] section of the JADE initialization file, and is constrained by the setting of the **EnableArchivalRecovery** parameter for the database.

Set the value of the **replayableReorg** parameter to **false** if you do not want to perform a replayable reorganization of your JADE database when the schema is loaded.

If you set the parameter to **true** or you let it use the default value, the parameter is set to the value of the database **EnableArchivalRecovery** parameter.

## RPS Database Creation (NFS 53778)

You can now create an RPS database for a specified schema and RPS mapping programmatically, by calling the **JadeDatabaseAdmin** class **createRpsDatabase** method. This method has the following signature.

```
createRpsDatabase (backupDir:      String;
                  schema:         Schema;
                  rpsMapping:     RelationalView;
                  rpsStorageMode: Integer;
                  verifyFiles:    Boolean;
                  overwriteFiles: Boolean;
                  quiesce:        Boolean) updating;
```

Specify the location of the RPS database to be created in the **backupDir** parameter. This directory must exist and be writable.

Specify the schema and RPS mapping for which the RPS database is to be created in the **schema** and **rpsMapping** parameters, respectively.

You must specify the storage mode of the created RPS database in the **rpsStorageMode** parameter, using one of the **JadeDatabaseAdmin** class constants listed in the following table.

Class Constant	Integer Value	Description
RpsStorageMode_Full	0	Full database replica RPS data store mode
RpsStorageMode_MappedExtent	1	Mapped extent RPS data store mode
RpsStorageMode_WorkingSet	2	Working set RPS data store mode

For details about the RPS database storage modes, see "RPS Data Store", in Chapter 2 of the *JADE Synchronized Database Service (SDS) Administration Guide*.

The **verifyFiles**, **overwriteFiles**, and **quiesce** parameters have the same values and definition as those parameters in the existing **JadeDatabaseAdmin** class **backupAllDbFiles** method.

---

**Note** This method can be executed on the primary system only.

---

## Save Method Message Box (NFS 60438)

In earlier releases, when you changed a method source and then clicked on a different method, a message box was displayed, stating that the specified method had been changed and prompting you to click the **Yes** button if you wanted to save the change (or the **No** or **Cancel** button).

Because clicking the **Yes** button saves *and* compiles the method, the wording of the message box now prompts you to click the **Yes** button if you want to compile the method and save the changes, the **No** button to discard your changes, or the **Cancel** button to continue editing the method.

## StringUtf8Array Elements (PAR 60605)

The length of elements in a **StringUtf8** array must be less than 8,000 UTF8 characters.

## Changes in JADE Release 7.0.08

This section contains details about changes in JADE release 7.0.08.

For details about changes in JADE:

- Releases 7.0.07, 7.0.06, 7.0.05, and 7.0.04, see "[Changes in JADE Release 7.0.07](#)", "[Changes in JADE Release 7.0.06](#)", "[Changes in JADE Release 7.0.05](#)", and "[Changes in JADE Release 7.0.04](#)", respectively, later in this document.
- Release 7.0.03 (the first general release of JADE 7.0), see the *JADE Version 7.0.03 Release Information (RelInfo7003)*.

### .NET Requirements (PAR 60253)

To import a .NET external component, one of the following .NET frameworks must be installed.

- 3.5
- 4.0
- 4.5

The .NET framework 4.0 is installed by the JADE installer.

### App Object in .NET Applications (PAR 59732)

You can now specify the application in the .NET connection string (that is, **Application=XXX**). Any initialize or finalize methods defined for that application will be called.

### Application Browser (NFS 59947)

The Application Browser now provides the following functionality.

Double-click on the name of:

- The startup form or startup document for an application now opens a new hierarchy browser window for that class.

If you do not have development environment security access to view the class, the request is rejected.

- An initialize or finalize method for an application now opens a new method source window for that method, to enable you to edit the method.

If you do not have development security access to view the method, the request is rejected. If you do not have change access, any change is rejected.

As a result of these actions, you can change the default application only by double-clicking on one of the first three columns in the table on the Application Browser (that is, the **Default**, **Name**, or **Application Type** column).

### Assigning an Object to a Partition (PAR 59567)

The **Object** class now provides the **setPartitionID** method, which has the following signature.

```
setPartitionID(partID: Integer);
```

This method, which must be called within the creating transaction, specifies the absolute partition in which to locate the receiver.

The value of the **partID** parameter must be a value in the range **1** through the **Max\_Integer** minus **15**, the target partition must be present, not frozen, and it must be version-compatible with the source object.

The value that is set is applied only when the transaction commits; that is, only the last value that was set is used.

The existing **Object** class **setPartitionIndex** method, which must also be called within the creating transaction, specifies the creation window-relative partition index in which to locate the receiver.

Exception 3146 is raised if the specified partition id or partition index is out of range. Exception 3187 is raised if the object is not being created; that is, it is already in a committed state.

## AutoComplete Functionality

This section describes the AutoComplete functionality changes in this release.

### Hiding the Display of RootSchema Entities (NFS 60118)

The AutoComplete feature now enables you to dynamically hide the display of RootSchema entities from the displayed list of possible AutoComplete entities. This reduces the available options in the list box, making it simpler to identify the required entity when it is known to be defined in a user schema.

The editor pane now provides the CTRL+R shortcut key sequence for the **AutoCompleteToggleRootSchema** action in the table on the **Editor Key Bindings** sheet of the Preferences dialog, which toggles the display of RootSchema entities in the **AutoComplete** list box.

Every time the **AutoComplete** list box is initiated, RootSchema entities are included in the displayed list, if they are available. Each subsequent use of the CTRL+R shortcut key sequence toggles the exclusion or inclusion of RootSchema entities. This option does not apply to global constants.

---

**Notes** If there is no non-RootSchema entity left after the first use of the CTRL+R sequence, the **AutoComplete** list box will close.

The next time the **AutoComplete** list box is displayed, the list will contain RootSchema entities regardless of the previous use of CTRL+R; that is, the sequence takes effect for the currently displayed list only. This shortcut key sequence is ignored by the JADE editor unless the **AutoComplete** list box is visible. In the JADE development environment, the CTRL+R shortcut keys initiate the Run Application dialog, and continues to do so if the **AutoComplete** list box is not visible.

---

### Translatable Strings (NFS 59442, PAR 59673)

The JADE editor AutoComplete feature now includes translatable strings.

When you type a dollar sign (\$) in a method, a list of the translatable strings for the current schema and superschemas is now displayed. As you type more of the name, the list is trimmed to show only those strings that include the entered sequence.

---

**Notes** For AutoComplete to continue to show the list of entries, the first character typed after the \$ must be uppercase.

Entering a period (.) after a translatable string prompts you with a list of **String** primitive type method suggestions.

---

## Cleaning up Virtual Directories (NFS 60054)

In earlier releases under Windows Internet Information Server (IIS), read-only files placed in the **JadeHttp.dll** virtual directory (that matched the **PurgeDirectoryRule** parameter in the [application-name] sections of the **jadehttp.ini** file) were deleted after a fixed 12-hour period of residency.

From this release, the [application-name] sections of the **jadehttp.ini** file can contain the **PurgeFileAge** parameter, which specifies the length of time since a read-only file was last modified before it is purged from the defined virtual directory for that application. This parameter applies only if the **VirtualDirectory** parameter has a specified value, and only files that match the **PurgeDirectoryRule** parameter criteria are considered (that is, all non-read-only files or by default, standard files of type **.jpg**, **.png**, or **.gif** that are more than the specified age).

The **PurgeFileAge** parameter value is a time-unit numeric, optionally followed by a modifier **D** (days), **H** (hours), or **M** (minutes). The **D** multiplier multiplies by 86,400; the **H** multiplier multiplies by 3,600; the **M** multiplier multiplies by 60; and all other time-unit values are treated as seconds.

Specify zero (**0**) for the **PurgeFileAge** parameter if you want to turn off the purging of the physical directory.

The minimum time-unit value that you can specify is **30** seconds; the maximum value is **365D** (days). Values outside this range are forced to their respective limits.

The following is an example of the **PurgeFileAge** parameter.

```
[webAppName]
PurgeFileAge=7M
```

## Compiling Method Constants (PAR 60173)

In earlier releases, a C++ stack overflow crashed the node when attempting to compile a method that has a large expression; for example, when a method declared a Binary constant that is the result of concatenating more than 5,000 literal values.

JADE now limits the number of consecutive operand/operator pairs to approximately 4,000. Exceeding this limit results in the new compiler error 6441 (*The expression is too complex*) being raised. These expressions may have compiled successfully in earlier releases.

Avoid the compile error by changing the expression to concatenate a series of sub-expressions enclosed in parentheses, where each sub-expression has significantly fewer than 4,000 operands.

The following string expression can be changed to use sub-expressions, as is shown in the second example.

```
"a" & "b" & "c" & "d" & "e" & "f" & "g" & "h" & "i" & "j" & "k" & "l"

("a" & "b" & "c") & ("d" & "e" & "f") & ("g" & "h" & "i") & ("j" & "k" & "l")
```

## Compressing Database Files (PAR 59324)

The JADE Database utility backup compress and decompress operation uses temporary files (**\*.lo\$** and **\*.da\$**) that are renamed if the operation succeeds (to **\*.lo\_** and **\*.da\_**). These files can remain if the operation fails. If a failure occurs, you can safely delete these **\*.lo\$** and **\*.da\$** files.

## Database Control File Integrity Checking (PAR 49652)

JADE now 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** parameter 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** parameter 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.

## DisableRPSRedintegrateOnRestart Parameter Default Value

The default value of the **DisableRPSRedintegrateOnRestart** parameter in the [SyncDbService] section of the JADE initialization file, introduced in JADE release 7.0.06, has now changed to **true**, which is the default value of the **DisableNativeRedintegrateOnRestart** parameter.

This default value change is due to currently unresolved issues that could occur during RPS secondary restart recovery when an audit stream being replayed contained a transaction abort sequence.

## Disabling Database Encryption (PAR 59650)

When database encryption is disabled (by executing the **DisableDatabaseEncryption** action in the **jdbcrypt** command line), 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 now 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** parameter 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.

---

## drawFillStyle Property Background (PAR 60265)

From JADE release 7.0.06, the original **Window** class **drawFillStyle** property values in the range **1** (**DrawFillStyle\_Transparent**) through **7** (**DrawFillStyle\_DiagonalCross**) were drawn with a white background instead of being transparent. This was intentional, for consistency with the fill styles in the range **8** through **55** introduced in JADE release 7.0.06. These styles have a white background because Microsoft does not provide a way of making a user-defined brush transparent for the background area. For details, see "[Draw Fill Styles \(NFS 58716\)](#)", later in this document.

From this release, the **drawFillStyle** property values in the range **1** through **7** (the Microsoft standard fill styles) are now drawn with a transparent background and the new JADE-defined fill styles in the range **8** through **55** are now drawn with a white background.

## Enabling or Disabling the read Instruction (NFS 59963)

The [JadeInterpreter] section of the JADE initialization file can now contain the boolean **ReadEnabled** parameter, which disables or enables the display of the User Input dialog when a method containing the **read** instruction is executed.

The parameter is read when the node is next initialized.

The **ReadEnabled** parameter, when set to **true** (the default), specifies that the User Input dialog is displayed when the method containing a **read** instruction is executed so that the required value of the primitive type variable can be specified. When this parameter is set to **false**, the **read** instruction returns null and the User Input dialog is not displayed for user entry.

Valid **ReadEnabled** parameter values are **<default>**, **true**, and **false**.



## Getting System Statistics (PAR 60021)

In earlier releases, a number of statistic values were truncated when the value exceeded **Max\_Integer** 32-bit integer value (2,147,483,647). These statistics have been changed to use the **Integer64** data type. Note that the existing **System** class **getStatistics** method will now raise exception 1406 if any of the statistic values exceed **Max\_Integer** (this can happen if your JADE system has been up for a long time; that is, the actual number of operations exceeds **Max\_Integer**).

The **getStatistics64** method, which loads the values of all the specified parameters with the corresponding system statistics, has the following signature.

```
getStatistics64(committedTransactions: Integer64 output;
               abortedTransactions: Integer64 output;
               getObjects: Integer64 output;
               queuedLocks: Integer64 output;
               createObjects: Integer64 output;
               deleteObjects: Integer64 output;
               updateObjects: Integer64 output;
               lockObjects: Integer64 output;
               unlockObjects: Integer64 output;
               beginNotifications: Integer64 output;
               endNotifications: Integer64 output;
               deliveredNotifications: Integer64 output;
               serverMethodExecutions: Integer64 output);
```

---

**Note** This method is not available on a Compact JADE node, where it would result in a 1068 - *Feature not available* exception.

---

The parameters for the **getStatistics64** method are listed in the following table.

Parameter	Obtains the number of...
committedTransactions	Committed transactions
abortedTransactions	Aborted transactions
getObjects	<b>getObject</b> operations performed
queuedLocks	<b>queuedLock</b> operations performed
createObjects	<b>createObject</b> operations performed
deleteObjects	<b>deleteObject</b> operations performed
updateObjects	<b>updateObject</b> operations performed
lockObjects	<b>lockObject</b> operations performed
unlockObjects	<b>unlockObject</b> operations performed
beginNotifications	<b>beginNotification</b> operations performed
endNotifications	<b>endNotification</b> operations performed
deliveredNotifications	Notifications that were sent
serverMethodExecutions	Methods executed in the server node operations

---

## JADE Command Files Creation (NFS 60055)

The **Schema** sheet of the Preferences dialog now contains the **Unset Schema Extract option 'Create Command File'** check box, which is unchecked by default.

This check box controls the setting of the **Create Command File** check box on the **Schema Options** sheet of the Extract dialog, which determines whether a JADE Command File (.jcf file) is created during the extract process. (In earlier releases, the **Create Command File** check box was checked by default.)

The **Unset Schema Extract option 'Create Command File'** check box controls whether the **Create Command File** check box is initially set or unset by default. The default value is **false**; that is, the **Create Command File** check box on the **Schema Options** sheet of the Extract dialog is checked and .jcf files are created.

## JADE Painter Menu Design (NFS 48839)

When the **formsManagement** style of the schema selected in the JADE Painter Menu Design uses the **formsManagement** style **FormsMngmt\_Single\_Multi (2)**, menu item captions that are a translatable string \$ name now display the translatable string text for the current base locale in the preview menu item. (This functionality already occurs for control captions displayed in the JADE Painter.)

## JadeAuditAccess::loadDescriptionByName Method (PAR 55770)

The **JadeAuditAccess** class now provides the **loadDescriptionByName** method, which has the following signature.

```
loadDescriptionByName(filename: String): Boolean updating;
```

This method attempts to load the description file specified in the **filename** parameter, rather than displaying the common File Open dialog. The value specified in the **filename** parameter is appended to the value of the **descriptionPath** property, to access the description file.

This method returns **true** if the specified description file loads correctly; otherwise it returns **false**. In addition, a two-digit or a four-digit year value in the file name of a description file is handled correctly.

This method ignores the value of the **autoDescription** property.

The **descriptionTS** property value is assigned only after a successful description load.

## Merge Iterator Functionality in the .NET JADE API (NFS 60352)

The merge iterator functionality in the .NET JADE Application Programming Interface (API) has been promoted from the **MemberKeyDictionary** collection class to the **JobDictionary** class, so that the **ExtKeyDictionary** collection class now also supports merge iterator functionality.

In addition, the **IMemberKeyDictionary** interface of the .NET framework has been renamed **IJobDictionary**.

## Moving Object Instances in an Encrypted Database (PAR 59648)

The batch JADE Schema Load utility (**jadloadb**) **commandFile** parameter now enables you to use the **Remap Class schema-name::class-name dbfile-name** and **MovelInstances [Workers number]** commands to move class instances between map files in an encrypted database.

In addition, you can now move **JadeBytes** shared-file and single-file instances between map files.

## Parameters Specified in the Command Line (PAR 59207)

The **NoReorgRecovery** parameter in the [JadeReorg] section of the JADE initialization file is no longer used. However, you can optionally specify the **noReorgRecovery=[true|false]** parameter in the command line of the JADE non-GUI client (**jadclient**) or batch Schema Load utility (**jadloadb**) program. (The default value of this optional command line parameter is **false**.)

The meaning and usages of the **noReorgRecovery** command line parameter are those of the obsolete **NoReorgRecovery** JADE initialization file parameter.

The following is an example of a JADE non-GUI client command line.

```
jadclient.exe path=C:\Jade\system ini=c:\Jade\system\jade.ini server=multiUser
app=JadeReorgApp schema=RootSchema endJade action=initiateReorgAllSchemas
waitForReorg=true initiateTransition=false replayableReorg=true
reorgAllowUpdates=false noReorgRecovery=true fastBuildBTreeCollections=false
```

You can optionally specify the **fastBuildBTreeCollections=[true|false]** parameter in the command line of the **jadclient** or **jadloadb** program. If you do not specify the **fastBuildBTreeCollections** parameter in the command line, the value of the **FastBuildBTreeCollections** parameter in the [JadeReorg] section of the JADE initialization file is used.

## Process Usages

This section describes the process usages display changes in this release.

### Including Subclasses in the Process Usages Display (PAR 60146)

The Classes In Use Browser now contains the **Include Sub Classes** check box, which enables you to specify that all subclasses of the selected class are also checked to see if they are also in use and include them in the display if they are. This check box is displayed only when the Classes In Use Browser is accessed from the **Show Process Usages** command from the Classes menu.

By default, the check box is checked only when the selected class is a subclass of the **Window** class. It is unchecked by default for all other classes.

### Multiple Processes Running under the Same User Code (PAR 60290)

When you display the current process or processes using a specific class in a non-production mode database by selecting the **Show Process Usages** command from the Classes menu and you have multiple JADE processes running under the same user code, each additional process that uses the same user code is now separately identified by a numeric qualifier. For example, the first process found for Wilbur is identified as **Wilbur**, the second is named **Wilbur(2)**, and so on.

## Removing a Class from a Schema View (NFS 59950)

The Class Browser of a schema view now provides the **Remove from Schema View** command from the Classes menu, to enable you to remove the selected class from the schema view. (This command is visible only when the Class Browser is open for a schema view.)

---

**Note** You can remove a class from a schema view only when the class that is selected is a leaf; that is, it has no displayed subclasses in the schema view.

---

### » To remove a class from a schema view

1. In the Class Browser, select the class that you want to remove from the schema view.
2. Select the **Remove from Schema View** command from the Classes menu.

A message box is then displayed, to enable you to confirm that the specified class is removed from the specified schema view.

3. Click the **Yes** button to confirm that the selected class is to be removed from the schema view. (The class is *not* deleted; it is removed only from the schema view.) Alternatively, click the **No** button to abandon the deletion.

The Class Browser of the schema view is then updated to reflect the removal of the selected class.

## Silverlight Painter (PAR 54312)

A Silverlight popup-type control cannot be directly displayed in the JADE XAML Painter because it does not have a development-mode operation.

JADE now substitutes a **XamlBorder** control for the popup window when displaying the control, provided that the **XamlPopup** window is the direct content of a top-level **<UserControl>** element. If it is not, neither the popup window nor its content can be displayed or selected. When a **XamlPopup** control can be displayed using a **XamlBorder** control substitute, the painter operation is seamless, except that only changes to the **width** and **height** properties for the popup window are reflected in the painter display. Other property changes to that control can be viewed only at run time.

## Support for HTML-based Online Help

JADE now supports HTML Web-based online help in your JADE applications, using the **Application** class **helpFile** property and the **Window** class **helpKeyword** property. When help is invoked directly via the **Window** class **showHelp** method or via the user pressing the help key (F1), a URL is created and the default browser is invoked to display the URL.

- **Application::helpFile** property

The help file can be an Adobe Acrobat Portable Document Format (**.pdf**) file, a Windows help (**.hlp**) file, compiled help (**.chm**) file, or Hypertext Markup Language (**.htm** or **.html**) files. For Web-based HTML help, the value of the **helpFile** property is set to a base URL, as shown in the following code fragment examples.

```
app.helpFile := "http://www.example.com/prodhelp";
app.helpFile := "http://www.example.com/prodhelp/" & app.version.String & "/";
```

- **Window::helpKeyword** property

When help is requested, if the help file specifies Web HTML help, detected by the value of the **helpFile** property starting with a recognized URL scheme (that is, **http://** or **https://**), JADE attempts to construct a URL to pass to the default Web browser.

If the value of the **helpKeyword** property starts with a recognized URL scheme, the **helpKeyword** URL is used; otherwise the value of the **helpKeyword** property is appended to the value of the **helpFile** property to become the URL to use, as shown in the following code fragment example.

```
mybtn.helpKeyword := "Form1/button1.html";
mybtn.showHelp;
```

- [JadeHelp] section of the JADE initialization file

The [JadeHelp] section of the JADE initialization file can now contain the following parameters.

- **HelpSchemes**, which has the default value of **http://,https://**.

This parameter enables you to specify other recognized URL schemes; for example, adding **file://** (that is, **http://,https://,file://**) enables you to load HTML help pages from your local disk.

- **HtmlHelpIndexUrl**, which has the default value of null, or blank.

If this parameter is present, it must be a complete URL. If present and the help index, help context, or help finder has been requested, this value will be used.

```
HtmlHelpContentsUrl = http://www.example.com/prodhelp/contents.htm
```

- **HtmlHelpContentsUrl**, which has the default value of null, or blank.

If this parameter is present, it must be a complete URL. If present, help has been requested, and no **helpKeyword** property value can be found, this value will be used.

## Thin Client Upgrade (NFS 59839)

When the JADE thin client downloads any files with an **.ocx** name type, an attempt is made to register these files to the Windows operating system. However, from Windows Vista and later, this registration process can be performed only under administration mode.

The JADE thin client installer now detects that an OCX file has to be installed and attempts to re-run itself in administration mode so that the OCX file can be registered with the Windows operating system. If the installer can be run in administration mode, the files downloaded are installed as normal and any OCX files are registered.

If the client environment does not permit the operation of administration mode, the attempted re-initiation fails and the download installation continues in non-administration mode. This results in the OCX registration process failing and being reported to the user. This is a non-fatal error, and use of the OCX will require manual intervention from an administrative user.

---

**Note** The other condition under which an attempt is made to run the JADE thin client installer in administration mode still remains; that is, when the download files are to be installed under the **Program Files** directory.

---

## TimeStampInterval::set Method (PAR 58981)

If the value of the **milliseconds** parameter is greater than one day in milliseconds (that is, **86400000**), the **TimeStampInterval** value that is set is now incremented by the number of whole days that the **milliseconds** parameter value represents.

## Transient Leak Analysis

This section describes the transient leak analysis changes in this release.

### Displaying Transient Leaks in a Methods Browser (NFS 59913)

When locating possible transient leaks (by using the **Find Possible Transient Leaks** command in the Schema menu), the Find Possible Transient Create Leaks dialog now enables you to specify the way in which possible transient leaks are reported; that is, to:

- Display in a Methods Browser (the default) the methods in which there are possible transient leaks, so that you can edit the method source whose transient leak list entry you have selected.

When you press TAB to move focus from the table of methods at the top of the browser to the editor pane, the cursor is positioned at the left of the source line in which the possible transient leak has been identified.

- Preview the print output.

### Excluding Code from Transient Leak Analysis (NFS 59911)

The transient leak analysis can report potential leaks that are not in fact leaks, for reasons that are outside of the understanding of the analysis.

You can now mark such entries for exclusion from the report that is produced, by placing the following tag in a comment on the same line.

```
[ExcludeFromTransientLeakReport]
```

The following code fragments are examples of marking lines of code for exclusion from the transient leak analysis.

```
create cust transient; // create the customer [ExcludeFromTransientLeakReport]
cust := methodReturnsTransient(); // [ExcludeFromTransientLeakReport]
```

To help you locate the required string, JADE now provides a **PossibleTransientLeaks** global constant category that contains the **ExcludeFromTransientLeakReport** global constant (which has a string value of "Exclude this from the Transient Leak Report").

The Reporting Options group box on the Find Possible Transient Create Leaks dialog now provides the **Ignore [ExcludeFromTransientLeakReport] tags** check box, which is unchecked by default. Check this if you want method lines containing this tag in a comment to be analyzed for possible transient leaks instead of being ignored.

## Locating Possible Transient Leaks (NFS 59910)

When locating possible transient leaks (by using the **Find Possible Transient Leaks** command in the Schema menu), situations are now detected where a **return** instruction is executed before the deletion of a created transient and the object is not deleted in a epilog. For example:

```
create cust transient;
...
if somecondition() then
    return;
endif;
...
delete cust;
```

## Unit Tests

This section describes the unit test changes in this release.

### Failure Count (PAR 59549)

When unit tests were running in earlier releases, if a **unitTestBefore**, **unitTestBeforeClass**, **unitTestAfter**, or **unitTestAfterClass** method failed, the **Failed** statistic incremented. When the unit tests finished running, the **Failed** statistic was reset to zero (0).

From this release, the final failure count now reflects the following.

- If a class instance create fails, the first class method executed will be counted as failed.
- If a **unitTestBeforeClass** method fails, the first class method executed will be counted as failed.
- If a **unitTestAfterClass** method fails, the last class method executed will be counted as failed.
- Any failure associated with a method will count as a failure only in the final failure statistics, whether it is a failure resulting from any of the previous items in this list or from a failure in the **unitTestBefore**, **unitTestAfter**, or **unitTest** method.

---

**Note** The total failures displayed while the tests are running shows the actual number of methods that failed. This figure is replaced on completion of the test run by the combined figure of each of the list items in the previous list.

---

### Unit Test Method Options in an Abstract Class (PAR 59914)

JADE now raises exception 6442 (*An abstract class method cannot have a unitTest option*) when you attempt to compile a unit test method on an abstract subclass of **JadeTestCase** and the method has a unit test option (for example, the **unitTestBeforeClass** or the **unitTest** method option).

In addition, changing a subclass of **JadeTestCase** to **abstract** causes all of the unit test methods defined in that class to be marked in error.

## Visibility of Loaded Schemas (PAR 58840)

When you added (loaded) a schema from outside of the JADE development environment in earlier releases (for example, programmatically via a method in the same JADE system or by using the **jadloadb** batch Schema Load utility), the schema was not displayed in the Schema Browser until you restarted the JADE system or you specified the schema in the Find Schema dialog.

The Schema Browser is now updated to display the new schema, regardless of the way in which it was added to the system.

## Changes in JADE Release 7.0.07

This section contains details about changes in JADE release 7.0.07.

For details about changes in JADE:

- Releases 7.0.06, 7.0.05, and 7.0.04, see "[Changes in JADE Release 7.0.06](#)", "[Changes in JADE Release 7.0.05](#)", and "[Changes in JADE Release 7.0.04](#)", respectively, later in this document.
- Release 7.0.03 (the first general release of JADE 7.0), see the *JADE Version 7.0.03 Release Information (RelInfo7003)*.

## About Form of an Application (PAR 57199)

When you use the JADE Painter menu designer to create a standard Help menu that contains an **About** menu item, JADE displays a message box at run time if you do not set the **Application** class **aboutForm** property.

This message box has been modified so that it now displays the following information.

```
----- Title:
"About Application:  application-name"

----- Contents:
Application:  application-name

Release:  application-release-property

Jade Version:  nn.nn.nn
```

This information is displayed on the JADE Web Application monitor About form, for example.

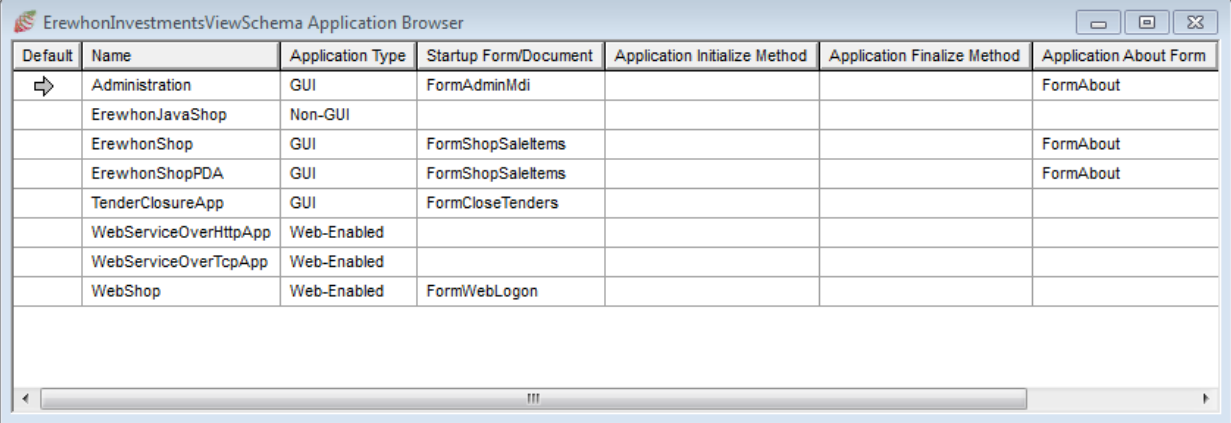
## Application Browser (NFS 44762)

The Application Browser now provides a summary of application attributes. These attributes are displayed in a table that has the following columns.

1. **Default**, with an arrow in the row of the default application.
2. **Name**, containing the application name.
3. **Application Type**, containing the application type.
4. **Startup Form/Document**, containing the name of the startup form or the startup Silverlight document, if specified.
5. **Application Initialize Method**, containing the name of the application **initialize** method, if defined.
6. **Application Finalize Method**, containing the name of the application **finalize** method, if defined.
7. **Application About Form**, containing the About form of the application, if specified.



The following diagram is an example of the Application Browser in this release.



Default	Name	Application Type	Startup Form/Document	Application Initialize Method	Application Finalize Method	Application About Form
⇒	Administration	GUI	FormAdminMdi			FormAbout
	ErewhonJavaShop	Non-GUI				
	ErewhonShop	GUI	FormShopSaleItems			FormAbout
	ErewhonShopPDA	GUI	FormShopSaleItems			FormAbout
	TenderClosureApp	GUI	FormCloseTenders			
	WebServiceOverHttpApp	Web-Enabled				
	WebServiceOverTcpApp	Web-Enabled				
	WebShop	Web-Enabled	FormWebLogon			

**Tip** You can right-click in the table to display the Application menu commands.

## Binary Length (PAR 59558)

Note that the upgrade to JADE 7.0 does *not* modify embedded **Binary** properties (that is, non-binary large object, or blob, properties), **Binary** key values, and the **BinaryArray** class and its subclasses created under JADE 6.3.

Because of this, the actual **Binary** value stored in JADE 6.3 and converted to JADE 7.0 may not compare as equal to the same **Binary** or **BinaryArray** value stored under JADE 7.0. For details, see "Binary Length (NFS 4822)" under "JADE 7.0 Changes that May Affect Your Existing Systems", in the *Version 7.0.03 Release Information (RelInfo7003)*.

## Date and Time Logged Formats

The date and time logged to the **jommsg.log** file can now be output in Coordinated Universal Time (UTC) or local time, by using the **UtcDateTime** parameter in the [JadeLog] section of the JADE initialization file. This parameter has the following format.

```
[JadeLog]
UtcDateTime=true|false
```

If the **UtcDateTime** parameter is not specified or it is set to **<default>**, the default value of **false** indicates that log entries are local time.

When a JADE executable starts up (for example, **jade.exe** or **jdbutilb**), a new current **UTC Time** and **Local Time** line are logged with the **Invocation:** entry. UTC dates have dash (-) separators; local dates have slash (/) separators.

The following is an example of the **jommsg.log** file output when the **UtcDateTime** parameter is set to **true**.

```
2013-03-06 22:15:11.202 020ac-302c WILBUR1A: (jade) Version 7.0.07 - Unicode,
Development only features: Enabled, OS Version: NT 6.01 build
7601 (Service Pack 1), Wow64: false, Architecture:
x64-msoft-win64-unicode
2013-03-06 22:15:11.203 020ac-302c WILBUR1A: Invocation:
"C:\Users\wilbur1\Developer\JadeDevel\jade70\Debug_Unicode\x64\bin\jade.exe"
ini=C:\Users\wilbur1\Developer\JadeDevel\jade70\Debug_Unicode\x64\etc\jade.ini
server=singleUser
2013-03-06 22:15:11.207 020ac-302c WILBUR1A: I18N: UTC Time 2013-03-06
22:15:11.204, Local Time 2013/03/07 11:15:11.204 +1300
2013-03-06 22:15:11.201 020ac-302c RPCMgr2:
RPCManager::createInstance: 346B640-1-1-NAG creates 36A1300-1-109-RPC
```

```
2013-03-06 22:15:11.218 020ac-302c RPCMgr2: TransportIdlePollInterval = 120000 ms
...
```

The following is an example of the **jommsg.log** file output when the **UtcDateTime** parameter is set to **false**.

```
2013/03/07 11:16:20.710 01b3c-2604 WILBUR1A: (jade) Version 7.0.07 - Unicode,
Development only features: Enabled, OS Version: NT 6.01 build
7601 (Service Pack 1), Wow64: false, Architecture:x64-msoft-win64-unicode
2013/03/07 11:16:20.711 01b3c-2604 WILBUR1A: Invocation:
"C:\Users\wilbur1\Developer\JadeDevel\jade70\Debug_Unicode\x64\bin\jade.exe"
ini=C:\Users\wilbur1\Developer\JadeDevel\jade70\Debug_Unicode\x64\etc\jade.ini
name=Debug_Unicode_Win64 server=singleUser
2013/03/07 11:16:20.711 01b3c-2604 WILBUR1A: I18N: UTC Time 2013-03-06
22:16:20.711, Local Time 2013/03/07 11:16:20.711 +1300
2013/03/07 11:16:20.709 01b3c-2604 RPCMgr2:
RPCManager::createInstance: 34BB640-1-1-NAG creates 3AF1300-1-109-RPC
2013/03/07 11:16:20.724 01b3c-2604 RPCMgr2: TransportIdlePollInterval = 120000 ms
...
```

## Date Format (PAR 58201)

In earlier releases, the **DateFormat** class **longMonthNames** and **shortMonthNames** properties contained 13 entries in their string arrays, even if the locale being used did not have a thirteenth month in the calendar. (The thirteenth entry was a null string, in that case.)

The string arrays of these properties now contain 12 entries, unless the locale defines a short or long name for the thirteenth month.

## Extracting all Exposures (PAR 58995)

When you use the **ExtractAllExposures** application in the **jadclient** non-GUI client to generate C# code for all classes in a schema, the **ExtractAllExposures** application exit code returns:

- 0, if successful
- 1, if the three parameters are not specified
- 2, if the specified schema does not exist
- 3, if the path name is invalid

## Function Keys (PAR 59439)

The CTRL+F shortcut keys now display the Find/Replace dialog. In earlier releases, this key combination displayed the External Functions Browser, which is now displayed by using SHIFT+CTRL+F.

In addition, if the result of the **Find Again** and **Find At Caret** commands in the Edit menu does not find the specified search text, a message box is now displayed, advising you of this.

## Interface Implementation (PAR 59129)

The caption of the first check box in the Interface Options group box on the **Miscellaneous** sheet of the Preferences dialog has been changed to **When implementing an interface, generate class methods with errors so they must be dealt with**. (The caption **When implementing an interface, do not compile automatically generated stub methods** in earlier releases did not reflect the action that occurred, as methods always compiled.)

## JADE Debugger (NFS 59455)

The JADE Debugger View menu now contains the **Restore Default Layout** command. Select this command if you want the Local Variables, Call Stack, and method source windows all visible, positioned, and sized according to the default JADE layout rules.

In the default layout mode, resizing the JADE Debugger window retains the relative default layouts of the three windows. The current left, top, width, and height values of the main JADE Debugger window are retained.

The **Restore Default Layout** command is disabled if the windows are already in the default layout positions and sizes.

## JADE Logical Certifier Detection of Abstract Class Instances (PAR 59203)

The JADE Logical Certifier utility can now output error 32 (*Invalid Instances in File*), which is detected when instances of an abstract class are found, or when instances of a class are found in the wrong database map file.

---

**Note** This fix is commented out in the repair file that is generated. Instances of abstract classes should not exist. Instances of non-abstract classes should exist only in the database map file in which the class is defined.

Any class instances that the JADE Logical Certifier utility reports in the wrong file cannot be accessed directly. You should investigate the nature of these instances before they are deleted, and consider whether these instances should be restored.

---

### Repair:

```
deleteInstances fileName className classNumber
```

## JADE Monitor

JADE Monitor changes in this release are documented in the following subsections.

### Additional Information in the Users View (NFS 21502)

The **Users** view of the JADE Monitor now includes columns of additional information; for example:

- Node IP address
- Offered connection details for the application server (port, and so on)
- IP address information for connected presentation clients

### JADE Monitor Default View (PAR 51595)

You can now specify the default view that is displayed when the JADE Monitor starts up; for example, the **Users** view.

Specify the valid view in the **DefaultPane** parameter in the [JadeMonitor] section of the JADE initialization file on the client node. You can specify one of the **Users|Locks|SystemStatistics|MonitorSetup|Summary** values. If you specify any other value, the **Summary** view is displayed.

## JADE Serial Port (PAR 58961)

The **JadeSerialPort** class now provides the **FlowControl\_None** constant, which has an Integer value of **2**.

You can now use serial ports with numbers greater than **9**. You must use the extended name format; for example, "\\.\COM11". (For details, see "Communications Resources" under the Microsoft Developer Network (MSDN) CreateFile function documentation.)

## JadeBytes Instances (PAR 58990)

In earlier releases, the **JadeBytes** class **allocate** method did not reject invalid size values; for example, the value of the **length** parameter was less than zero (**0**) or greater than the maximum instance size. (The maximum length of a **JadeBytes** instance is approximately 1,019G bytes.)

From this release, an invalid value is rejected and exception 1342 (*JadeBytes maximum content size exceeded*) is raised.

## Loading Forms Definition Files (PAR 56094)

The forms definition (**.ddb**) load process now displays a warning message if a control does not have a property reference on the form when handling form translations. The form becomes invalid, and a **jommsg.log** entry with the following format is output.

```
*****Warning: Control name on form name has no form control reference.
```

To avoid this warning, ensure that the schema (**.scm**) and forms definition (**.ddb**) files match.

## Local Implementors (PAR 53510)

The local implementors list for a method selected in the Class Browser now always includes the schema name, regardless of the schema in which the method is defined.

## MDI Window List Order (NFS 59101)

You can now control the order in which MDI windows are listed in the Window menu. The order is no longer affected by the MDI style setting of the Preferences dialog.

The **Browser** sheet of the Preferences dialog now contains the Mdi Window List Order group box, which contains the following option buttons,

- **Creation Order** (the default value), indicating that the MDI windows are listed in creation order of the open MDI child forms (that is, the oldest through to the most recent)
- **Last Use Order**, indicating that the order is in which the MDI child forms were last used (that is, the most recent use through to the oldest use)

## Printing

Printing changes in this release are documented in the following subsections.

### Opening a Printer (PAR 58242)

When a printer is about to be opened and the currently defined printer name is no longer valid, the following log message is now written and the current default printer is used instead.

```
Printer name is no longer valid: printer-name - selecting default printer
```

---

**Note** If the default printer is used, JADE does not retain the printer name between print jobs. If you change the default printer, the next print job to the default printer is output to a different (the new default) printer. If you have changed the default printer and you want to output the job to the earlier printer, specify the actual printer name.

---

### Printing Draw Fill Styles (PAR 59082)

Printing the draw fill styles in the range **DrawFillStyle\_4DotDiamond99** (8) through **DrawFillStyle\_Balls** (55) using the Windows Enhanced Meta Files (EMF) format produces a solid gray hue only, because of an EMF problem that is outside JADE's control.

To print these styles, you must use the Scalable Vector Graphics (SVG) format.

## Purging a Partition (PAR 59558)

You can now 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.

## Schema Inspector

Schema Inspector utility changes in this release are documented in the following subsections.

### Inspecting Collections (NFS 59009)

The Schema Collection Inspector has been enhanced, as follows.

- The Schema Collection Inspector window contains a list box that displays the properties of the collection and one that lists the entries in the collection. Clicking on a:
  - Property name displays the property value.
  - Collection entry displays the collection entry details.

- If the collection class has non-system properties, both list boxes are automatically made visible.

If the collection class does not have non-system properties, only the collection entries list box display is initially made visible.

- The Options menu now provides the **Show Collection Properties** command, which toggles the display of the collection properties list box and the check mark on the menu item.

This command is not visible if the collection is an internal JADE collection (for example, that used to display all instances of a class).

- When both the collection properties and collection entries list boxes are visible, a resize bar is displayed between them.

To change the size of each of the list boxes, drag this resize bar vertically.

### Inspecting Dynamic Objects (PAR 59423)

In earlier releases, the JADE Inspector displayed only the value of any dynamic property of a dynamic object and did not allow drilling down on any referenced objects.

If a dynamic property is a reference, double-clicking the property name now opens an inspector for that reference. In addition, dynamic property names are now displayed in dark blue text.

### Inspector Forms (NFS 59331)

The following changes have been made to JADE Inspector forms.

- The default initial height and width of the JADE Inspector utility forms has been increased,
- Any change to the font information is now stored in the **Font** parameter in the [JadeInspector] section of the JADE initialization file on the client. The parameters in this section are read whenever an Inspector form is displayed.

The stored information is in the form *font-name*, *font-size*, *font-bold*; for example:

```
[JadeInspector]
Font=Arial,9.75,true
```

If the information is missing or not in the expected format, the default font is used (for example, MS Sans Serif, 8 points, regular).

- Inspector forms now provide the following menu commands.
  - **Save Position/Size** command in the Options menu. Selecting this command saves the current position and size of the form in the **WindowPos** parameter in the [JadeInspector] section of the JADE initialization file on the client.

This information, which is used to position and size any subsequent Inspector forms that are displayed, is stored in the form *left, top, width, height*, for example:

```
[JadeInspector]
WindowPos=100, 150, 600, 700
```

The values are pixels, relative to the top left corner of the window.

If the information is missing or not in the expected format, the default position and size are used.

- **Clear Saved Position/Size** command in the Options menu. Selecting this command replaces the value of the **WindowPos** parameter in the [JadeInspector] section with **<default>**.

When the next Inspector form is opened, the default position and size will be used.

---

**Note** If more than one user uses the same JADE initialization file on the client, they will share the same values for parameters in the [JadeInspector] section.

---

## Search Functionality (PAR 54465)

When searching using SHIFT+CTRL+F3, the find again (F3) action used the last set of search criteria that was established. If multiple search result windows were open, the last search criteria that were actioned were used in each window.

From this release, when you press F3, each search results window now restores its own search criteria if it is not current so that F3 works as expected if you have multiple search result windows open.

## Web Service Consumer Generate Test Case Dialog (PAR 53623)

If you click the **OK** button in the Web Service Consumer Generate Test Case dialog and you have not selected any methods, *No methods were selected - Continue?* is displayed in a message box.

Click **No** if you want to cancel the process and return to the dialog, or click **Yes** to continue with the generate process.

The shortcut keys of the **Generate Parameter Defaults** check box are now ALT+P and those of the **Generate Expected Defaults** check box are now ALT+E.

Unchecking the **Select All Methods** check box no longer affects the list of selected methods.

## Relationship View Diagram Printing (PAR 61264)

In earlier releases, the relationship view of classes diagram always printed in landscape mode.

This diagram now prints in the orientation mode currently set for the printer to which it is output.

## Changes in JADE Release 7.0.06

This section contains details about changes in JADE release 7.0.06.

For details about changes in JADE:

- Releases 7.0.05 and 7.0.04, see "[Changes in JADE Release 7.0.05](#)" and "[Changes in JADE Release 7.0.04](#)", respectively, later in this document.
- Release 7.0.03 (the first general release of JADE 7.0), see [RelInfo7003.pdf](#), in your JADE **documentation** directory.

## C# Exposure and .NET

The following subsections contain the C# exposure and .NET changes in this release.

### .NET Exposure Generation (NFS 58838)

The JADE C# exposure wizard now enables you to specify a namespace for the generated C# that differs from the exposure name. The exposure name can be 30 characters in length only, and the namespace name can be any length up to 244 characters.

The C# Exposure wizard now provides the **NameSpace (exposure name is used if empty)** text box. When this text box receives focus and the text box is empty, the text is set to the name of the C# exposure and the entire text is selected. If the text box is empty, the C# exposure name is used to generate the:

- Namespace in the files
- Name of the project file
- Name of the **config** file

If this text box is not empty, the specified namespace text is used to generate these entities.

The namespace can contain only the characters **a** through **z**, **A** through **Z**, **0** through **9**, a period character (**.**), and an underscore character (**\_**).

---

**Note** As the value of the **NameSpace (exposure name is used if empty)** text box is not currently retained between initiations of the exposure wizard, you must specify it each time. (This may be implemented in a future major JADE release.)

---

### Format of the Generated Source (NFS 58847)

When the C# exposure is generated from JADE, the C# source now includes any text defined for an exposed class, property, and method, so that it can be carried over to .NET in the exposure for inline help. The generated format is:

```
/// <summary>
/// <text-line-1>
/// <text-line-2>
/// <...>
/// </summary>
```

## Converting a Partitioned Database (PAR 58782)

The **JadeConvertDb** application run from the non-GUI **jadclient** executable can now contain the optional **copySingleFileJadeBytes** parameter, which enables you to control whether single file **JadeBytes** instances are copied during the database conversion.

The default value is **true**; that is, single file instances are copied. If you want to copy these files manually, set the parameter to **false**; for example, if the total size of these files is large, you may want to copy them manually.

## Database Administration

The following subsections contain the database changes in this release.

### Multiple Workers for Database Compaction (NFS 48609)

Compacting multiple database files in parallel is now supported.

---

**Note** When you do not specify the number of workers or you specify a number less than two workers, serial compact actions are performed, as was the case in earlier releases.

---

### JadeDatabaseAdmin::*compactDbFiles* Method

The **JadeDatabaseAdmin** class now provides the **compactDbFiles** method, which has the following signature.

```
compactDbFiles (dbFiles:      DbFileArray;
                workers:      Integer;
                workPath:     String;
                updatesAllowed: Boolean);
```

The **compactDbFiles** method requests the database reorganization function compacts the database files in the **DbFileArray**.

The **workers** parameter enables you to specify the number of workers that you require. If you want to override the reorganization work directory, specify a valid path in the **reorgWorkDirectory** parameter.

The **updatesAllowed** parameter specifies whether an updating or a read-only compact is performed.

### Jade Database Utility Compact Operation

The **compact** command in the **jdbutilb** batch JADE Database utility now enables you to specify the optional **workers** parameter and an optional file list; that is, the syntax of the **Compact** operation is now as follows.

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

The **jdbutil** JADE Database utility Select Files to Compact dialog, accessed from the **Compact Files** command in the Operation menu, now contains the **Use multiple workers** check box and the **workers** text box, which are unchecked and display zero (**0**), respectively, by default. When you check the **Use multiple workers** check box, the **workers** text box is enabled so that you can specify the number of workers (greater than **1**) that you require.

### JADE Database Administration Utility Compact Action

The **Compact** action in the **jdbadmin** batch JADE Database Administration utility now enables you to specify the optional **workers** parameter and whether updates are allowed when multiple workers are in action. By default, updates are not allowed.



The syntax of the **Compact** action is now 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]
```

## JADE Monitor (PAR 57802)

The **Interrupt User** command in the JADE Monitor Users view now enables you 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.

## Transient File Compaction (PAR 57802)

You can now defragment a transient file, by using the **Process** class **compactTransientFile** method, which has the following signature.

```
compactTransientFile;
```

## Dynamic Property Methods

The **Class** class now provides the dynamic property methods documented in the following subsections.

### *compactDynamicPropertyClusters* Method

The **Class** class now provides the **compactDynamicPropertyClusters** method, which has the following signature.

```
compactDynamicPropertyClusters(interval: Integer;
                               receiver: Object;
                               callback: Method;
                               statistics: JadeDynamicObject);
```

This method compacts dynamic property clusters in which dynamic property instances were deleted. This method works by iterating all instances of the receiving class and subclass. For each instance, it locates any clusters that may contain deleted dynamic property values. If such a cluster exists, deleted values are removed and the cluster is compacted.

You can use this method to reduce the size of dynamic property clusters that contain values for dynamic property definitions that have been deleted.

The **interval**, **receiver**, and **callback** parameters enable you to invoke your own callback to allow the transaction to be committed and restarted periodically. The values of these parameters can all be null, in which case no callback is invoked. The **interval** parameter specifies the period in milliseconds between successive calls to the callback. The **receiver** parameter specifies the receiver of the callback.

The **statistics** parameter, which is also optional, enables some simple statistics to be returned (for example, the number of instances, the number of deleted values, and so on).

The signature of the callback method is:

```
callback(count: Integer): Boolean;
```

The **count** parameter specifies the number of instances in the current class that have been read. The return value is a continue or stop indicator; that is, continue when **true** or stop when **false**. The following is an example of the **compactDynamicPropertyClusters** method.

```
vars
    statistics      : JadeDynamicObject;
    instanceCount  : Integer;
    deletedValues   : Integer;
begin
    create statistics;
    beginTransaction;
    // must be in transaction state
        MyClass.compactDynamicPropertyClusters(60000, app, Application::callback,
            statistics);
    commitTransaction;
    instanceCount := statistics.getPropertyValue("instanceCount").Integer64;
    // number of instances of MyClass processed
    deletedValues := statistics.getPropertyValue("deletedValues").Integer64;
    // number of deleted dynamic properties removed
end;
```

The following is an example of the **callback** method defined in the **Application** class.

```
Application::callback(count: Integer): Boolean;
// callback called once a minute (60,000 milliseconds) when compacting
// dynamic property clusters
vars
    continueProcessing : Boolean;
begin
    continueProcessing := true;
    // limit the number of updates in the current transaction: close
    // and re-start the transaction
    commitTransaction;
    beginTransaction;
    // display progress information
    // allow the operation to be cancelled
    return continueProcessing;
end;
```

## ***deleteDynamicPropertyCluster* Method**

The **Class** class now provides the **deleteDynamicPropertyCluster** method, which has the following signature.

```
deleteDynamicPropertyCluster(clusterName: String[30]);
```

This method deletes a dynamic property cluster from the receiving class. (In JADE 7.0.05, you could not delete dynamic property clusters.)

An exception is raised if instances of the class or any subclasses exist.

## **Dynamic Property Clusters in Partitioned Database Files (PAR 58866)**

You can now add dynamic property clusters to classes mapped to a partitioned database file.

## **Database Production Mode (PAR 58555)**

You can now specify a value of **nochange** for the optional **production** parameter in the batch Schema Load utility (**jadloadb**).

In JADE 6.3 releases, this was interpreted as a value of **false** (that is, production mode was unset). Production mode was set only when the **production** parameter was set to **true**. When this parameter was not specified, the existing database mode was retained.

From JADE release 7.0, the **nochange** value of the **production** parameter is now interpreted as follows. If production mode is:

- Set (that is, enabled), it remains set.
- Unset (that is, disabled), it remains unset.

## Development Environment MDI Styles

The **Browser** sheet of the Preferences dialog in the JADE development environment contains the Mdi group box, which enables you to specify the type of Multiple-Document Interface (MDI) style to be displayed.

The Mdi group box contains the following option buttons.

- **Use Mdi**, which uses the standard MDI style (the default value).

This style is identical to earlier JADE releases.

- **Use Mdi With Tabs**, which displays a line of tabs above the MDI client window.

Each tab is associated with a displayed MDI child form and displays the form's caption. Clicking on that tab brings the associated form to the front. The MDI child forms can still be maximized, restored, and minimized within the MDI client area.

- **Use Tabs Only**, which displays a line of tabs above the MDI client window.

Each tab is associated with a displayed form and displays the form's caption. Clicking on that tab brings the associated form to the front. The forms are always maximized and cannot be restored or minimized. The parent of these forms is the associated tab sheet; not the MDI client window.

For the MDI with tabs and the tabs-only styles:

- Each tab displays a close button unless the **allowClose** property of the form is set to **false** (as is always the case for the Schema Browser). Clicking on the close button at the right of the tab closes the form.
- The width of each tab is restricted. If the caption is too large to be displayed, the left and right parts of the caption are displayed, separated by points of ellipsis (...).
- Class Browser tabs exclude the **Class Browser** part of the tab's caption, and display only the *schema-name: class-name* part.
- Moving the mouse over the tab displays a bubble help window containing the full form caption.
- Only those tabs that fit within a single line are displayed.
- The right of the tab strip displays a down arrow, which when clicked, displays a menu list of the MDI child forms in alphabetic caption text order with the currently active MDI child form checked. Clicking on a list entry brings that form to the top.
- You can alter the order of the tabs by clicking the tab of the active child form and dragging it left or right.
- If the MDI frame menu has a Window menu list, the menu list is displayed in descending order of the last access, instead of the creation order of the standard MDI style.
- Right-clicking on a tab displays a menu that provides the following commands.
  - **Close**, which provides the ability to close the form, and which is available only when the value of the **allowClose** property is set to **true**.
  - **Close All But This**, which closes all MDI child forms that have the **allowClose** property set to **true** except for the form associated with the tab.

- **Close All But Pinned**, which closes all MDI child forms that have the **allowClose** property set to **true** except for those that have been pinned.
- **Dock**, which docks a floating MDI child form back into the MDI frame. The docked position will be the saved position prior to the form being floated, unless the form was floated as part of the development environment restore process, in which case it will be docked at position 0,0.
- **Float**, which floats an MDI child form; that is, it takes an MDI child form out of the MDI frame and allows it to be moved independently from the MDI frame, (for example, on to another monitor on the PC).

The floated MDI child form is made a Window's child of the frame and will then always be above the MDI frame in z-order. The menus associated with the form when active remain in the MDI frame.

- **Pin**, which pins a tab by placing it to the left of all unpinned tabs in the tab list. The tab has a pinned icon drawn on the left of the tab. Clicking on the checked **Pin** command in the popup menu or clicking on the pinned icon unpins the tab.

Note that pinned tabs can be dragged only to a position within the pinned tab list. Similarly, an unpinned tab can be dragged only to a position within the unpinned tab list.

---

**Notes** For those MDI styles with tabs, right-clicking on the caption of an MDI child form displays the same menu as the one displayed when the tab is right-clicked. For the default standard MDI style, right-clicking the MDI child form caption displays a similar menu that provides all commands other than **Pin**. Standard MDI-style child forms can still be floated and docked.

The current MDI style is saved when the development environment is closed and restored when it is next initiated. In addition, when the **Save Windows** check box on the **Exit** sheet of the Preferences dialog is checked, the current order and pinned statuses of the saved MDI child forms is saved and will be restored when the development environment is next initiated. If a saved MDI child form is floating, it will be restored in its float state and position.

---

## Disk Cache Maximum Segments (PAR 58521)

The default value of the **DiskCacheMaxSegments** parameter in the [PersistentDb] section of the JADE initialization file is now calculated based on half of physical memory.

## Extracting a Method (PAR 58472)

When you extract a method to refactor it, if the calling method text is changed to **self.method-name**, the change is accepted, provided that the *method-name* in the call matches the generated method name when you click the **Add Method** button.

In addition, the Refactor Extract Method dialog now contains the **Include 'self.' in method call** check box, which you can check if you want to ensure that **self.** prefixes the method call. If **self.** exists in the method call, this check box is checked, so unchecking it removes **self.** from the method call.

The value of the check box is set automatically when the method name in the generated method is changed, based on whether **self.** prefixes the method call text.

## Hierarchy Browser History (PAR 58906)

The behavior of the JADE hierarchy browser history list differs between the three methods of selection, as follows.

- Clicking a history item in the History menu moves the item to the top of the list.
- Clicking the **History Back** and **History Next** buttons does not move the history item to the top of the list, and the item remains in its current position.
- Using the CTRL+ALT+left or right arrow keys shortcut does not move the item to the top of the list.

---

**Note** Unless you use the History menu, entries will be eligible to be removed earlier, because the last entry in the list is removed when a new entry is added and the list size exceeds 25.

---

## JADE Logical Certifier Utility (NFS 51392)

The JADE Logical Certifier diagnostic utility can now return error 31, which is detected when a collection is found to have a missing or invalid collection block. To repair this error:

```
rebuild coll
```

## JADE Version Information

The JADE Version Information utility (**jverinfo**) now enables you to specify the optional **noThinClients** command line parameter, which can be used to disable the version details of thin client download binaries, as follows.

- As well as listing the version details of executables and libraries found in the **bin** directory, **jverinfo** now lists the version information of thin client download binaries if they are in the default download locations (that is, it will not find the download location using the values defined in the JADE initialization file).

If the directory does not exist or no executable or library files are found, nothing is output.

The following is an example subset of directories.

```
i686-msoft-win32-ansi\download\bin
i686-msoft-win32-unicode\download\bin
armv4i-msoft-wm60-unicode\download\bin
i686-msoft-x86emu-unicode\download\bin
```

- The new optional **extraDirs=[file-name]** command line parameter, which you can use to list custom directories to scan for binaries to display version information.

If you do not specify the optional **=file-name** value, this parameter defaults to **jverinfo.extradirs**. The **file-name** value can be an absolute path or a relative path from the JADE Home directory (that is, one level up from the **bin** directory).

The file specified by the **extraDirs** parameter must contain ANSI strings; wide-character strings are not supported by any version of **jverinfo**.

If the directory does not exist or it contains no files, it is silently ignored and nothing is output.

- The new optional **noExtraDirs** command line parameter, which you can use to disable the **extraDirs** functionality (that is, ignore the default **jverinfo.extradirs** file, if it exists).

If the first character on an input line is the hash (**#**) character, the line is treated as a comment and is ignored.

The input line can be an absolute path or a relative path from the JADE Home directory (that is, one level up from the **bin** directory).

---

**Note** The **jverinfo** command line options are case-insensitive.

---

## Schema Inspector (PAR 54665)

The JADE Schema Inspector application has been moved from the **RootSchema** schema to the **JadeMonitorSchema** schema and the application name has changed from **Schemainspector** to **JadeSchemainspector**.

When your system has application restrictions (that is, the **EnableAppRestrictions** parameter is set to **true** in the [JadeAppServer] section of the JADE initialization file), the **JadeMonitorSchema** must be specified as a schema that can be executed from presentation clients attached to the application server; for example:

```
AllowSchemaAndApp2 = JadeMonitorSchema
```

---

**Note** The old schema and application from earlier releases will continue to function, because the started **SchemaInspector** application invokes the JADE Schema Inspector application using the **JadeMonitorSchema** schema and **JadeSchemaInspector** application names.

---

## Silverlight XAP File Generation using jadclient (PAR 58614)

When the Silverlight XAML presentation client (XAP) generation process was performed using the **jadclient** command line to execute the JadeSchema **XapFileBuilder** application and the generation failed, the **jadclient** exit code returned zero (**0**).

As this did not provide a way to automatically detect and handle the failure, when you run the **XapFileBuilder** application in **jadclient** and the generation fails, the exit code is now set to **1**; successful generation returns zero (**0**). Reasons for failure include:

- Invalid arguments for the XAP build process
- When a JADE method has not been compiled
- When a JADE method is in error
- When a syntax error occurs in the generated C# compile

## Synchronized Database Service (PAR 56965, 57901)

On SDS secondaries, a restart recovery mechanism has been implemented to ensure integrity is re-established after restart. This restart recovery mechanism applies journal updates beginning at the farthest-back Log Sequence Number (LSN), where the oldest incomplete transaction starts, and continuing through journals until all available changes have been applied (this ensures that the files are synchronized with the journals).

The recovery then performs a "revert" pass, which undoes the effects of all transactions back to farthest back LSN, which ensures that the database is in a consistent state before resuming replay.

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

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

JADE now implements an SDS secondary restart recovery audit. These journals, which have the **.rlog** extension, are numbered created, written, and removed the same as the normal **.log** journals. When a block in a file is made free, information from the block header is audited to the **.rlog** journal so that the "revert" pass can correctly recreate block state when reversing that free operation, going backwards undoing the effects of the transactions. The **.rlog** and **.log** journals are structurally identical, and can be selected for dumping using the JADE Database utility (**jdbutil**). The **.rlog** journals have no operational requirements; they are created and removed as necessary by secondary replay and restart recovery.

## Thin Client Automatic Download (PAR 57876)

In earlier releases, if the **JadeConfig.cpl** file was not present in the **Windows** directory on a mobile device, a download of that file was required. If the automatic download process was turned off, the presentation client refused to run until that file was present.

From this release, the **JadeConfig.cpl** file is required only if other files also need to be downloaded or a download check is forced by changing the value of the **DownloadVersion** parameter in the [JadeAppServer] section of the JADE initialization file on the application server.















## Windows



















The following subsections contain Windows user-interface (GUI) class-related changes in this release.

### Draw Fill Styles (NFS 58716)



















The **Window** class **drawFillStyle** property can now contain 48 additional draw fill styles, which are represented by Window class constants.







The settings of the **drawFillStyle** property (including those in the Integer value range **0** through **7** that were available in earlier releases) are listed in the following table.

Window Class Constant	Integer Value	Image
DrawFillStyle_Solid (the default value)	0	
DrawFillStyle_Transparent	1	
DrawFillStyle_HorzLine	2	
DrawFillStyle_VertLine	3	
DrawFillStyle_UpDiagonal	4	
DrawFillStyle_DownDiagonal	5	
DrawFillStyle_Cross	6	
DrawFillStyle_DiagonalCross	7	
DrawFillStyle_4DotDiamond99	8	
DrawFillStyle_EveryOther	9	
DrawFillStyle_UpDiagonal4	10	
DrawFillStyle_VertLine4	11	
DrawFillStyle_HalfUpDiagonal	12	
DrawFillStyle_HorzWaves4	13	

Window Class Constant	Integer Value	Image
DrawFillStyle_Triangles	14	
DrawFillStyle_Cross55	15	
DrawFillStyle_4DotDiamond95	16	
DrawFillStyle_FilledDiamond	17	
DrawFillStyle_DownDiagonal4	18	
DrawFillStyle_HorzLine4	19	
DrawFillStyle_HalfDownDiagonal	20	
DrawFillStyle_HorzWaves3	21	
DrawFillStyle_DottedCross	22	
DrawFillStyle_Cross99	23	
DrawFillStyle_4DotDiamond55	24	
DrawFillStyle_Checkered	25	
DrawFillStyle_DbleUpDiagonal	26	
DrawFillStyle_VertLine2	27	
DrawFillStyle_HorzDash	28	
DrawFillStyle_DownRectangle	29	
DrawFillStyle_8DotDiamond99	30	
DrawFillStyle_AltSquares2	31	



Window Class Constant	Integer Value	Image
DrawFillStyle_4DotDiamond53	32	
DrawFillStyle_Rev4DotDiamond55	33	
DrawFillStyle_DbleDownDiag	34	
DrawFillStyle_HorzLine2	35	
DrawFillStyle_VertDash	36	
DrawFillStyle_HorzRectangle	37	
DrawFillStyle_UpKeyShape	38	
DrawFillStyle_AltSquares4	39	
DrawFillStyle_8DotDiamond55	40	
DrawFillStyle_Rev4DotDiamond95	41	
DrawFillStyle_TripleDownDiag	42	
DrawFillStyle_DbleVertLine	43	
DrawFillStyle_Speckled	44	
DrawFillStyle_Interlocked	45	
DrawFillStyle_ReverseHorzDash	46	
DrawFillStyle_DiagonalHatch	47	
DrawFillStyle_InvertedCross	48	
DrawFillStyle_Rev4DotDiamond99	49	

Window Class Constant	Integer Value	Image
DrawFillStyle_TripleUpDiagonal	50	
DrawFillStyle_DbleHorzLine	51	
DrawFillStyle_PatchworkSquares	52	
DrawFillStyle_Tartan	53	
DrawFillStyle_ShinyBalls	54	
DrawFillStyle_Balls	55	

When the **drawFillStyle** property is set to **DrawFillStyle\_Transparent** (1), the **drawFillStyle** property is ignored. In addition, draw fill styles are not available in Compact JADE (that is, the **drawFillStyle** property is ignored).

## Drawing Form and Control Elements

The following subsections describe the changes to the drawing of some form and control elements in this release.

### Skins

If an application is active with a skin set, a second application initiated from the same **jade.exe** executable is now drawn using the current Windows theme if that application does not have a skin set.

In earlier releases, the application turned off Windows theming while the first application was still active.

### Menus

Menus are now drawn in the newer .NET style. A menu is drawn with a left gutter border, uses new icons for the check mark symbol and submenu icons, and draws the selected background using the current Windows theme set.

In addition, menus activated from the form's menu are drawn as though the form menu item is part of the activated menu. The form's menu bar is also drawn using the current Windows theme. The exception to this is when a skin is currently active for the form, in which case the skin definition of any menu elements is still used instead.

### Button Control

Buttons are now drawn using the current Windows theme that is in use, unless the currently active JADE skin defines the button look and feel.

The exception to this is any button that has the button **backColor** value set to any color other than the default. In this case, the button is drawn in the JADE style of earlier releases.

### CheckBox Control

Check boxes are now drawn using the current Windows theme that is in use, unless the currently active JADE skin defines the check box look and feel.

## OptionButton Control

Option buttons (radio buttons) are now drawn using the current Windows theme that is in use, unless the currently active JADE skin defines the option button look and feel.

## Changes in JADE Release 7.0.05

This section contains details about changes in JADE release 7.0.05.

For details about changes in JADE release 7.0.04, see "[Changes in JADE Release 7.0.04](#)", later in this document. For a summary of changes and new features in JADE 7.0.03 (the first general release of JADE 7.0), see [RelInfo7003.pdf](#), in your JADE **documentation** directory.

### Database Identities

A JADE database has three Universally Unique Identifiers (UUID) identity 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** parameters 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 7.0.0.4, Copyright (c) Jade Software Corporation
Limited 2010
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 is a new method in this release that enables you to programmatically obtain 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 is created and registered. Each database should have a unique value.

If you clone a database from an existing one, use the **generateUUID** command in the **jdbutilb** batch JADE Database utility to give the new database a new unique identity.

The database is given a new identity automatically when:

- The database role in an SDS environment changes from *undefined* to *primary*.
- 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 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 for use other than as an SDS secondary, use the **generateEnvironmentIdentity** command in the **jdbutilb** batch JADE Database utility to give the new database a new unique environment identity.

---

**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 an SDS secondary database from an existing one, use the **generateServerIdentity** command in the **jdbutilb** batch JADE Database utility to give the new database a new unique server identity.

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.

## Format of the Server Uniform Resource Identifier (URI) String

Use the **server** parameter in a command line to specify the server Universal Resource Identifier (URI) target database and the client-server transport, instead of the **server** and **path** parameters in a command line and the **ServerNodeSpecifications** parameter in the [JadeClient] section of the JADE initialization file.

The syntax of the server URI command line parameter is as follows.

```
server=scheme://target-address/environment-UUID[/server-UUID] [?parameters]
```

The *scheme* and *target-address* values specify the client-server transport definition. The environment identity UUID (in display form) specifies the database to which the client node is connecting. This is optionally followed by the database server identity UUID.

Specify both environment and server identities when you want the client node to connect to a specific SDS secondary database.

The following subsections explain how you can specify the target address for each URI scheme.

## File Scheme (Single User)

The format of the single user **File** scheme is as follows.

```
File://database-directory
```

The *database-directory* value specifies the name of the directory where the database server finds the database control file (**\_control.dat**). Use this format when running a single user client node; for example:

```
File://c:\jade\system
```

This example is equivalent to **server=SingleUser** and **path=c:\jade\system** in the command line.

## Tcplp, Tcplpv4, and Tcplpv6 Schemes

The format of the target address in the **Tcplp**, **Tcplpv4**, and **Tcplpv6** schemes is as follows.

```
host:port-number
```

You can specify the host using a:

- Simple host name; for example, **george**, **server129**, or **localhost**.
- Fully qualified domain name (FQDN); for example, **wilbur.example.com**.
- Dotted decimal IPv4 address; for example, **127.0.0.1**.
- Quadded hexadecimal IPv6 address enclosed in square brackets; for example, **[fe80::bdf1:cbe:6902:7deb]**.

The **Tcplpv4** scheme uses the IPv4 protocol, so quadded hexadecimal IPv6 addresses are not allowed. (**Tcplp** is a synonym for **Tcplpv4**.)

The **Tcplpv6** scheme uses the IPv6 protocol, so dotted decimal IPv4 addresses are not allowed.

The *port-number* value is a decimal integer in the range **1** through **65535**.

When the host is a simple host name or an FQDN, name resolution always returns **Tcplpv6** addresses before **Tcplpv4** addresses. When a specific protocol version is not specified, the first returned address is used.

The following examples show target addresses for the **Tcplp**, **Tcplpv4**, and **Tcplpv6** schemes.

```
server=TcpIp://localhost:6005/48cf13df-bf6d-df11-87e2-2e5925024153
```

```
server=TcpIpv4://host.company.com:6005/48cf13df-bf6d-df11-87e2-2e5925024153?localhostname=aHostName
```

```
server=TcpIpv6://[fe81::6fe:7fff:fe87:bd20]:6005/48cf13df-bf6d-df11-87e2-2e5925024153?localPort=54321
```

```
server=TcpIp://127.0.0.1:6005/48cf13df-bf6d-df11-87e2-2e5925024153/48cf13df-bf6d-df11-87e2-2e5925024153
```

The following optional parameters enable you to specify the client node source address and port.

- **LocalInterface=host**

The *host* value is a simple host name, a fully qualified domain name, or an appropriate IP address. You can use this to select a specific source address when the client machine has more than one network interface.

- **LocalPort=port-number**

The *port-number* value is a decimal integer in the range **1** through **65535**. You can use this when the default random source port number cannot be used; for example, when the client and server machines are separated by a firewall, which limits the client source port numbers.

## Hybrid Pipe Shared Memory (HPSM) Scheme

In the Hybrid Pipe Shared Memory (**HPSM**) scheme, shared memory is used to establish a connection so the client node must be on the same machine as the server. The format of the target address is as follows.

```
localhost:base-name
```

The host must be **localhost**. The *base-name* contains up to 60 ASCII graphic characters excluding the forward slash (/), percent (%), and comma (,) characters. When the database server is installed as a Windows service, the *base-name* should be prefixed with **Global**.

The following example shows a target address for the **HPSM** scheme.

```
server=HPSM://localhost:SRCJade-HPSM/48cf13df-bf6d-df11-87e2-2e5925024153
```

## JadeLocal Scheme (Shared Memory)

In the **JadeLocal** scheme, shared memory is used to establish a connection so the client node must be on the same machine as the server.

The format of the target address is as follows.

```
localhost:base-name
```

The host must be **localhost**. The *base-name* contains up to 60 ASCII graphic characters excluding the forward slash (/), percent (%), and comma (,) characters. When the database server is installed as a Windows service, the *base-name* should be prefixed with **Global**.

The following example shows a target address for the **JadeLocal** scheme.

```
server=JadeLocal://localhost:SRCJade-SHM/48cf13df-bf6d-df11-87e2-2e5925024153
```

## Debugger Initialization (PAR 57923)

From this release, the JADE debugger is initiated earlier than previously occurred.

As a result, the debugger will now stop on the first logic line encountered; that is, in the **Global** class **getAndValidateUser** or **isUserValid** method, if present, otherwise the first logic line in the **Global** class **initialize** or **Form** class **load** event methods. Breakpoints in the **getAndValidateUser** and **isUserValid** methods are now also honored.

In earlier releases, you could not debug the **getAndValidateUser** and **isUserValid** methods.

## Development Environment

The following subsections contain the changes to the JADE Development Environment in this release.

### AutoComplete

The following subsections contain the AutoComplete enhancements in this release.

### Colon Character Recognition (PAR 58199)

The colon (:) character, which is now recognized by the **AutoComplete** list box, indicates that the current selection should be selected, followed by entry of the colon character. The complete list of recognized characters is now: space, period (dot), left and right parentheses, left and right square brackets, semicolon, comma, and colon (that is, ".()[];:").

## Default AutoComplete Setting (PAR 58351, NFS 57721)

The AutoComplete feature is now turned on by default when:

- A new user creates a profile.
- You are an existing user and you click the **Defaults** button on the Preferences dialog.

## Recognition of Parameter Usage Types (PAR 57358)

The AutoComplete feature now prompts for the valid parameter usage types of **constant**, **input**, **io**, and **output** after you have entered the first character after a method parameter type.

## JADE Painter

The following subsections describe the changes to the JADE Painter in this release.

### Loading a Form into the JADE Painter (NFS 47177)

On the Load Form dialog in the JADE Painter, you can now use the up (↑) and down (↓) arrow keys to scroll up or down the list of forms when the **Form** text box has focus, so that the form selected in the list is displayed in the **Form** text box.

### Menu Design Form (NFS 46595)

When the Menu Design form of the JADE Painter is opened, it is now sized so that the width and height are larger than the required sizes needed to display the menu top line and the display of any second-level menu list with room for additional entries.

The maximum height and width are now the height and width of the work area on the monitor on which the form is displayed. It remains without a control box and maximize button.

## Refactoring

The following subsections contain the refactoring enhancements in this release, available from menu items of the **Refactor** command in the Edit menu. This enables you to restructure a component of an existing method.

---

**Note** This functionality is enabled only when the AutoComplete feature is turned on (that is, the **Use AutoComplete** check box is checked in the Display Options group box on the **Editor Options** sheet of the Preferences dialog).

---

The result of a refactoring action is displayed in the status line; for example:

```
Local variable docExample : String was added to the method vars
```

The following JADE configurable shortcuts have been added to the **Editor Key Bindings** sheet of the Preferences dialog, to support refactoring.

- CTRL+F2 displays the Refactor menu at the cursor position. This menu is also displayed when you select the **Refactor** command from the Edit menu or by right-clicking in the editor pane and then selecting the **Refactor** command.
- SHIFT+F2 executes the **Rename / Change** command from the Refactor submenu of the Edit menu. This option will attempt to process the identifier under the cursor to perform a rename or change.



## Declaring an Unknown Local Identifier as a Local Variable (NFS 58186)

When editing a JADE method, you can now declare an unknown local identifier as a local variable. To do this:

1. Position the cursor on the identifier.
2. Right-click and then select the **Declare Local Variable** command from the Refactor submenu of the Edit menu.
3. If the identifier is assigned by an expression or it is used as an assignment value, the type of the local variable is determined from that expression. If the type cannot be determined or it is not an assignment expression, a dialog is displayed. In the **Variable Type** combo box, select the type of the local variable and then click the **OK** button.

Text in the status line displays the result of the refactor action; for example:

```
Local variable docExample : String was added to the method vars
```

A local variable of the determined or selected type is added to the **vars** section of the method, which is created if it does not exist. The cursor remains at its previous position.

Examples of an unknown **indx** local identifier in a method are as follows.

```
indx := 0;

posn := indx;
```

Positioning the cursor over **indx**, where this is undefined, would result in the following code fragment.

```
vars
...
    indx : Integer;
begin
    ...
```

## Extracting Existing Logic and Creating a New Method (NFS 58231, PAR 58351)

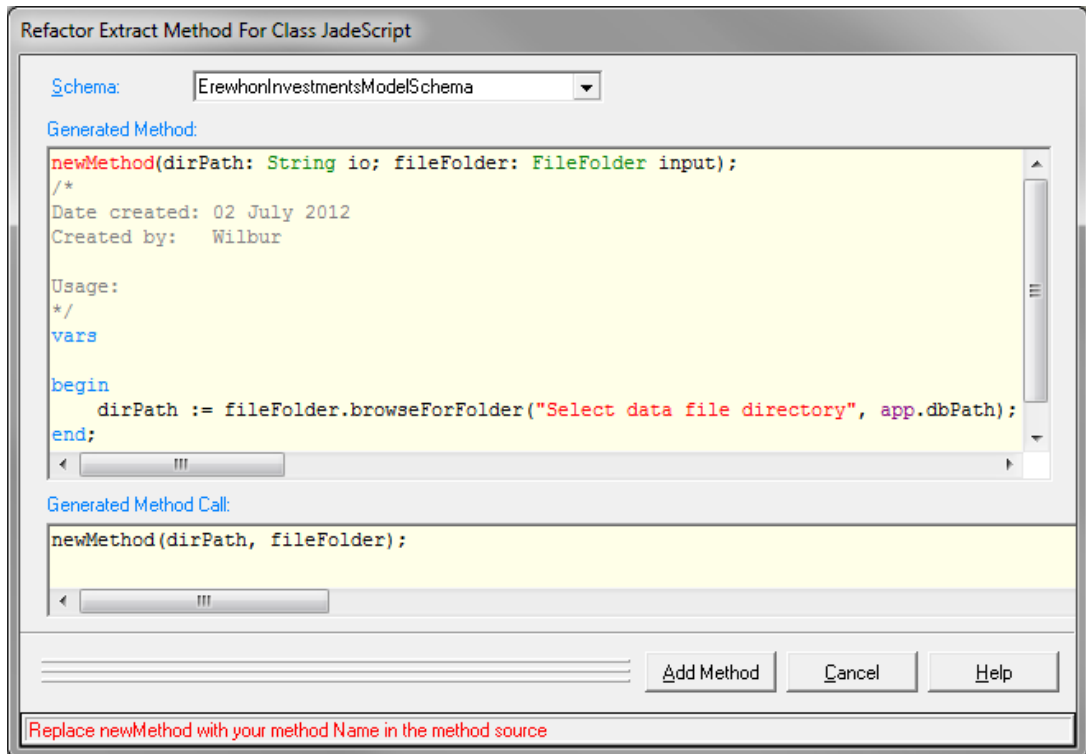
The JADE editor now enables you to select logic within a method and create a new method in the currently selected class with that logic.

### » To refactor an extracted method

1. Select the logic to be extracted.
2. Right-click and then select the **Extract Method** command from the Refactor submenu of the Edit menu.

If the start of the selection includes a whole expression, the Refactor Extract Method for Class *class-name* dialog is then displayed. If it does not include a whole expression, an error message is displayed.

The **Generated Method** editor pane contains the source of the new method, consisting of the parameter signature determined by the analysis of the selected logic, the method template expansion if you have a template defined in your text templates preferences, and the selected logic, as shown in the following diagram.



The status line initially displays:

```
Replace newMethod with your method Name in the method source
```

3. The ENTER and ESC keys initially control the dialog cancel and commit functions.
  - When the dialog is first displayed, a transparent text box containing the text **newMethod** overlays the method name in the **Generated Method** editor pane, and it has focus.
 

This text box, which has a maximum length of 30 characters, must have a lowercase first character.
  - As you change the method name, the text box is updated and resized to fit, and the actual method source, template entry (if present), and method call text in the **Generated Method Call** editor pane are all adjusted.
  - The method name text is shown in red, which indicates that you can use the ESC and ENTER keys to cancel the dialog or to perform the add method action, respectively.
  - You can use the right and down arrow keys, the TAB key, and the mouse to move to the **Generated Method** editor pane itself.
  - The text box is hidden as soon as it loses focus, the method name is displayed in its normal color in the editor pane, and the ESC and ENTER keys no longer cancel or commit the dialog, because they are now editor-related.
  - You can still change the method name in the editor pane before you click the **Add Method** button.

---

**Notes** If you change the signature of the method, the text of the call that will replace the original selected text in the source method is not changed. You should also change this text in the editor pane of the Class Browser to meet your requirements.

If the text prior to the original text that you selected is a replacement statement and you selected one statement only, the return type of the method is set to the replacement type; for example, if the original method has the statement `int := (a * b + c);` and you select `(a * b + c)`, the result is a method return type of **Integer**.

---

4. In the **Schema** combo box, select the schema in which the method is to be created. This combo box, in which the current schema is selected by default, lists all of the schemas in which the method could be created.
5. Click the **Add Method** button, to add the new method and to replace the selected source with a call to the new method. Alternatively, click the **Cancel** button to cancel the current action and return focus to the original editor pane in the Class Browser.

When you click the **Add Method** button, the new method is added to the currently selected class in the selected schema, using the method name specified in the **Generated Method** edit pane, and the selected content of the original source method is replaced with the contents of the **Generated Method Call** editor pane. This last change is not committed, so you can undo it by using the CTRL+Z shortcut keys.

If the new method compiles successfully, the Refactor Extract Method for Class *class-name* dialog is closed and focus returns to the Class Browser editor pane of the original method.

If the new method fails to compile:

- The Refactor Extract Method for Class *class-name* dialog remains open and states that the new method was added but that the compile failed. The first error is displayed and the source generating the error is selected.
- The **Generated Method Call** editor pane is hidden and a message is displayed, stating that the original selection has been replaced by the text that was in the **Generated Method Call** editor pane.
- The **Add Method** button is renamed **Recompile**. Clicking this button attempts to recompile the method. If the recompile succeeds (following your changes to the method source), the dialog is closed.
- Clicking the **Cancel** button leaves the new method in error and the original method changed.
- The new method logic calls a method where the parameters are usage **io** or **output** and JADE has not detected that the parameter for the new method signature should be **io** or **output** (that is, no assignment is done on the variable in the selected logic or it is not used as the variable of a **foreach** instruction).

Change the signature of the new method, to handle this situation.

The following is an example of a code fragment selected in the original source (**Generated Method**) editor pane.

```
d := (a * b + c);
```

The following example is the generated method in which the whole line is selected (where **doCalc** has been specified as the method name).

```
doCalc(d: Integer io; a: Integer; b: Integer; c: Integer);
vars
begin
    d := (a * b + c);
end;
```

The original statement then becomes:

```
doCalc(d, a, b, c);
```

If (**a \* b + c**) were selected, the method would be:

```
doCalc(a: Integer; b: Integer; c: Integer): Integer;
vars
begin
    return (a * b + c);
end;
```

## Generating a Method Stub from the Editor Pane (NFS 58158, PAR 58389, PAR 58388)

When editing a JADE method, you can now generate a method stub (whose logic must be filled in later) from a call statement. To do this:

1. Specify a method call with all of the required parameters.
2. Position the cursor over the new method name and then select the **Generate Method Stub** command from the Refactor submenu of the Edit menu.

A new method stub is then defined on the receiver of the method call. The types of the parameters are determined from the call statement. If the method call is assigned, the method return type is the type of the assignment variable.

If the receiver of the method is not in the current schema, the Select Schema dialog is displayed, prompting you to select the schema in which to add the method. For a class, the schema of the root of the class is selected by default. For a primitive method, the current browser schema is selected by default.

An example of an undefined method call is:

```
bool := cust.anewMethod(custName, bank.id, descStr);
```

This method call results in the creation of the **anewMethod** method stub on the **Customer** class, as follows.

```
anewMethod(custName: String;
           integer1: Integer;
           descStr: String): Boolean;
vars
begin
    return null;
end;
```

When generating a method stub from the editor pane, note that:

- The parameter can be an identifier or dot (.) expression only.  
  
The *Cannot determine the type of a parameter for defining the method* message is displayed if the parameter is not a single expression; for example, **2 + 4** will fail because the AutoComplete feature is not a compiler and therefore does not understand the **+** operator and the resulting type.
- If successful, text in the status line displays the schema, class, and name of the method that was created. If rejected, a message box displays the reason why the method stub generation was not performed.
- The method name must begin with a lowercase character.
- All of the elements in the call statement, apart from the method name, must be known elements.
- Any template defined for methods in your user preferences is invoked.
- The JADE development security library is called, to ensure that you have access to the schema in which the method will be defined when it is not the currently selected schema.
- The generated method compilation could fail, and a warning message be displayed; for example, if the parameter types are not visible in a superschema.

- If the method is defined on a superclass or subclass, the generation will be rejected if the signature of the method does not match. A warning will also be displayed, advising you that the new method will be a re-implementation or superclass instance.
- If the method is defined on a superclass or subclass, the parameter names and return type is taken from the existing method.
- If the parameter of the method call is a local variable, parameter, or local constant, the name will be used as the parameter name in the method (with a numeric postscript, if the same variable is used more than once).
- Other types of parameters will use the type name of the parameter with a numeric postscript; for example, **integer1**.
- If the current unary expression selected in the editor pane is preceded by the **write** instruction, as shown in the following code fragment, a return type of **String** is assumed.

```
write accountToReportOn.toString();
```

The current unary expression preceded by **unary-expression :=** also handles the type of the expression.

## Promoting a Local Constant to be a Type Constant (NFS 58179)

When editing a JADE method, you can now promote a local method constant to a constant on the type in which the method is defined. To do this:

1. Select the local constant reference in the method (in the logic or in the **constants** section).
2. Select the **Promote to Type Constant** command from the Refactor submenu of the Edit menu. The Add constant to *schema-name::type-name* dialog is then displayed.

The name of the constant in the dialog is set to the local constant name. The value and the type of the constant are set in the editor pane.

3. Set any attributes that you require and then click the **OK** button.

The constant is then added to the type (that is, the class or primitive type) and the local constant definition is removed from the method.

If the line containing the constant definition will then be empty, the entire line is removed. If there is any other text on the line (including comments), only the constant definition portion is removed.

To undo the text change in the method, press CTRL+Z. The constant definition is not returned.

## Promoting a Local Variable to be a Parameter of the Current Method (NFS 58190, PAR 58390)

When editing a JADE method, you can now promote a local variable to be a parameter of the method. To do this:

1. Position the cursor on the name of a local variable (in logic or in the **vars** section).
2. Right-click and then select the **Promote to Parameter** command from the Refactor submenu of the Edit menu.

The local variable is then removed from the local variables section and added as the last parameter of the method. Any usages of the method must be edited to include the parameter. The signature change does not take effect until you compile the method. To undo the change, press CTRL+Z.

If you attempt to promote a user identifier to be a method parameter and the identifier is not a local method variable, an error message of the following form is displayed.

```
Identifier identifier-name is a type-name not a local variable.
```

For example, if you attempt to promote an existing method parameter called `saleItem`, the following error message would be displayed.

```
Identifier saleItem is a method parameter not a local variable.
```

## Promoting a Local Variable to be a Property on the Current Class (NFS 58174)

When editing a JADE method, you can now promote a local variable to a property on the class in which the method is defined. To do this:

1. Select the local variable reference in the method (in the logic or in the **vars** section).
2. Select the **Promote to Class Property** command from the Refactor submenu of the Edit menu.

If the caret is positioned in a local variable, the Define Attribute or Define Reference dialog is displayed, depending on the type of the local variable.

The name of the property in the dialog is set to be the local variable name and the dialog property type is set to the type of the local variable.

3. Set any attributes that you require and then click the **OK** button.

The property is then added to the class and the local variable definition is removed from the method.

If the line containing the variable definition will then be empty, the entire line is removed. If there is any other text on the line (including comments), only the relevant variable definition portion is removed.

To undo the text change in the method, press CTRL+Z (the property definition will remain).

## Promoting a Method Value to be a Local Method Constant (NFS 58182)

When editing a JADE method, you can now promote a method string or numeric value to a local method constant. To do this:

1. Click on a numeric or string value within the logic of the method.
2. Select the **Promote to Method Constant** command from the Refactor submenu of the Edit menu.

A dialog is then displayed, showing the selected value.

3. In the **Constant Name** text box, specify the name that you require for the local method constant.
4. Click the **OK** button.

The constant is then added to the **constants** section of the method (which is created if it does not exist) and all instances of the use of that constant in your method (but not in comments) are replaced by the specified name.

To undo the change or changes, press CTRL+Z to undo each change in that method.

## Renaming or Changing an Entity from the Editor Pane (NFS 58129)

The JADE editor pane, when displayed within a Class Browser, now enables you to position the cursor on almost any identifier (for example, a local variable or a method) within the body of a method and then select the **Rename / Change** command from the Refactor submenu of the Edit menu (or press SHIFT+F2). Alternatively, you can select the whole identifier.

---

**Note** The JADE development security library is called, to ensure that you have access to the schema in which the entity will be renamed when it is not defined in the currently selected schema.

---

The result of the refactor action depends on the type of the identifier, as follows. When the:

- Identifier is a local constant, variable, or method parameter, the Rename *entity-name* dialog is then displayed so that you can specify the required name in the **New Name** text box.

When you specify the new name and click the **OK** button, the local identifier is renamed throughout the method and the cursor is positioned after the identifier.

To undo a change, press CTRL+Z for each identifier position in the method.

---

**Note** The change or rename action is performed, even if the method has been modified and is unsaved.

---

- Variable is a user-defined global constant, the Global Constant maintenance dialog is displayed for that constant definition, regardless of the schema in which the global constant is defined. You can change or rename the global constant definition.

After the change or rename action is committed, the cursor is positioned after the global constant name.

You cannot use CTRL+Z to undo this change.

---

**Notes** This action is rejected if the method is changed or locked, because changes made will cause that method to be recompiled.

This action is equivalent to browsing to the global constants in the schema in which it is defined and then selecting the **Change** command from the Constants menu.

---

- Variable is a method, the Rename Method dialog is displayed, regardless of the class or schema in which the method is defined.

After the rename action is committed, the cursor is positioned after the method name.

You cannot use CTRL+Z to undo this change.

---

**Notes** This action is rejected if the method is changed or locked, because changes made will cause that method to be recompiled.

This action is equivalent to browsing to the method in the Class Browser and then selecting the **Rename** command from the Methods menu.

---

- Variable is a class constant, the Define Constant dialog is displayed, regardless of the schema or class in which the class constant is defined.

After the change or rename action is committed, the cursor is positioned after the constant name.

You cannot use CTRL+Z to undo this change.

---

**Notes** This action is rejected if the method is changed or locked, because changes made will cause that method to be recompiled.

This action is equivalent to browsing to the class constant in the Class Browser and then selecting the **Change** command from the Constants menu.

---

- Variable is a class property, the respective Define Attribute or Define Reference dialog is then displayed, regardless of the schema or class in which the property is defined.

After the change or rename action is committed, the cursor is positioned after the property name.

You cannot use CTRL+Z to undo this change.

---

**Notes** This action is rejected if the method is changed or locked, because changes made will cause that method to be recompiled.

This action is equivalent to browsing to the property in a Class Browser and then selecting the **Change** command from the Properties menu.

---

- Variable is a class name, the Define Class dialog is displayed, regardless of the schema in which the class is

defined.

After the change or rename action is committed, the cursor is positioned after the class name.

You cannot use CTRL+Z to undo this change.

---

**Notes** This action is rejected if the method is changed or locked, because changes made will cause that method to be recompiled.

This action is equivalent to browsing to the class in a Class Browser and then selecting the **Change** command from the Classes menu.

---

If the cursor is not positioned on one of the variables in this list or it is a package or an interface entity, a message box is displayed advising you that the cursor is not positioned on an entity that can be changed or renamed in this way.

The action can also be rejected because the entity cannot be changed for a variety of standard reasons or you are not entitled to change that entity (that is, security library rules apply).

---

**Note** Normal versioning and patch versioning rules apply, based on the schema of the entity that is changed.

---

## Search Functionality

The following subsections contain the search enhancements in this release.

### Finding an Entity in a Schema View (NFS 43327, 43794)

In earlier releases, pressing F4 from a Schema View of the Class Browser displayed the Find Type dialog, which contained only those classes already added to the Schema View.

From this release, the Find Type dialog contains all classes relevant to the currently selected schema. If you select a class that is not in the current Schema View, a message box prompts you to confirm that you want to add the class to the Schema View when you click the **Current Browser** or **New Browser** button.

When you select a class from the list of all classes on the **Select Required Entry** list box and then click the:

- **New Browser** button, a *new* Class Browser for the same Schema View is then opened, with that class selected.
- **Current Browser** button, that class is selected in the current Class Browser of the same Schema View.

As the **Current Browser** and **New Browser** buttons apply only to Class Browsers, the Find Type dialog for entities other than a class retain the **OK** button of earlier releases instead of the **Current Browser** button.

### Finding an Entity in the Schema in which the Class is Defined (NFS 49367)

From this release, when you press F4 from the Class Browser of the schema in which the class is defined, the Find Type dialog now provides the **Current Browser** and **New Browser** buttons.

When you select a class from the list of all classes on the **Select Required Entry** list box and then click the:

- **New Browser** button, a new Class Browser in the schema in which the class is defined is opened, with that class selected.

---

**Note** You can browse to that schema only if the JADE development security library rules enable you to do so.

---

- **Current Browser** button, that class is selected in the current Class Browser.

As the **Current Browser** and **New Browser** buttons apply only to Class Browsers, the Find Type dialog for entities other than a class retain the **OK** button of earlier releases instead of the **Current Browser** button.



## Locating Possible Transient Object Leaks (NFS 58062)

You can now scan all methods in a schema or class for possible transient leaks involving local variables and method parameters.

The Schema menu in the Schema Browser now contains the **Find Possible Transient Leaks** command.

When you select this command, the Find Possible Transient Create Leaks dialog is then displayed, with the name of the schema selected in the Schema Browser displayed in the first combo box in the Search Criteria group box.

1. If you do not want to search for possible transient leaks in all methods in the schema selected in the Schema Browser (the default value), perform one of the following actions in the Search Criteria group box. Select:
  - Another schema in the combo box below the **Search selected schema for all creates** option button if you want to locate possible transient object leaks in a schema other than the one currently selected in the Schema Browser.
  - The **Search all schemas** option button if you want to locate possible transient leaks in all methods in all schemas.
  - The **Search selected schema and all sub-schemas** option button and in the combo box above the option button, select the schema in which you want to locate possible transient leaks in all methods in that schema and its subschemas.
  - The **Search for creates of selected class in selected schema** option button and in the combo box below the option button, select the class of the schema selected in the first combo box or schema that has a local copy class whose methods will be scanned for possible transient leaks.
2. Uncheck the **Report created objects not deleted** check box if you do not want to locate any transient **create** statement leaks on local variables or method parameters where there is no delete of that variable and that variable was not returned by the method and was not passed as a parameter to another method.

By default, this check box is checked, indicating that created objects are checked for deletion.
3. Uncheck the **Report created in loop but not deleted in loop** check box if you do not want to locate possible leaks where a create is performed inside a loop but the delete is outside of the loop. This may not actually be a leak, because only one create may have been performed.

By default, this check box is checked, indicating that objects created in a loop are checked for deletion within that loop.
4. Check the **Report objects not deleted but assigned** check box if you want to locate creates performed where there is no delete, but the variable was assigned to another property or variable. This may not represent a leak, as it is not known whether this value is subsequently deleted.

By default, this check box is unchecked, indicating that assigned objects are not checked for deletion.
5. Check the **Report objects not deleted but passed to other methods** check box if you want to locate creates performed where there is no delete, but the variable was passed as a parameter to another method. This may not represent a leak, as it is not known what subsequently happens to that object.

By default, this check box is unchecked, indicating that objects passed to other methods are not checked for deletion.
6. Check the **Report created objects returned from method** check box if you want to locate creates performed where the object is returned from the method. This may not represent a leak, as it is not known what the calling method will do with that object.

By default, this check box is unchecked, indicating that objects returned from methods are not scanned for possible leaks.
7. Check the **Report deletion not performed in epilog** check box if you want to locate objects that have been created and deleted but where the object is not deleted in the epilog section of the method. This situation would result in a leak if an exception occurred during the method call.

By default, this check box is unchecked, indicating that epilogos are not scanned for possible leaks.

8. Click the **Search** button to initiate the search. Alternatively, click the **Cancel** button to abandon your selections.

The result of your search is output to a report that is displayed in print preview mode. This report contains the:

- Schema of the method
- Class in which the method is defined
- Method name where the create was performed
- Line number of the **create** statement in the method
- Type of the object being created
- Variable name used in the **create** statement
- Description of the possible leak

---

**Note** The report will also track calls to RootSchema methods that create and return a transient object that needs to be deleted (for example, **Class::allProperties**) and methods scanned that return a transient object that was created. If a possible leak is detected for such situations, the description will include the method name involved and the line number will be the statement in which the call was made.

---

## Locating Unused Local Variables and Parameters (NFS 58066)

You can now scan all methods in a schema or class for unused local variables and parameters. (In earlier releases, you could locate unused local variables for a specific method only.)

The Schema menu in the Schema Browser now contains the **Find Unused Local Variables/Parameters** command. When you select this command, the Find Unused Local Variables and Parameters dialog is then displayed.

1. If you do not want to search for unused local variables and parameters in all methods in all schemas (the default value), perform one of the following actions in the Search Criteria group box. Select the:
  - **Search selected schema** option button and in the combo box below the option button, select the schema in which you want to locate unused local variables and parameters in all methods.
  - **Search selected schema and all sub-schemas** option button and in the combo box above the option button, select the schema in which you want to locate unused local variables and parameters in all methods in that schema and its subschemas.
  - **Search selected class in selected schema** option button and in the combo box below the option button, select the class of the schema selected in the first combo box or schema that has a local copy class whose methods will be scanned for unused local variables and parameters.
2. In the Reporting Options group box, uncheck the:
  - **Local Variables** check box if you do not want to display unused local variables. This check box is checked, by default; that is, unused local variables are output to the report.
  - **Method Parameters** check box if you do not want to display unused method parameters. This check box is checked, by default; that is, unused method parameters are output to the report.

If you want to display unused method parameters in *all* methods in the schema instead of all methods *except* for event methods (the default), select the **All Methods** option button. The Parameters group box is visible only when the **Method Parameters** check box is checked.
3. Click the **Search** button to initiate the search. Alternatively, click the **Cancel** button to abandon your selections.

The result of your search is output to a report that is displayed in print preview mode. This report contains the:

- Schema of the method
- Class in which the method is defined
- Method name where the create was performed
- Line number in the method of the unused variable statement or parameter
- Variable name of the unused local variable or parameter
- Type of value; that is, **variable** or **parameter**

## Search and Replace Dialogs (NFS 38838, 58078)

The following changes have been made to the Global Search and Replace dialog, the local Find/Replace dialog, the XAML Browser Find dialog, and the translatable strings Search Strings dialog.

- Any displayed text in the **Search** combo box is initially selected.
- The **Search** text box has been changed to a combo box that contains a list of the text of the last 50 unique searches that were performed.

You can repeat previous searches by selecting an entry from the combo box list. This list is maintained in the user profile of your current user logon.

- The **Replace** text box has been changed to a combo box that contains a list of the text of the last 50 unique replacements that were performed.

You can repeat previous replacements by selecting an entry from the combo box list. This list is maintained in the user profile of your current user logon.

- The **Replace** combo box is always left blank, to prevent accidental replacements with text from previous replacements.
- The local Find/Replace dialog, the XAML Browser Find dialog, and the translatable strings Search Strings dialog now provide the **Full Identifier Only** check box, which enables you to specify that only text that is an identifier in which the whole word matches is located and replaced. If this check box is checked and the text is not an identifier, the locate and replace actions will fail.

As this check box is unchecked by default, any identifier text that matches the string that is being searched for will be located and replaced, regardless of whether it is embedded or not in another word.

- Any displayed text in the **Search** combo box is initially selected.
- The **Search** text box in earlier releases has been changed to a combo box that contains a list of the text of the last 50 unique searches that were performed.

You can repeat previous searches by selecting an entry from the combo box list. This list is maintained in the user profile of your current user logon.

- The **Replace** text box in earlier releases has been changed to a combo box that contains a list of the text of the last 50 unique replacements that were performed.

You can repeat previous replacements by selecting an entry from the combo box list. This list is maintained in the user profile of your current user logon.

- The **Replace** combo box is always left blank, to prevent accidental replacements with text from previous replacements.

## Unit Tests (NFS 58045)

When you press F9 in a Class Browser to run a JADE unit test, if a **JadeTestCase** class is selected and it has no unit test methods defined on it, all test methods for all subclasses of that class in that schema will now be run. (In earlier releases, the Unit Test Runner form started up, but nothing was run.)

## Disk Cache Memory (PAR 57865, 58354)

The default free memory target on a system that has more than 1G byte of physical memory has increased from 256M bytes to 512M bytes.

The default value of the **DiskCacheFreeMemoryTarget** parameter in the [PersistentDb] section of the JADE initialization file, which specifies the amount of physical memory the disk cache will try to keep free while honoring pool minimum and maximum segment settings, is now therefore **512M**.

Because the default value of 512M bytes could be larger than a quarter of physical memory on a virtual machine configured with 2G bytes, to support virtual machines (for prototyping, testing, and so on), the default value of the **DiskCacheFreeMemoryTarget** parameter for virtual machines is the minimum of 512M bytes or a quarter of physical memory. For production databases, the minimum physical memory that is supported is 2G bytes.

## Dynamic Properties

The dynamic properties feature is an efficient and easy way to add properties to existing instances without requiring a database reorganization. Dynamic properties can be added at run time. They are also convenient for situations when ad hoc values need to be attached to existing classes.

Dynamic properties are stored in separate storage units, called *dynamic property clusters*. A cluster is associated with the object where standard, or fixed, properties are stored. For better performance, dynamic properties that are typically accessed together can be assigned to specific clusters.

---

**Note** The dynamic properties feature is enabled only when you specify the **AllowDynamicProperties** parameter with a value of **true** in the [JadeExecuteFlags] section of the JADE initialization file.

---

## Benefits of Dynamic Properties

Dynamic properties can be added:

- By user applications to open classes
- To a class with persistent instances, without requiring a database reorganization

## Dynamic Property Classes, Properties, and Methods

The allowed dynamic property types are shared reference and any primitive embedded type.

Changes to the name, type, and length of dynamic properties are supported.

If the maximum length of a **Binary**, **String**, or **StringUtf8** dynamic property is changed and longer values have been assigned, the value is truncated to the defined length when the property is accessed. The new length will be enforced when a new value is assigned to the property.

## Restrictions

You cannot add:

- More than 255 dynamic property clusters to a class.
- A dynamic property to a system class.
- A dynamic property to a **Collection** class or subclass.

## Limitations in the Current Release

In this release, you cannot:

- Delete or change a dynamic property selected in the Properties List of the Class Browser.  
You can do this only from the **Property** class **changeDynamicProperty** method or the **JadeDynamicPropertyCluster** class **deleteDynamicProperty** method.
- Use dynamic properties by name in JADE methods. Exception 1068 (*Feature not available*) is raised if a dynamic property is accessed directly (by name) in a method.  
Access a dynamic property with the **Object** class **getPropertyValue** and **setPropertyValue** methods.
- Select a dynamic property from the list of properties in the following places.
  - Dictionary key
  - Key path of a dictionary
  - Properties on a class exported in a package
  - Java, C#, or Web service exposures
  - Design-time property in the JADE Painter
  - Relational view
  - RPS mapping
- Add a dynamic property cluster to a subclass of the **Control** class.
- Have dynamic property clusters on a class in a partitioned database file.
- Rename or delete a dynamic property by using a JCF command file in the batch Schema Load utility.
- Reduce the length of a **Binary**, **String**, or **StringUtf8** dynamic property that is longer than 540 bytes or characters.
- Delete a **Binary**, **String**, or **StringUtf8** dynamic property that is longer than 540 bytes or characters.
- Delete dynamic property clusters.

## Object::getPropertyValue Method Change

If the value of the **Object** class **getPropertyValue** method **name** parameter:

- Identifies a static property, this method returns its value.
- Identifies a dynamic property that has been instantiated (that is, a value has been assigned), it returns the value; otherwise it returns the null value for the property type.
- Is unknown, an exception is raised.

## Object::setPropertyValue Method Change

If the value of the **Object** class **setPropertyValue** method **name** parameter:

- Identifies a dynamic or a static property, this method sets its value for the current instance.
- Is unknown, an exception is raised.

## Object::tryGetPropertyValue Method

The **Object** class now provides the **tryGetPropertyValue** method, which has the following signature.

```
tryGetPropertyValue(name: String;  
                    instantiated: Boolean output): Any allowBrowserExecution;
```

If the value of the **name** parameter:

- Identifies a static property, the value of the **instantiated** parameter is set to **true** and the value of the property is returned.
- Identifies a dynamic property that has been instantiated (that is, a value has been assigned), the value of the **instantiated** parameter is set to **true** and the value of the property is returned.
- Identifies a dynamic property that has not been instantiated (that is, no value has been assigned or the value has been deleted), the value of the **instantiated** parameter is set to **false** and null is returned.
- Is unknown, returns **false** in the **instantiated** parameter.

## Object::deletePropertyValue Method

The **Object** class now provides the **deletePropertyValue** method, which has the following signature.

```
deletePropertyValue(name: String) allowBrowserExecution;
```

If the value of the **name** parameter:

- Identifies a static property, this method sets the property value to null.
- Identifies a dynamic property, this method removes the value of the specified dynamic property from its cluster, making the property un-instantiated for that object instance.
- Is unknown, an exception is raised.

## Class::addDynamicPropertyCluster Method

The **Class** class now provides the **addDynamicPropertyCluster** method, which has the following signature.

```
addDynamicPropertyCluster(name: String): JadeDynamicPropertyCluster;
```

This method adds a new dynamic property cluster with the name specified in the **name** parameter to the class of the receiver and returns a reference to the dynamic property cluster that was created.

The cluster name must be unique within the class and the superschema branch, and within all classes and subschemas. The cluster name, which must start with a lowercase letter, has a maximum length of 30 characters. It can include numbers and underscore characters, but it cannot include punctuation, spaces, or other non-alphanumeric characters.

## Class::addDynamicProperty Method

The **Class** class now provides the **addDynamicProperty** method, which has the following signature.

```
addDynamicProperty(clusterName: String;  
                   propertyName: String;  
                   propertyType: Type;  
                   length: Integer;  
                   scaleFactor: Byte): Property;
```

This method adds a new dynamic property with the name, type, length, and scale factor specified in the respective **propertyName**, **propertyType**, **length**, and **scaleFactor** parameters to the dynamic property cluster specified in the **clusterName** parameter and returns a reference to the dynamic property that was created.

The **length** parameter value is required for **Binary**, **String**, and **StringUtf8** primitive types. The **scaleFactor** parameter is required for **Decimal** primitive types.

The property name must be unique for both static and dynamic properties within the class and the superschema branch, and within all classes and subschemas. The property name, which must start with a lowercase letter, has a maximum length of 30 characters. It can include numbers and underscore characters, but it cannot include punctuation, spaces, or other non-alphanumeric characters.

The process must be in transaction state when a dynamic property definition is added. This transaction must be committed before the property can be used. After the transaction has been committed, your application should call the **Property** class **waitForDynamicProperty** method, to ensure that the property can be accessed.

## Class::*findDynamicPropertyCluster* Method

The **Class** class now provides the **findDynamicPropertyCluster** method, which has the following signature.

```
findDynamicPropertyCluster(name: String): JadeDynamicPropertyCluster;
```

This method returns a reference to the dynamic property cluster with the name specified in the **name** parameter from the class of the receiver or it returns null if no cluster with the specified name is defined in the class of the receiver.

## JadeDynamicPropertyCluster Class

JADE now provides the **JadeDynamicPropertyCluster** class, which is a subclass of the **Object** class.

The properties defined in the **JadeDynamicPropertyCluster** class are summarized in the following table.

Property	Type	Description
name	String[30]	Read-only name of the dynamic property cluster
properties	PropertyNDict	Collection of dynamic properties in the cluster
schemaType	Type	Type of the dynamic property cluster instance

The methods defined in the **JadeDynamicPropertyCluster** class are summarized in the following table.

Method	Description
addDynamicProperty	Adds the specified dynamic property to the cluster
addExclusiveDynamicProperty	Adds a new exclusive dynamic property with the specified name and type to the receiving cluster
deleteDynamicProperty	Deletes the specified dynamic property from the cluster
findDynamicProperty	Returns the specified dynamic property from the cluster of the receiver class

For details, see the following subsections.

## JadeDynamicPropertyCluster::*addDynamicProperty* Method

The **addDynamicProperty** method of the **JadeDynamicPropertyCluster** class adds a new dynamic property with the name, type, length, and scale factor specified in the respective **propertyName**, **propertyType**, **length**, and **scaleFactor** parameters to the dynamic property cluster of the receiver class and returns a reference to the dynamic property that was created.

The **length** parameter value is required for **Binary**, **String**, and **StringUtf8** primitive types. The **scaleFactor** parameter is required for **Decimal** primitive types.

The property name must be unique for both static and dynamic properties within the class and the superschema branch, and within all classes and subschemas. The property name, which must start with a lowercase letter, has a maximum length of 30 characters. It can include numbers and underscore characters, but it cannot include punctuation, spaces, or other non-alphanumeric characters.

This method has the following signature.

```
addDynamicProperty(propertyName: String;
                  propertyType: Type;
                  length: Integer;
                  scaleFactor: Byte): Property;
```

### **JadeDynamicPropertyCluster::addExclusiveDynamicProperty Method**

The **addExclusiveDynamicProperty** method of the **JadeDynamicPropertyCluster** class adds a new exclusive dynamic property with the name and type specified in the respective **propertyName** and **propertyType** parameters to the receiving dynamic property cluster, and returns a reference to the exclusive dynamic property that was created.

The type must be a subclass of the **Collection** class.

This method has the following signature.

```
addExclusiveDynamicProperty(propertyName: String;
                           propertyType: Type): Property;
```

### **JadeDynamicPropertyCluster::deleteDynamicProperty Method**

The **deleteDynamicProperty** method of the **JadeDynamicPropertyCluster** class deletes the dynamic property with the name specified in the **propertyName** parameter from the receiving cluster.

An exception is raised if the dynamic property is not defined in the cluster.

This method has the following signature.

```
deleteDynamicProperty(propertyName: String);
```

### **JadeDynamicPropertyCluster::findDynamicProperty Method**

The **findDynamicProperty** method of the **JadeDynamicPropertyCluster** class returns a reference to the dynamic property with the name specified in the **name** parameter from the receiving cluster or it returns null if the specified dynamic property is not defined in the cluster.

This method has the following signature.

```
findDynamicProperty(name: String): Property;
```

## **Property Class Methods**

The abstract **Property** class now provides the methods documented in the following subsections.

### **Property::addDynamicInverse Method**

The **Property** class **addDynamicInverse** method, which has the following signature, adds an inverse between the receiving dynamic property and the dynamic property specified in the **property** parameter.

```
addDynamicInverse(property: Property;
                 mode: Character;
                 kind: Character): Property subschemaFinal, updating, abstract;
```

This method returns a reference to the receiving dynamic property with the inverse added. Note that when the first inverse is added, this will be different from the receiver.



The **property** parameter must identify a compatible dynamic property; for example, if the type of the property is a **Collection** class, the schema type of the receiver must be compatible with the membership of the collection.

---

**Note** If instances of the class of either dynamic property exist and the dynamic properties have non-null values, the inverse relationship is not validated or populated when the inverse definition is added.

---

The values for the **mode** parameter are defined by the following **ExplicitInverseRef** class constants.

- UpdateMode\_Automatic
- UpdateMode\_ManAuto
- UpdateMode\_Manual

The values for the **kind** parameter are defined by the following **ExplicitInverseRef** class constants.

- Kind\_Child
- Kind\_Parent
- Kind\_Peer

### **Property::changeDynamicProperty Method**

The **Property** class **changeDynamicProperty** method, which has the following signature, changes an existing dynamic property.

```
changeDynamicProperty(propertyName: String;
                      propertyType: Type;
                      length: Integer;
                      scaleFactor: Byte) updating;
```

You can change any or all of the **propertyName**, **propertyType**, **length**, and **scaleFactor** parameter values. The **length** parameter is required for **Binary**, **String**, and **StringUtf8** primitive types. The **scaleFactor** parameter is required for **Decimal** primitive types.

The type of a primitive property cannot be changed to a reference, and the reverse.

No reorganization is required when you change the type of a dynamic property. If the actual value of a dynamic property differs from the type of the property definition, the value is converted when the value is retrieved. Dynamic property values are always stored using the type of the property definition when the value is updated.

---

**Note** If the length of a **String** dynamic property is decreased, any populated values that are longer than the new length are truncated when the property is fetched.

---

### **Property::changeExclusiveDynamicProperty Method**

The **Property** class **changeExclusiveDynamicProperty** method, which has the following signature, changes the name of the existing exclusive dynamic property specified in the **propertyName** parameter.

```
changeExclusiveDynamicProperty(propertyName: String) updating, abstract,
                             subschemaFinal;
```

You can change only the name of an existing exclusive dynamic property.

### **Property::removeDynamicInverse Method**

The **Property** class **removeDynamicInverse** method, which has the following signature, removes an inverse reference between the dynamic property specified in the **property** parameter and the receiver.

```
removeDynamicInverse(property: Property): Property subschemaFinal,
                    updating, abstract;
```

This method returns a reference to the receiving dynamic property with the inverse removed. Note that when the last inverse is removed, this will be different from the receiver.

## Example Usages

The code fragment in the following example adds a dynamic property.

```
vars
    cluster : JadeDynamicPropertyCluster;
begin
    beginTransaction;
    // create a cluster
    cluster := Cl.addDynamicPropertyCluster("aCluster");
    // add a property to the cluster
    cluster.addDynamicProperty("anInteger", Integer, 0, 0.Byte);
    commitTransaction;
```

The code fragment in the following example adds a dynamic property and sets the value.

```
vars
    cluster : JadeDynamicPropertyCluster;
    property : Property;
    cl      : Cl;
begin
    beginTransaction;
    // create a cluster
    cluster := Cl.addDynamicPropertyCluster("aCluster");
    // add a property to the cluster
    property := cluster.addDynamicProperty("anInteger", Integer, 0, 0.Byte);
    commitTransaction;
    create cl transient;
    // set the property on an instance
    cl.setPropertyValue("anInteger", 123);
    ....
```

The code fragment in the following example changes a dynamic property and sets the value.

```
vars
    cluster : JadeDynamicPropertyCluster;
    property : Property;
    cl      : Cl;
begin
    // find the cluster
    cluster := Cl.findDynamicPropertyCluster("aCluster");
    // find the property to the cluster
    property := cluster.findDynamicProperty("anInteger");
    beginTransaction;
    property.changeDynamicProperty("anInteger", Integer64, 0, 0.Byte);
    commitTransaction;
    cl := Cl.firstInstance();
    // retrieve the current (Integer) value (as Integer64)
    write cl.getPropertyValue("anInteger");
    // set the value
    cl.setPropertyValue("anInteger", 456);
    ....
```

## Exception Handling (NFS 52373)

When a process raises an exception that gets back to the JADE unhandled exception handler, the call stack is logged in the application log.

The unhandled exception dialog and the exception log heading line now display the process instance identifier, the system process identifier, and the thread identifier in addition to the application name and computer name. The unhandled exception dialog title now contains the following information.

```
by [process-oid], pid process-identifier, tid thread-identifier
```

The following are examples of log entries.

```
Unhandled SystemException on 2012/05/14 14:10:03 by [187.3] pid 00ec0, tid 1a80
```

```
Unhandled Lock Exception on 2012/05/14 14:33:59 by [187.7] pid 01950, tid 1848
```

## Installed Thin Client Download Files

Installed thin client download files now have the **architecture – character-set\download\bin** structure; for example:

```
i686-msoft-win32_vs2005-ansi\download\bin
```

## JADE Initialization File (PAR 57781)

The default value of the **ResourceErrorRetryLimit** parameter in the [PersistentDb] section of the JADE initialization file was documented as zero (0), which signified infinite retries. However, the default value is 2; that is, there are two retries after the initial failure and a value of zero (0) indicates that there are no retries.

## Logical Certifier (PAR 57908)

The Jade Logical Certifier dialog now includes the **Display Progress** check box, which controls whether progress messages are output to the Jade Interpreter Output Viewer window. (This check box is checked by default.)

You can control the frequency of these messages by specifying the number of instances in the **Instances** text box. By default, messages are output every 10,000 instances.

For the non-GUI **certify** operation only or when you selected **Certify All Schemas** in the **Operation** combo box, all classes can be processed in parallel by controlling the number of worker processes available, in the **Workers** text box or the optional **workers** command line parameter, respectively. By default, a single worker process is used. When more than one worker is specified, the elapsed time to certify a system *may* be reduced.

---

**Note** The number of worker processes will depend on a multitude of different factors, including things such as available I/O bandwidth, CPU cores, the number of instances, and the number of inverses in each class. As a guide, the number of workers should not exceed the number of CPU cores minus one.

---

The JADE Logical Certifier non-GUI application now provides the following optional command line parameters.

- progress=true|false (the default value is **true**)
- instances=10000 (the default value)
- workers=1 (the default value, which applies only to the **certify** operation when no **\_logcert.in** input file is used)

## Monitor Display of the Node Edition (NFS 56397)

The **Node Statistics**, **Cache Performance**, and **Users** views in the JADE Monitor now display the edition (that is, 64-bit or 32-bit) of each node entry.

## Multiple Database Access for .NET

The JADE .NET API now provides support for multiple database access. This feature allows a single .NET application (operating system process) to access a number of JADE databases concurrently.

### Restrictions

The restrictions for the multiple database support feature are as follows.

- The JADE databases must be at the same feature level.
- Multiple connections to the same database are not allowed.
- There is a limit of 20 concurrent connections.
- Some JADE initialization file parameters (for example, the **LogDirectory** parameter in the [JadeLog] section) are read only when the first database connection is opened, and they remain unchanged for the life of the process.

---

**Tip** Use a common JADE initialization file for all database connections, to ensure that the same **jommsg.log** file is used, regardless of the order in which database connections are established.

---

## Using the Multiple Database Access Support Feature

To use the multiple database access support feature, you must supply connection details via the appropriate **JobContext** constructor. This can take the form of an existing **JobConnection** object, or it can be via a **JobConnectionStringBuilder** object, as shown in the following example.

```
<BLOCK 1A>
using (JobContext context = new JobContext
    (JobConnectionStringBuilder.CreateFromConfig("LocalJob")))
{
    // Established context to database with schema MultipleDb1 as
    // specified by the "LocalJob" connection string in the // app.config
    <BLOCK 2A>

    using (JobContext remoteContext = new JobContext
        (JobConnectionStringBuilder.CreateFromConfig("RemoteJob")))
    {
        // Established context to database with schema MultipleDb2 as
        // specified by "RemoteJob" connection string.
        <BLOCK 3>
    }
    <BLOCK 2B>
}
<BLOCK 1B>
```

The **app.config** file reads as follows.

```
<connectionStrings>
<add name="RemoteJob" providerName="JadeSoftware.Job.JobConnection"
    connectionString="DataSource=tcpip://remoteHost:6005/7bf0f9cd-680a-
e111-9eaf-5ae520524153;ConfigFile=\\remoteHost\C:\ProgramData\
Jade Software\JobServer\system\Job.ini;
    Schema=MultipleDb2;IntegratedSecurity=True"></add>
<add name="LocalJob" providerName="JadeSoftware.Job.JobConnection"
    connectionString="DataSource=C:\ProgramData\Jade Software\
    JobServer\system;ConfigFile=C:\ProgramData\Jade Software\
    JobServer\system\Job.ini;Schema=MultipleDb1;
```

```

    IntegratedSecurity=True"></add>
</connectionStrings>

```

The model for multiple database access follows the pattern of requiring a live **JoobContext** before a **JoobObject** can interact with its database.

When an appropriate **JoobContext** has been instantiated for a database, **JoobObject** instances from that database can be retrieved, created, and de-referenced. Each **JoobObject** instance knows the database to which it belongs and provided that a **JoobContext** is alive for its database on the current thread, **JoobObject** manipulation can occur.

For example, a **JoobObject** variable of a **MultipleDb2** class can be declared in *BLOCK 1A* or *BLOCK 2A*. You can set, get, or invoke JADE schema-defined properties and methods only from within *BLOCK 3*. You can locate instances of classes only from schema **MultipleDb2** within *BLOCK 3*. However, you can manipulate instances of classes from **MultipleDb1** within *BLOCK 2A*, *BLOCK 2B*, and *BLOCK 3*.

## Supporting Features

The multiple database access supporting features are documented in the following subsections.

### Database Identity

JADE databases now have environment identity and the server identity UUID values. These uniquely identify a database and they form part of a server URI that can be used to define a connection to a specific database.

For details, see "[Database Identities](#)", earlier in this document.

### Connection String Extensions

You can now specify the intended target database in connection strings in one of two ways.

- The existing action; that is, the database control file path and the **ServerNodeSpecifications** parameter value in the [JadeClient] or [ConnectionParams] section of the JADE initialization file.
- Specifying the server URI, which has the following general format.

```

scheme://host-name:port-number or base-name/environment-UUID
[/server-UUID] [?parameters]

```

The *scheme* value specifies the transport type, which can be one of the following values.

- Tcplp
- Tcplpv4
- Tcplpv6
- HPSM
- JadeLocal

The "*host-name:port-number or base-name*" value specifies the target address of the server.

The *environment-UUID* value specifies the database, which is expected to be at the target address. This is optionally followed by the database server identity UUID.

The optional *parameters* values enable you to specify the local address or port for TCP/IP transports.

For details, see "[Format of the Server Uniform Resource Identifier \(URI\) String](#)" under "[Database Identities](#)", earlier in this document.

---

**Tip** Connection establishment is fastest when your `connectionString` uses a server URI with both the environment and server UUIDs specified.

---

The following are examples of `connectionString DataSource` values using:

- TCP/IP version 4 implicitly

```
"DataSource=TcpIp://localhost:6005/48cf13df-bf6d-df11-87e2-2e5925024153;..."
```

- TCP/IP version 4, explicitly declaring an explicit local address

```
"DataSource=TcpIpv4://host.company.com:6005/48cf13df-bf6d-df11-87e2-2e5925024153?localHostname=aHostName;"
```

- TCP/IP version 6, declaring an explicit local port

```
"DataSource=TcpIpv6://[fe81::6fe:7fff:fe97:bd20]:6005/48cf13df-bf6d-df11-87e2-2e5925024153?localPort=54321;..."
```

- HPSM, declaring both environment and server UUIDs

```
"DataSource=HPSM://localhost:SRCJoob-HPSM/48cf13df-bf6d-df11-87e2-2e5925024153/48cf13df-bf6d-df11-87e2-2e5925024153;..."
```

## JADE Assembly Public Keys

JADE assemblies are signed. Each release uses the same public key token for its assemblies, but they have different versions, as shown in the following table.

Build/Configuration	PublicKeyToken
Release_Ansi x64	4ffc2b9eb5630e47
Release_Unicode x64	b5033a0291fb93d9

An example of the `app.config` file is as follows.

```
<configSections>
  <section name="joob"
    type="JadeSoftware.Joob.Configuration.JoobConfigurationSection,
    JadeSoftware.Joob, Version=7.0.5.0, Culture=neutral,
    PublicKeyToken=4ffc2b9eb5630e47">
  </section>
</configSections>
```

An application run with this example `app.config` file requires `JadeSoftware.Joob.dll` with an assembly version of 7.0.5.0 and signed with the `Release_AnsiX64` key.

Hot fix versions of `JadeSoftware.Joob.dll` (and so on) are released with the same assembly version as the initial consolidated release (for example, 7.0.5.0) but the file version will indicate the hot fix number. Your assemblies, therefore, do not need to be rebuilt when a hot fix is installed but they must be rebuilt when a subsequent consolidated release is installed.

## SDS Exception (PAR 57818)

JADE can now raise exception 3220 - *Operation not permitted when primary is connected*, which is reported when a hostile takeover is initiated at a secondary while it is connected to the primary. If this exception is raised, shut down the connection between the secondary and the primary.

## Secure Sockets Layer (SSL) Protocol

The following subsections contain the Secure Sockets Layer (SSL) changes in this release.

## OpenSSL Version Upgrade (PAR 58040)

The OpenSSL library has been upgraded to version 1.0.1 from version 1.0.0d (in JADE 7.0) and version 0.9.8e (in JADE 6.3).

In this 1.0.1 release, internal defaults in OpenSSL have changed, making the default list of cipher names more restrictive. If you want to use the sample-only certificates provided with JADE, which are intended for testing purposes only and *not* for production use, you must make additional changes to the JADE initialization file, to enable an appropriate list of cipher names because the default cipher list does not work with the sample-only certificates.

In the [JadeAppServer] and [JadeThinClient] sections of the JADE initialization file, the following value of the **SSLCipherNames** parameter works with the sample certificates.

```
SSLCipherNames=AES:ALL:!aNULL:!eNULL:+aECDH:+kRSA:+RC4:@STRENGTH
```

## Secure Socket Layer Security (PAR 58011)

The [JadeAppServer] section of the JADE initialization file can now contain the boolean **SSLPermitClientRenegotiation** parameter, which defines whether client-initiated renegotiation is allowed. The default value of **true** retains the behavior of earlier releases.

Payment Card Industry (PCI) compliance checks can require that measures are present to prevent vulnerability to CVE-2009-3555-based attacks.

If you want PCI compliance or to protect against potential Denial of Service (DoS) attack, set this parameter to **false** so that any client-initiated renegotiation will cause the network connection to be dropped.

The **SSLPermitClientRenegotiation** parameter is read once, on the first SSL connection.

An additional log message has been added to log the build version of OpenSSL and the version of the OpenSSL library DLLs used (that is, **sseay32.dll** and **libeay32.dll**).

---

**Note** To support secure client renegotiations, you require a minimum version of 0.9.8l of the OpenSSL libraries.

---

## Silverlight Stateless (NFS 57307)

To generically handle the **XamlDocument** class **registerEventHandler** method for controls defined in a **DataTemplate**, the generation of a Silverlight Stateless application has been changed as follows.

- When a document is saved, any third-party controls are investigated to determine where they fit in the JADE **XamlObject** hierarchy. In earlier releases, these controls were generated as a **XamlObject**-type property.

From this release, when the document is saved, the assemblies for that control are referenced to determine whether the control inherits from a **JadeXamlControl** type; for example, the **RadComboBox** in the Telerik Controls assembly inherits from **XamlItemsControl**. As a result, the property associated with that control would be created as a **XamlItemsControl**.

This also means that the control can be referenced via the property, as that type and the events associated with that control type can be directly defined rather than having to call the **XamlDocument** class **registerEventHandler** method.

---

**Note** This change means that any assemblies required must be located in the directory associated with the **DefaultAssemblyLocation** parameter in the [JadeSilverlight] section of the JADE initialization file. If the required assemblies cannot be found, a message box advises you of the failure and asks if you want to display the details in an exception dialog. The exception dialog lists the assembly that could not be found.

---

- In earlier releases, when a named item in the XAML was used in a **DataTemplate**, you could not define events for that item in JADE and a property reference was not generated.

From this release, a named item in a **DataTemplate** will be generated as a virtual property of the defined type, which enables you to directly define events in JADE. However, because it is virtual, any logic references will be rejected by the JADE compiler. This is deliberate, because at run time, the template can be used multiple times to generate different instances of the control that all have the same name. When an event occurs on one of these controls, the first parameter of the event is the control instance, which can then be referenced in logic.

To get this change to take effect, you must re-save your user-defined **XamlDocuments**.

Clicking on one of these virtual properties in JADE will display the following text.

Note: This property is defined inside a Xaml data template and CANNOT be directly accessed at run time from logic. However, events can be defined that can be called in the JADE Silverlight Stateless version. Those events will not be called in the JADE Thin Client Silverlight version.

Attempted use of this property in logic will generate a compile error: Error 6187: Cannot access virtual property with no mapping method.

This is deliberate and used to prevent invalid logic being written. Do not add a mapping method as that will cause Silverlight Xap generation failure.

- For events to be received for named **DataTemplate** items, the XAP generation process now modifies the XAML to insert the event method references and generates sufficient code behind the XAML to implement the events in the JADE stateless model. (This change is transparent to you.)

For third-party controls used that are named and can be defined as a **XamlUIElement** subclass, you can define events not available in JADE by performing the following actions.

1. Define an event handler method for the event.

In the following example, the source type matches the type of the control property.

```
radCombo_selectionChanged(source: XamlItemsControl input;
    origin: XamlObject input);
```

2. Define a loaded event on the control in JADE that does the following (where the source type matches the type of the control property).

```
radCombo_Loaded(control: XamlItemsControl input;
    originalSource: XamlObject input);
begin
    if control.tag <> "Done" then // The loaded event for a control can be
        // called multiple times if the control is
        // hidden and then made visible, for example.
        self.registerEventHandler(source, "SelectionChanged",
            xaml-document-name::radCombo_selectionChanged);
        control.tag := "Done";
    endif;
end;
```

This logic will then register that event for every instance of the **DataTemplate** item that is created.

## System::getDbDiskCacheStats Method

With the deimplementation of the **System::getDbObjectCacheStats** method in JADE 7.0 (because the database engine no longer utilizes an object cache), the **System** class now provides the **getDbDiskCacheStats** method, which has the following signature.

```
getDbDiskCacheStats(jdo: JadeDynamicObject input);
```

The **getDbDiskCacheStats** method returns statistics relating to the persistent database cache. The values are returned as **Integer64** properties in the dynamic object specified by the **jdo** parameter.



The returned values are **Integer64** values representing counts of actions pertaining to the persistent database cache, most of which are cumulative, starting from when the database was opened. If your applications use this method, you therefore need to compare values from one call to the next, to work out the value differences.

The calling process is responsible for creating and deleting the **JadeDynamicObject** instance. Properties are added to the object when the method is first called. The object can then be used in subsequent calls.

If the dynamic object passed to the method already contains properties that do not match the properties to be returned, the existing dynamic object properties are removed and replaced with the appropriate properties. This method is most efficient when the properties match those to be returned.

The cumulative values are held as 64-bit unsigned integer values, and are copied to the dynamic object as **Integer64** values. The maximum value before they wrap around to negative values is therefore **2<sup>63</sup> - 1** (approximately 8 Exabytes).

The method in the following example caters for a cumulative **Integer64** value overflowing to a negative value when calculating differences.

```
getDelta(oldValue, newValue: Integer64): Integer64;
begin
  if newValue >= oldValue then
    return newValue - oldValue;
  else
    //here if it wrapped to a negative value. Need to add 1 because
    //of the way 2s complement works. Parentheses are needed to avoid
    //truncation errors.
    return -((Max_Integer64 - oldValue) + 1 + (newValue -
      Min_Integer64));
  endif;
end;
```

The following example shows the use of the **getDbDiskCacheStats** method.

```
showDbDiskCacheStats();
vars
  jdo : JadeDynamicObject;
begin
  create jdo transient;
  system.getDbDiskCacheStats(jdo);
  write jdo.display;
epilog
  delete jdo;
end;
```

The output from the **showDbDiskCacheStats** method shown in the previous example is as follows.

```
---DatabaseDiskCacheStatistics(210)---
cacheMisses = 21
gets = 8162
puts = 331
blockReads = 8
getsWithFetch = 0
putsWithFetch = 0
blocksFetched = 1258
blockReadsMultiple = 333
bufferReassigns = 0
bufferSteals = 0
maxHashCollisions = 0
maxConcFlushIos = 15
blockWrites = 4
blockWritesMultiple = 111
```

The properties returned in the dynamic object are listed in the following table.

Property	Description
cacheMisses	The number of cache misses; that is, where the address referenced a block that was not found in the cache
gets	The number of read requests received
puts	The number of write requests received
blockReads	The number of block reads from disk performed
getsWithFetch	The number of read requests with additional block prefetch received
putsWithFetch	The number of write requests with additional block prefetch received
blocksFetched	The number of blocks prefetched with read from disk
blockReadsMultiple	The number of prefetch reads performed
bufferReassigns	The number of times buffers were reassigned for use with a different address
bufferSteals	The number of times the buffer reassigned was dirty, requiring writing
maxHashCollisions	The maximum number of hash collisions
maxConcFlushlos	The maximum number of concurrent flush write operations
blockWrites	The number of single block writes to disk performed
blockWritesMultiple	The number of multiple block writes to disk performed

For more details about these values, see the JADE Monitor knowledge base.

Properties are added to the object when the method is first called. The object can then be used in subsequent calls. If the dynamic object passed to the method already contains properties that do not match the properties to be returned, the existing dynamic object properties are removed and replaced with the appropriate properties. This method is most efficient when the properties match those to be returned.

## Thin Client Errors (PAR 58279)

The following errors can now occur when you are running JADE in thin client mode.

- 14118 - The start of a second application by a thin client did not end up on the correct application server

This error occurs when a presentation client has initiated a second application and the TCP connection was made to an application server that differs from the one running the first application. The network configuration is not set up correctly for JADE operation. When the first connection succeeds, the application server's IP address is queried. This IP address is subsequently used to ensure that the correct application server is reached if another application is initiated. Use of this IP address did not achieve the expected result and another application server was reached instead.

Adjust the network configuration accordingly.

- 14119 - A thin client reconnection attempt ended up on the wrong application server

A presentation client network connection was lost and an attempted reconnection to the original application server reached a different application server instead. The network configuration is not set up correctly for JADE operation. When the first connection succeeds, the application server's IP address is queried. This IP address is subsequently used to ensure that the correct application server is reached if the connection is lost and an attempted reconnect is initiated. Use of this IP address did not achieve the expected result and another application server was reached instead.

Adjust the network configuration accordingly.

## Windows Ghosting (PAR 57699)

When JADE running under Windows 8, Windows 7, or Vista became non-responsive in earlier releases, the title bar dropped the form skin.

The JADE executable now calls the **DisableProcessWindowsGhosting()** Microsoft API on initiation, which disables Windows' ghosting. As a result, a non-responsive form no longer shows *Not Responding*, nor does it have the ghosting effect applied by Windows.

The form will still not automatically paint itself when the presentation thread is busy processing JADE logic. This will be an issue only if the user covers the form with another window and then uncovers it again on Windows XP or Vista. Under Windows 8 or Windows 7, Windows automatically redraws that part of the form or forms that need refreshing from a saved copy of the previously painted image or images.

JADE has implemented the following methods that you can use to call while performing a long processing loop to address the repainting issue.

- **Application::paintIfRequired;**

The **paintIfRequired** method:

- Causes all forms of the application to be repainted if a **paint** event is required.

A **refreshNow** event is performed on that part of the form that needed refreshing. If **paint** events are not required, no action is performed.

- Performs any repainting required without having to perform an **app.doWindowEvents** method call, and therefore does not allow the user interface to be active.

Other than any **paint** events, no other events, notifications, or timer events will be processed as a result of this **paintIfRequired** method call.

- **Form::paintIfRequired;**

The **Form** class **paintIfRequired** method performs the same action as the **Application** class **paintIfRequired** method except that only the form on which the call is made will be affected.

---

**Note** After a repaint, any clicked button that initiated the processing loop will be drawn in the up position, so it will be important that the user is given a visual indication that the processing is still progressing by some other means; for example, by using the **app.mousePointer := 11** (busy) property value.

---

You will need to add a call to your logic loop that is regularly performed; for example, call it when the **Cancel** button is checked for a **click** event, when a progress bar update ticks over a percentage, or at a specified number of seconds, as shown in the following code fragment.

```
cancelled := false;
  while not cancelled do
    ... logic
    // the click event sets the cancelled property
    btnCancel.doWindowEvents(0);
    app.paintIfRequired();
  endwhile;
```

## Changes in JADE Release 7.0.04

This section contains details about changes in JADE release 7.0.04.

For a summary of changes and new features in JADE 7.0.03 (the first general release of JADE 7.0), see **RelInfo7003.pdf**, in your JADE **documentation** directory.

### Behavioral Change of Web Services (PAR 56921)

When using a Web service consumer or the **JadeHTTPConnection** class, the default communications protocol has changed from using the WinINet (the JADE 6.3.05 and 7.0.03 default value) to the WinHTTP library. WinHTTP is more appropriate for server-type environments; WinINet is more appropriate for low-performance client situations.

---

**Caution** A potential impact of this change is that if proxy servers are used, configuring the Web service consumer to use a proxy externally from JADE code will change.

For WinINet, you can use the Internet Options dialog accessed from the Control Panel. WinHTTP proxy settings are configured using the **netsh** tool. To obtain help, specify the following command.

```
netsh winhttp set proxy help
```

---

Use the **EnableWinHTTP** and **EnableWinINet** parameters in the [JadeEnvironment] section of the JADE initialization file to control the communications library that is used. The default values are now:

```
[JadeEnvironment]
EnableWinHTTP = true
EnableWinINet = false
```

If you set both parameters to **true**, WinHTTP is used in preference.

If you set both parameters to **false**, the support of both protocols is disabled and a Web service consumer can use only the JADE Direct scheme; that is, **jadehttp.tcp**. In JADE 6.3.09 and 7.0.03, an exception was raised resulting from the Windows ERROR\_RESOURCE\_DISABLED error (4309).

### Certificate Authentication (PARs 56904, 57638)

To ensure that the verification process correctly handles the Certificate Authority (CA) path and JADE initialization file parameters, the *Depth Zero Self Signed Cert* and *Invalid Purpose X.509* non-fatal errors are ignored.

In addition, the default values of the **SSLRemoteCertCheck** and **SSLRemoteVerify** parameters in the [JadeAppServer] and [JadeThinClient] sections of the JADE initialization file parameters are now as follows.

```
[JadeAppServer]
SSLRemoteCertCheck=false
SSLRemoteVerify=false
SSLVerifyDepth=9

[JadeThinClient]
SSLRemoteCertCheck=false
SSLRemoteVerify=false
SSLVerifyDepth=9
```

---

**Note** As the verify depth (specified in the **SSLVerifyDepth** parameter) is now honored, you should not set the value of this parameter to zero (0).

The following example of these values allows authority certificates to work.

```
[JadeAppServer]
SSLRemoteCertCheck=false
```

```

SSLRemoteVerify=true

[JadeThinClient]
SSLRemoteCertCheck=true
SSLRemoteVerify=true
;SSLVerifyDepth=0

```

## Converting a User Database (PAR 56876)

When converting a user database, the **JadeConvertDb** application converts up to 1,000 files.

If more than 1,000 files are being converted, an error message is output and the application will not convert any files.

## Date Parsing Routines for Two-Digit Years (PAR 56832)

The **JadeEditMask** class now uses the Windows Control Panel setting to convert a two-digit year into a four-digit year for a two-digit edit mask year of 'yy' when the value of the **EnhancedLocaleSupport** parameter in the [JadeEnvironment] section of the JADE initialization file is set to **true**.

By default, years:

- 00 through 29 become 2000 through 2029
- 30 through 99 become 1930 through 1999

If the value of the **EnhancedLocaleSupport** parameter is **false** (the default), the year is calculated using the current century.

## Deleting and Renaming Entities

The following subsections contain the commands that you can now include in a JADE command **.jcf** text file that is specified in the **jadloadb** batch JADE Load utility **commandFile** parameter.

### Deleting Packages (NFS 45928)

The **jadloadb** batch JADE Load utility **commandFile** parameter now enables you to delete both imported and exported packages from a schema, by using the following command syntax.

```
Delete Package schema-name::package-name
```

The following **jadloadb** example contains the fully qualified name of a JADE command **.jcf** text file that deletes a package.

```
jadloadb path=d:\jade\system commandFile=d:\temp\DeletePackages.jcf
ini=d:\jade\myjade.ini loadStyle=currentSchemaVersion
```

The **DeletePackages** command file in the previous example deletes imported package **P99** from the **Par99999Importer** schema, as follows.

```
JadeCommandFile
JadeVersionNumber 7.0.04
Commands
AbortOnError True
Delete Package Par99999Importer::P99
```

An exported package cannot be deleted if it is imported by another schema. An imported package cannot be deleted if it imports a class or interface that is used as the type of a property.

## Deleting a Schema (NFS 57540)

The **jadloadb** batch JADE Load utility **commandFile** parameter now enables you to delete a specified schema, by using the following command syntax.

```
Delete Schema schema-name
```

The following **jadloadb** example contains the fully qualified name of a JADE command **.jcf** text file that deletes a schema.

```
jadloadb path=d:\jade\system commandFile=d:\temp\DeleteSchema.jcf  
ini=d:\jade\myjade.ini loadStyle=currentSchemaVersion
```

The **DeleteSchema** command file in the previous example deletes schema **Par9999Importer**, as follows.

```
JadeCommandFile  
JadeVersionNumber 7.0.04  
Commands  
AbortOnError True  
Delete Schema Par9999Importer
```

A schema cannot be deleted if it has subschemas, it is versioned, or it exports a package that is imported by another schema.

## Renaming a Schema (NFS 57075)

The **jadloadb** batch JADE Load utility **commandFile** parameter now enables you to rename an existing schema, by using the following syntax.

```
Rename Schema existing-schema-name new-schema-name
```

The following **jadloadb** example contains the fully qualified name of a JADE command **.jcf** text file that renames a schema.

```
jadloadb path=d:\jade\system commandFile=d:\temp\RenameSchema.jcf  
ini=d:\jade\myjade.ini loadStyle=latestSchemaVersion
```

The **RenameSchema** command file in the previous example renames the **ExampleSchema** to **TestSchema**, as follows.

```
JadeCommandFile  
JadeVersionNumber 7.0.04  
Commands  
AbortOnError True  
Rename Schema ExampleSchema TestSchema
```

You cannot rename the current version of a schema.

## Dock Controls (PAR 56859)

The drawing of dock control drag rectangles was changed from drawing on the desktop because such drawing under Windows 8 or Windows 7 was slow and problematic (Windows can erase the drawing without warning).

If a dock container and the MDI client window share the same parent and overlap, when drawing on the dock container, the MDI client window causes that drawing to clip out its own window area.

JADE now recognizes the situation where the drawing is over a sibling MDI client window and it reverts to drawing on the desktop in that situation. This may then result in the drawing flickering and leaving drawing remnants, because of Windows interference.

## Initiating an Application (PAR 57541)

A call to the **Application** class **setApplicationSkin** or **setSkin** (the old style of skin) method from the **Global** class **getAndValidateUser** method now results in the initiating application being told that the new application has been successfully initiated.

In earlier releases, this occurred only when the **signOn** process completed or when the **getAndValidateUser** method created a form.

## Iterating Backwards through a Virtual Collection (PAR 57322)

The JADE compiler now allows the **reversed** option in the **foreach** instruction to be used in virtual collections.

## JADE Unit Testing Framework

The following subsections contain changes to the JADE unit testing framework.

### Debugging the Execution of a JADE Unit Test (NFS 52101)

You can now debug the execution of a JADE unit test.

When you select the **JadeTestCase** class in the Class Browser, the Jade menu now includes the **Unit Test Debug** command, which initiates the JADE unit test framework in JADE debug mode for the selected **JadeTestCase** class or selected method of the class if a method is selected.

In addition, the Jade menu in the Schema Browser contains the **Unit Test Debug** command if the schema contains a subclass of the **JadeTestCase** class. When you select this command from the Schema Browser, the JADE unit test framework is initiated in JADE debug mode for all **JadeTestCase** class subclasses.

---

**Tip** You can use the SHIFT+F9 shortcut instead of the **Unit Test Debug** command in the Jade menu.

---

### Initializing and Finalizing the JADE Unit Test Framework (PAR 56498)

When you run the JADE unit testing framework by using the F9 shortcut key to run the **JadeUnitTest** application from the JADE development environment or by running the **JadeUnitTestBatch**, **JadeUnitTestRun**, or **JadeUnitTestRunner** non-GUI client application using the **jadclient** executable, your **Application::finalize** method was called when the application finished. However, the **Application::initialize** method was not called.

It was inconsistent to execute the **finalize** method and not the **initialize** method, so from this release, neither the **finalize** nor the **initialize** method is called when a JADE unit test framework is used.

If your tests must initialize and finalize the application, they should do this explicitly, by adding a **unitTestBeforeClass** method and **unitTestAfterClass** method, which in turn could call the **app.initialize** and **app.finalize** methods, if appropriate, as shown in the following examples.

```
initialize() unitTestBeforeClass, updating;
begin
    app.initialize;
    // ... any other initialization
end;

finalize() unitTestAfterClass, updating;
begin
    app.finalize;
    // ... any other finalization
end;
```

## JadeLocal Transport (PAR 57602)

The JADE initialization file parameters required to use the **JadeLocal** transport, documented in the "JadeLocal Transport" section in Chapter 3 of the *JADE Installation and Configuration Guide*, stated that **JoobLocal** was required for both the server and client specifications.

The parameter should be **JadeLocal**; that is:

```
[JadeServer]
NetworkSpecification<n>=JadeLocal,enabled|disabled,[Global\]base-name

[JadeClient]
ServerNodeSpecifications=JadeLocal,[Global\]base-name
```

## JadeWebServiceProvider::getServerVariable (PAR 57163)

The Web service provider requires the name of the method to invoke. In order to obtain this, the **getServerVariable** method is called. When the name that is retrieved is longer than 30 characters, the name is now truncated to 30 characters. In addition, if the first character of the name is uppercase, it is changed to lowercase.

This name is used to determine the method to invoke when using non-wrapped document literal format messages. When the name does not meet the method-naming requirements of JADE, the method invoked will likely fail and a SOAP fault will be returned to the Web service consumer.

## Java Framework Support for HugeStringArray Objects (NFS 54330)

The JADE Java framework now supports **HugeStringArray** objects in addition to the existing support for **StringArray** objects. This allows arrays to contain strings that are longer than 62 characters, as shown in the following code fragments.

```
import com.jadeworld.jade.rootschema.HugeStringArray;
...
@EntityProperty(name="fred", type="HugeStringArray")
public HugeStringArray getFred() {
    return (HugeStringArray)EntityAccess.getReferenceProperty(this, "fred");
}

public void setFred(HugeStringArray strArray) {
    EntityAccess.setReferenceProperty(this, "fred", strArray);
}

public void addFred(String str) {
    HugeStringArray hugeSA = getFred();
    hugeSA.add(str);
}

...
HugeStringArray hsa = new HugeStringArray();
em.persist(hsa);
...
myClass.setFred(hsa);
myClass.addFred("A String");
```

## Journal Rate Analysis Sampling (PAR 57125)

The sampling interval granularity has been changed from seconds to milliseconds. The default sampling interval remains one minute, expressed as **60000** milliseconds.



Analysis, generation of summary and total values, and interval snapshot data have been enhanced, by the transaction count now being the number of active transactions seen derived from the transaction IDs of the update records processed, rather than the number of **BEGIN\_TRANSACTION** records seen.

An additional sample column containing the number of **COMMIT\_TRANSACTION** records seen has been added. The file summary and totals summary have also been enhanced to report on the transaction and commit activity.

## Monitoring Node Locks (NFS 54406)

The **Locks** view in the JADE Monitor now provides the **Exclude node locks** check box, which you can check if you want to exclude node locks from the persistent locks list (for example, if you want to avoid filling the lock table with large numbers of stable object node locks when investigating locking issues).

## Reblocking Collection Class Maps

You can use the reblocking collection class maps facility after upgrading an environment from release 6.3 to 7.0, to reblock collections for the best fit to the new file structure.

You can reblock all collections in one or more map files, by using the **JadeCollectionReblocker** application in the non-GUI client (**jadclient**) program in single user or multiuser mode; that is:

```
jadclient.exe path=database-path
             ini=jade-initialization-file
             server=SingleUser|MultiUser
             schema=user-schema-name
             app=JadeCollectionReblocker
             File=file-name|mask
             [File2=file-name [File3=file-name]...]
             [workers=number-of-worker-threads]
```

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 reblocking of all user files, including **rootdef**. Specify the mask **\_\*** (underscore asterisk characters) to indicate the reblocking of all system files such as **\_userscm**.

The following are examples of the reblocking action.

```
jadclient path=d:\jadeuser ini=d:\salesdb\jade.ini schema=Sales server=singleUser
app=JadeCollectionReblocker File=*

jadclient path=d:\jadeuser ini=d:\salesdb\jade.ini schema=Sales server=singleUser
app=JadeCollectionReblocker File1=testdb File2=banking File3=_userdev workers=5
```

The optional **workers** parameter enables you to specify the number of multiple concurrent worker threads that are used. The default value of **1** is used if you do not specify the **workers** parameter. You can specify a value in the range 1 through 16. If you specify a value greater than **16**, the maximum number of 16 worker threads is used.

Reblocking is done in normal update transactions, which are committed every 1,000 collections or 30 seconds.

Reblocking skips collections that do not require reblocking. The **JadeCollectionReblocker** application can be terminated part way through a file, in which case only uncommitted changes are discarded. When reblocking is restarted for the same file, collections that have already been reblocked are skipped.

## Relational Population Service (RPS)

The following sections describe the RPS changes in this release.

## Adding an Existing Property or Method to an RPS Table (NFS 57061)

The new **DropHistoricalTableOnAddExisting** parameter in the [JadeRps] section of the JADE initialization file controls the behavior when you add an existing property or method to a historical table in an RPS mapping. The default value of this parameter is **true**; that is, the historical table is dropped if an existing property or method is added.

To modify the historical table (rather than drop the table) when an existing property or method is added, set the value of this parameter to **false**; which will then result in an **ALTER TABLE <> ADD** being used to modify the table when an existing property or method is added. Any existing rows will have a column value of **NULL**.

This parameter is read when the alter table script is created.

## Converting OID Columns (PAR 57473)

The conversion of an OID column type from **String** to **INTEGER/INTEGER** is now supported.

In addition, the following JADE schema changes are achieved using **ALTER TABLE** commands.

- Changing a JADE primitive type from **Byte** to **Integer** or **Integer64**
- Changing a JADE primitive type from **Integer** to **Integer64**

## Datapump Application Execution (PAR 53009)

The **Datapump** application must always be in a state in which it can be executed without error. If schema changes put the **Datapump** application into a non-executable state, the RPS node cannot continue replay and must be recreated from the primary.

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

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

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

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

## Extracting an RPS Table (PAR 55803, PAR 56829)

When extracting data for an RPS mapping, the data is now buffered to improve performance.

The [JadeRps] section of the JADE initialization file can now contain the **ExtractBufferSize** parameter, which enables you to set the buffer size when extracting RPS files. A buffer of the specified size is allocated for each concurrent file being written.

The default value is **1M** and the minimum value is **8K**.

The parameter is read when the RPS node is initialized.

## Silverlight

The following sections describe Silverlight XAML changes in this release.

### Accessing Third-Party Controls (PAR 56871)

The **XamlDocument** class now provides the **registerCallback** method, which has the following signature.

```
registerCallback(callback: JadeMethod): XamlObject;
```

This method registers a JADE method for calling back and returning a **XamlObject** instance that can then be passed as a parameter to a method or property accessed by the **XamlObject** class **setXamlProperty** or **invokeXamlMethod** method.

---

**Note** The JADE method must have parameters that are compatible with the .NET method expected.

---

The following method defined in a **XamlObject** subclass is an example of the **registerCallback** method.

```
layoutRoot_loaded(control: XamlGrid input; originalSource:
                    XamlObject input) updating, browserExecution;
vars
    dto      : DataDTO;
    callback : XamlObject;
begin
    // Set dragDrop callback
    callback := registerCallback(DragDrop::onDragQuery);
    rightBox.invokeXamlMethod(DragDropManagerType &
        ";AddDragQueryHandler", root, callback);
    callback := registerCallback(DragDrop::onDropQuery);
    rightBox.invokeXamlMethod(DragDropManagerType &
        ";AddDropQueryHandler", root, callback);
    callback := registerCallback(DragDrop::onDragInfo);
    rightBox.invokeXamlMethod(DragDropManagerType &
        ";AddDragInfoHandler", root, callback);
    callback := registerCallback(DragDrop::onDropInfo);
    rightBox.invokeXamlMethod(DragDropManagerType &
        ";AddDropInfoHandler", root, callback);
    create dto transient;
    rightBox.itemsSource := dto.getData();
    delete dto;
end;
```

## Extracting a Schema to Generate a XAP File (PAR 57458)

When a Silverlight schema is extracted to generate a XAML Application (XAP) file, properties of type **MemoryAddress** are now dropped.

## JADE Properties for XAML Controls inside DataTemplates (PAR 57187)

When you save XAML in JADE, any elements with an **x:Name** attribute have a corresponding control property representing the XAML document created in the **XamlDocument** subclass. These names can then be used in JADE logic to access properties of the controls.

During the C# and Silverlight project generation, this JADE logic is converted to C# and adds logic to call methods in a utility assembly to map the JADE property that corresponds to the control on to the actual Silverlight object.

However, when elements are part of a **DataTemplate**, they are not created on document load and the C# generated from your JADE logic is therefore unable to find the real Silverlight object. Accessing such properties caused exceptions to be raised. To avoid this, JADE does not create JADE properties for XAML controls that are inside **DataTemplates**, even if they do have an **x:Name** attribute.

Because subclasses of **DataTemplate** will not be detected when a document is loaded, existing XAML documents may raise exception 14451 if they have elements with a name attribute that are subelements of a **DataTemplate** element (or a **DataTemplate** subclass).

## XAML Browser (PAR 57192)

The following changes have been made to the XAML Browser.

- The XAML menu now provides the **Find Document** command, which accesses the Find Document dialog, to enable you to select from a list of XAML documents.
- The Edit menu is now displayed when the XAML Browser has focus. (This modified Edit menu does not provide the **Global Find** or the **Macro** command.)

This Edit menu is also displayed as the context (popup) menu for the XAML editor pane in the XAML Browser.

## Synchronized Database Service (SDS) (PAR 57217)

The following subsections describe the changes made to improve the utilization of the links between a primary database and its secondaries.

- Setting the socket buffer sizes (the default value is 128K bytes, compared to the Windows operating system default value of 8K bytes).
- Consolidating multiple small journal blocks into a single message.
- Changing secondary journal writing to use buffered I/O.

### JadeDatabaseAdmin::getCurrentJournalOffset Method

The **JadeDatabaseAdmin** class now provides the **getCurrentJournalOffset** method, which has the following signature.

```
getCurrentJournalOffset (currentJournal: Integer64 output;
                        currentOffset: Integer64 output;
                        lastSwitchJournal: Integer64 output;
                        lastSwitchOffset: Integer64 output;
                        nominalSize: Integer64 output);
```

The **getCurrentJournalOffset** method retrieves the journal number and byte offset of the last record written to the journal in the respective **currentJournal** and **currentOffset** parameters.

The **lastSwitchJournal** and **lastSwitchOffset** parameters retrieve the respective journal number and byte offset of the last record written to the penultimate journal.

The **nominalSize** parameter returns the nominal size of a journal file (that is, the value of the **JournalMaxSize** parameter in the [PersistentDb] section of the JADE initialization file).

These values enable you to calculate amounts and rates of journal output.

Use this method in conjunction with the **JadeDatabaseAdmin** class **sdsGetSecondaryProxy** method to determine the amount of journal data that has not been sent to the secondary.

### Changed JadeDatabaseAdmin::sdsGetSecondaryProxy Method

The following Integer64 attributes are now added to the returned **JadeDynamicObject** by the **JadeDatabaseAdmin** class **sdsGetSecondaryProxy** method.

- **totalSends**, which is the count of messages sent to the secondary.
- **totalBlocksSent**, which is the count of journal blocks sent to the secondary. There can be from 1 through 16 blocks per message.
- **totalBytesSent**, which is the count of bytes sent to the secondary; that is, the total size of all messages sent.
- **totalUncompressedBytes**, which is the count of bytes sent to the secondary if compression was disabled.

- **lastRecordSentJournal**, which is the journal number of the last journal record sent.
- **lastRecordSentOffset**, which is the byte offset of the last journal record sent.

## [ConnectionParams] Section SocketBufferSize Parameter

The [ConnectionParams] section of the JADE initialization file can now contain the **SocketBufferSize** parameter, which enables you to customize TCP/IP buffers to match the link characteristics.

The default value is 128K bytes, the minimum value is 8K bytes, and the maximum value is 16M bytes.

You can also set this parameter to zero (**0**), in which case the operating system default value is used (for example, Windows 7 has a default value of 8K bytes).

This parameter is read when the secondary opens a connection to the primary and when the primary accepts a connection from a secondary.

## System::*verifyDbEncryptionMasterKey* Method (NFS 57338)

The **System** class now provides the **verifyDbEncryptionMasterKey** method, which specifies whether the database encryption master key is present and correct; otherwise it returns **false**.