


Upgrading to JADE Release 7.1

VERSION 7.1.03

jade™

Copyright©2015 Jade Software Corporation Limited. All rights reserved.

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

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

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

Copyright © 2015 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

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Upgrading to JADE Release 7.1.03

This document covers the following topics.

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** document in the **\documentation** directory.

Note To customize the deployment upgrade on Windows, see "Customizing the Deployment Upgrade Process", in Appendix A of the *JADE Runtime Application Guide*.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Important Information

You must perform each of the following actions before *and* after upgrading to JADE release 7.1.

Caution Proceed to the next step only when all errors reported in the current step have been resolved.

1. On your current system *before* upgrading to JADE 7.1, perform each of the following actions.
   a. Perform a physical certify operation using the JADE Database Utility (**jdbutil.exe** or **jdbutilb.exe**), to ensure that the system is structurally sound.
   b. Perform a meta logical certify operation, to ensure that the meta model is clean.
   c. Perform a logical certify operation, to ensure that the user data is referentially correct.
   d. If your database has partitioned database files, ensure that any offline partitions are brought online.
   e. Take a full backup copy of your existing JADE 7.0 or JADE 6.3 directories, first ensuring that your database is not in recovery mode.
   f. If the **JadeReportWriterSchema** has been installed, you must extract the data and then delete the schema before upgrading; that is:
      i. In the JADE Report Configuration application, select the **Unload All** command from the View menu to unload all of your report data to a .**rwa** file. In the **File Name** text box of the Unload All dialog, specify the name and location of the file to which you want to extract all of your report writer data; for example:

         ```
         d:\jade\rpts\alldata.rwa
         ```

         Alternatively, you can use the **executeMethod** in the batch JADE Load utility to extract all reports, as shown in the following example.

         ```
         jadloadb path=d:\jade\system ini=d:\jade\myjade.ini
         schema=JadeReportWriterSchema executeSchema=JadeReportWriterSchema
         executeClass=JadeReportWriterGlobal executeMethod=unloadAllToFile
         executeParam=d:\jade\rpts\alldata.rwa
         ```

      ii. Delete the **JadeReportWriterSchema** schema by using the **Remove** command in the Schema menu of the JADE development environment or by using the **deleteSchema** parameter in the batch JADE Load utility; for example:

         ```
         jadloadb path=d:\jade\system ini=d:\jade\myjade.ini
         deleteSchema=JadeReportWriterSchema
         ```

2. In JADE 7.1 immediately after your upgrade from an earlier release, perform each of the following actions.
   a. Perform a physical certify operation using the JADE Database Utility (**jdbutil.exe** or **jdbutilb.exe**), to ensure that the system is structurally sound.
   b. Perform a meta logical certify operation, to ensure that the meta model is clean.
   c. Perform a logical certify operation, to ensure that the user data is referentially correct.
   d. If the **JadeReportWriterSchema** is required, reload the 7.1 **JadeReportWriterSchema** and then reload the reports and data extracted in step 1.f of this instruction, by using one of the following actions.
      – In the JADE Report Configuration application, select the **Load All** command from the View menu to load all report data from your extracted .**rwa** file.

        In the **File Name** text box of the Load All dialog, specify the name and location of the report data extract file you want to load; for example:

        ```
        d:\jade\rpts\alldata.rwa
        ```

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- Use the **reportLoadAllFile** parameter in the batch load utility to specify the fully qualified name of a single unload (extract) file that contains all JADE Report Writer view, folder, system option, user, and report definitions that you want to load, as shown in the following example.

```
jadloadb path=d:\jade\system reportLoadAllFile=d:\jade\rpts\alldata.rwa
ini=d:\jade\myjade.ini
```

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

JADE Release Support

For details about the JADE 7 release policy, see http://www.jade.co.nz/jade/updates.htm#releasesched.

JADE 7.1 is built using Microsoft Visual Studio 2013, which requires the installation of appropriate C++ runtime binaries.

Deimplementations and Deprecations

This section contains the deimplementations and deprecations in this release.

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. 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

As notified in JADE 7.0.10, JADE's Java framework has been deprecated in this release.

A schema load now fails with error 6449 (*Java exposures no longer valid*) if a Java exposure exists in the schema being loaded.

NoCRTRuntimeUpgrade Initialization File Parameter

The **NoCRTRuntimeUpgrade** parameter in the [JadeAppServer] section of the JADE initialization file is no longer supported.

Process Class *getTransientOnlyClassesUsed* Method

Because the extension of class numbers has resulted in the deprecation of transient-only classes, the **Process** class **getTransientOnlyClassesUsed** method has been deprecated in this release.

This method is still defined in the **RootSchema**, but calling it raises exception 1112 *(Feature has been deprecated)*.

Replication Framework Deprecation

The JADE Replication Framework has been deprecated in this release.

Server Read-Only User Deprecation

As notified in JADE 7.0.10, the **ReadOnlyUser** value of the **server** parameter in the **jadclient** executable has been deprecated in this release, as there is no longer a requirement to run JADE from read-only media. (**ReadOnlyUser** is therefore no longer a valid value for the **Server** parameter in the [NonGuiClient] section of the JADE initialization file.)

Silverlight XAML

As notified in JADE 7.0.10, JADE's Silverlight Extensible Application Markup Language (XAML) implementation has been deprecated in this release.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Transient-Only Classes

With the extension of class numbers in this release, the transient-only class feature is no longer available. (See also "Class Number Extensions" under "Changes and New Features in JADE Release 7.1.03".)

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Highlights in this Release

The highlights in JADE release 7.1, which help you to deliver high-performance, interoperable applications on Windows for both 32-bit and 64-bit platforms, are as follows.

- Class number extensions

  JADE has extended the number of user classes permitted to just under one million. In earlier releases, the limit for user classes was 31,279, with some class numbers reserved for transient-only classes. For details, see "Class Number Extensions", later in this document.

- Ad hoc indexes

  Ad hoc indexes enable you to create indexes suitable for optimizing ad hoc queries without requiring database reorganization. For details, see "Ad Hoc Indexes", later in this document.

- Web services extensions

  JADE now provides the Representational State Transfer (REST) Application Programming Interface (API), which provides you with an alternative to the Simple Object Access Protocol (SOAP) protocol to implement Web services. For details, see "REST-Based Web Services", later in this document.

- .NET API enhancements

  JADE has made a number of changes to facilitate your .NET development, including:

  - User class functionality, dynamic property access, and JADE method call invocation performance improvements
  - A **jadserv** equivalent of the **JoobDatabaseServer** that works with standard JADE binaries to enable you to write your own plug-in .NET assemblies for authentication and authorization support via interface implementation
  - Implementation of a **ServiceModel** assembly that enables C# proxy objects to be serialized across .NET Windows Communication Foundation (WCF) Web services, also providing support for .NET WCF client applications to participate in system transactions
  - The *JADE .NET Developer's Reference* has been enhanced to include .NET and JADE requirements, object management information, a sample application, and an introductory tutorial

  For details, see ".NET", later in this document.

- Extended dynamic property support

  You can now access JADE dynamic properties explicitly, rather than using calls to the **getPropertyValue** and **setPropertyValue** methods. The JADE compiler supports enhancement this explicit dynamic property access and allows the extraction and load of schemas that contain dynamic properties. For details, see "Dynamic Properties", later in this document.

- Unaudited database files and partitions

  Database files and partitions can now be updated with auditing disabled, to eliminate journal disk space use and I/O overhead when loading data into a file or partition and restart recovery is not a requirement. For details, see "Unaudited Database Files and Partitions", later in this document.

- Snapshot database recovery

  JADE now supports the recovery of database restored from a snapshot created by third-party technology; for example, VMWare snapshots. For details, see "Snapshot Database Recovery".

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Platform Requirements for this Release

JADE 7.1 runs on the Windows operating systems specified in the following list, and provides a 64-bit database server with both 32-bit and 64-bit clients available.

- Database server, available in 64-bit only
  - Windows Server 2012 with the latest security updates
  - Windows 8.1 with the latest security updates
  - Windows Server 2008 (R2 recommended) with the latest security updates
  - Windows 8 with the latest security updates
  - Windows 7 with the latest security updates
  - Windows Small Business Server 2011 with the latest security updates
  - Windows Vista Business (recommended for development and testing only)

    Windows Vista support from Microsoft is currently scheduled to end in November 2017, at which time JADE will no longer support it.
- Application servers and standard clients, available in 32-bit and 64-bit but must be 64-bit if hosting a database
  - Windows Server 2012 with the latest security updates
  - Windows 8.1 with the latest security updates
  - Windows Server 2008 (R2 recommended) with the latest security updates
  - Windows 8 with the latest security updates
  - Windows 7 with the latest security updates
  - Windows Small Business Server 2011 with the latest security updates
  - Windows Vista Business (recommended for development and testing only if hosting a database)

    Windows Vista support from Microsoft is currently scheduled to end in November 2017, at which time JADE will no longer support it.
- Presentation client workstation, available in 32-bit and 64-bit
  - Windows Server 2012 with the latest security updates
  - Windows 8.1 with the latest security updates
  - Windows Server 2008 (R2 recommended) with the latest security updates
  - Windows 8 with the latest security updates
  - Windows 7 with the latest security updates
  - Windows Small Business Server 2011 with the latest security updates
  - Windows Vista Home Premium with the latest security updates

    Windows Vista support from Microsoft is currently scheduled to end in November 2017, at which time JADE will no longer support it
- Compact JADE thin client only

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- Windows Mobile 5.0 for Pocket PC Phone Edition
- Windows Mobile 5.0 for Pocket PC
- Windows Mobile 6.0 or 6.1 Classic
- Windows Mobile 6.0 or 6.1 Professional
- Windows Mobile 6.5 Professional

Note Windows Mobile 6 Standard and Windows Phone 7 are not supported.

To deploy JADE Web applications under Apache, a minimum of Apache 2.4.10 for Microsoft Windows is required.

Any node that hosts a JADE 7.1 database must be 64-bit, including single user application servers and single user standard clients. The 32-bit clients that are available are multiuser clients (application servers, presentation clients, and standard clients) that connect to the 64-bit database server.

Any system that is to run JADE 7.1 must have the Microsoft Visual C++ Redistributable Package installed. For JADE 7.1:

- 64-bit versions, the package is Visual C++ 2013 Redistributable Package (x64), version 12.0.30501
- 32-bit versions, the package is Visual C++ 2013 Redistributable Package (x86), version 12.0.30501

To install Visual C++, execute one of the following from **<jade>\<a_>bin** or **<u_>bin** on the release medium, as applicable.

- vcredist_x64.exe
- vcredist_x86.exe

For example:

```
jade7103.002\Files\Ansi64\a_bin\vcredist_x64.exe
```

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

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 Managements 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 parsyssupport@jadeworld.com. You can download the install shield for **Parsys** from the following URL.

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.
   a. On the **Basic Search Criteria** sheet, the **Latest** option button is selected in the Mode group 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** and **NFS** types are selected.
   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 **A hot fix was 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 **7.1.03** list item).
3. Click the **Search** button.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Upgrading to JADE 7.1

This section covers the following topics.

Caution Before you upgrade to JADE 7.1, refer to "Important Information" and to "JADE 7.1 Changes that May Affect Your Existing Systems", elsewhere in this document.

Upgrading to JADE 7.1 from an Earlier Release

Server nodes (database upgrades) require the 64-bit edition. Client nodes can upgrade the 64-bit edition or the 32-bit edition. This upgrade involves a migration of the data from an earlier release to JADE 7.1, due to the nature of the structural changes of the objects in the database.

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

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.

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 an earlier JADE release. If you require these files for your JADE system, you must save them before you upgrade or restore them from the original JADE 7.0 or 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.

The JADE Setup program enables you to upgrade your binary and database files to JADE 7.1 from a JADE 6.3 release or higher, by performing the following actions.

1. Ensure that your JADE environment is JADE release 6.3 or higher.

   On the JADE 7.0 or 6.3 system, 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

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

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) 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 7.1 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 Visual C++ 2013 Redistributable Package (x64) is required. If you are upgrading the 32-bit edition of JADE, the **vc_red.msi** Microsoft Visual C++ 2013 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 **Jade71** release medium or execute the executable program downloaded from the JADE Web site.
6. If it is 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 installation.
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 7.1 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
    ```

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

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:\jade71** in the **Install Directory** text box, the database directory defaults to **c:\jade71\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\Jade71** in the **Install Directory** text box, the default database location is **c:\ProgramData\Jade71\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.

19. Install any **jadrap** database services (that is, JADE Remote Node Access services) you had set up in JADE 7.0 or 6.3. (For details, see "Running the Server Node as a Service", in the *JADE Remote Node Access Utility User's Guide*.)

After the upgrade has completed, the database files of the earlier release are retained in the directory ***database-directory*_70** or ***database-directory*_63** (for example, **c:\jade\system_70**). The *database-directory* (for example, **c:\jade\system**) contains the upgraded JADE 7.1 database.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

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 7.0 or 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 7.0 or 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 7.1 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.

JADE Thin Client Upgrade

When upgrading a presentation client to JADE release 7.1:

- Presentation client upgrades from JADE release 6.3.04 or earlier are rejected.
- Reverting from JADE 7.1 to JADE 6.2 is rejected.
- If you are upgrading presentation clients to JADE release 7.1, ensure that you have the appropriate privileges or capabilities to install applications. The configuration of User Account Control (UAC) and your current user account privileges may affect the behavior of the upgrade to JADE 7.1. For details about UACs, standard user accounts, and administrator accounts, see the Microsoft documentation.

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 the machine has had UAC disabled, the thin client upgrade will fail because of lack of permissions for standard users. For administration users, the necessary privileges are automatically granted so the upgrade will succeed.
- If UAC is not disabled, 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

For versions prior to JADE 7.1 and 7.0.11, 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).

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- 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 2013 C++ runtime binaries.

Notes Visual C++ runtimes are always upgraded (that is, the **x64** version is installed) as part of the upgrade process.

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.

In JADE 7.1, 32-bit presentation clients can be upgraded to 64-bit, as follows.

- The **UpgradeRuntimeTo64bit** parameter in the [JadeAppServer] section only on each application server, when set to **true**, causes a 32-bit client running on a 64-bit operating system (WoW64) to be upgraded to 64-bit binaries.
- The **CustomRuntimeUpgrade** parameter in the [JadeAppServer] section only on each application server allows the environment type of the binaries to be specified for individual clients. Use this parameter to upgrade specific presentation clients to 64-bit binaries, or use the parameter to force specific clients to remain using 32-bit binaries when the value of the **UpgradeRuntimeTo64bit** parameter is set to **true**.

If the client is not running a JADE version that is using Visual Studio 2005 runtime libraries, the JADE thin client download will not revert to a version that uses Visual Studio 2005 runtime libraries.

For more details, see "Upgrading 32-bit Thin Client Binaries to 64-bit (NFS 61726)", later in this document.

Caution The **UpgradeRuntimeTo64bit** parameter takes affect *only* after the client binaries have downloaded the release that supports this option. This requires two thin client downloads to achieve the 32-bit to 64-bit upgrade (that is, one download to update to JADE 7.1, and a second to upgrade to 64-bit).

Upgrading a Synchronized Database Environment (SDE)

A JADE release 6.3 or 7.0 to JADE release 7.1 upgrade of a Synchronized Database Service (SDS) node is not supported.

When the upgrade to 7.1 has completed successfully, re-clone the secondary databases from the upgraded primary database.

Upgrading an RPS Node

A JADE release 6.3 or 7.0 to JADE release 7.1 upgrade of a Relational Population Server (RPS) node is not supported.

When the upgrade to 7.1 has completed successfully, re-clone the RPS node from the upgraded primary database and recreate the Relational Database Management System (RDBMS) target database from the recreated 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 **jommsg*n*.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.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

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.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

JADE 7.1 Changes that May Affect Your Existing Systems

This section describes only the changes in the JADE 7.1 release that may affect your existing systems. Some changes may result in compile errors during the load process, or cause your JADE release 7.1 systems to behave differently. For details about the changes and new features in JADE release 7.1.03, see "Changes and New Features in JADE Release 7.1", later in this document.

Compact JADE (PAR 61967)

With the release of Windows Mobile 6.5.3, Microsoft has moved the **OK** and **X** buttons from the top of a maximized form to the soft menu bar at the bottom of the screen. However, as JADE creates a user-defined menu bar and Windows does not add the **OK** or **X** button to that type of menu bar, users could not close a maximized form.

This has been changed as follows for all supported versions of Windows Mobile.

- The Compact JADE control panel configuration program (**jadeconfig.cpl**) has been changed to add a permanent soft menu bar. Under Windows Mobile 6.5.3, this bar contains an **OK** button. For earlier Windows Mobile versions, the **OK** button is displayed in the caption line of forms.

  This enables users to commit configuration changes under version Windows Mobile 6.5.3.

- The **jade.exe** program has been changed for all Windows Mobile versions so that it creates a standard Windows soft menu bar if there are two or fewer visible top-level menu items and they do not have images defined. Those top-level menus are assigned to the soft menu buttons on the menu bar (a maximum of two are available).

  Under Windows Mobile 6.5.3, the menu bar also contains the Windows button and an **X** or **OK** button, if required, for a maximized form. For earlier Windows versions, the **X** or **OK** button continues to be displayed in the caption line of the form. If there are more than two visible top-level menu items or a top-level item has a defined image, a standard Windows soft menu bar cannot be created. The menu bar is user-defined, and under Windows Mobile 6.5.3, there is no **X** or **OK** button displayed.

  Notes In this situation, your application has to include a user-defined menu item that performs the close functionality using JADE logic. The alternative solution is to redefine the menu structure so that there are two top-level menu items only.

  With the use of a standard Windows menu bar, a soft menu item button displays only a small amount of text. If the text is too long, the text is truncated with trailing points of ellipsis (**...**).

If you want your application to retain the previous JADE menu bar style:

1. Ensure that the form is created with three or more visible top-level menu items (or one with an image).
2. In the **load** event of the form, hide any menu items that do not need to be shown. If the number of menu items is decreased to two or fewer, the menu bar style remains as user-defined.

   However, if the number of visible top-level items is increased beyond two, Windows changes the menu bar to a user-defined style and the **X** or **OK** button will again not be displayed.

JADE Remote Node Access (PAR 60364)

When the **Run as Service** check box is unchecked when the JADE server node is still running as a service, a message box is now displayed, advising you that you cannot uncheck this option when the service is running and that you must first stop the service.

In addition, the service state (for example, running, stopped, and so on) is now continuously updated to reflect service state changes made externally; for example, by using the Task Manager.

Note The same behavior applies also to the JADE application server (**jadapp.exe**).

See also "JADE Remote Node Access Utility", under "Changes and New Features in JADE Release 7.1".

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Relational Population Service (RPS) Mapping

If you are likely to use class numbers greater than 99,999 in your JADE 7.1 system, you should ensure that your **OID Mapping** option on the **Define RPS** sheet of the Relational Population Service wizard is set to **Split Into Columns Int/BigInt**. (See also "Class Number Extensions" under "Changes and New Features in JADE Release 7.1".)

Caution If you reach the 99,999 class limit in the RPS system and the oid mapping is set to **Map to String**, you will have to recreate the RPS node.

Replayable Reorganization (NFS 60428)

In the batch JADE Schema Load utility (**jadloadb**), the handling of the optional **replaybleReorg** 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.

SOAP XML Handling in Web Services (PAR 60987)

When handling Simple Object Access Protocol (SOAP) Extensible Markup Language (XML) in Web services in earlier releases:

- **TimeStampOffset** JADE type values were encoded in the generated XML with the wrong sign for the Coordinated Universal Time (UTC) bias; that is, **-** instead of **+** and **+** instead of **-**. The JADE SOAP parser also expected the wrong sign to establish a passed value. This meant that JADE to JADE handling was be correct, but C# to JADE communication resulted in the wrong value.
- A negative **TimeStampInterval** JADE type value ignored the negative sign when the value was parsed by JADE, and the interval was always positive. In addition, the XML constructed by JADE did not include a negative sign for a negative **TimeStampInterval** value.

The **TimeStampOffset** value now has the correct sign for UTC bias and the **TimeStampInterval** value negative sign is now recognized by the JADE parser and included in the XML constructed by JADE.

String Arrays and Dictionary Keys

As the 30-character limit for schema entity names has increased in JADE 7.1 to the maximum of 100 characters (for details, see "Entity Name Maximum Length"), the **StringArray** class now provides the **JadeIdentifierArray** subclass, 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 **JadeIdentifierArray** is 100.)

If you are using entity names as dictionary keys, you must also consider the size of your dictionary key, 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.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Changes and New Features in JADE Release 7.1

This section summarizes the changes and new features in JADE release 7.1. For details about the changes in release 7.1 that may affect your existing systems, see "JADE 7.1 Changes that May Affect Your Existing Systems", earlier in this document.

.NET

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 JoobContext()".

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 JoobContext 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.

C# Exposure Wizard

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

Exposure Entity Maximum Lengths (NFS 61179)

The JADE C# exposure wizard enables you to specify an exposure name with a maximum of 100 characters. (In earlier releases, the exposure name could not exceed 30 characters.)

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.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Saving Definition and Generate Values (PAR 59367)

The C# Exposure Wizard now saves and restores the following values.

- Name space
- The **Current schema only** or **Superschemas up to** value
- The name of the **Top Most Schema** if the **Superschemas up to** option button is selected
- The last output directory used to generate the exposure
- The **Sign-on Schema Name** value
- The **Sign-on Application Name** value
- The value of the **Sign-on to Jade as Multiuser** check box

Custom Control Tab Order (PAR 61929)

In releases earlier than JADE 7.0.11, tabbing into and within .NET controls in JADE that use Microsoft control subclasses was not handled.

Tab handling for .NET controls in JADE that use Microsoft control subclasses has now been implemented so that tabbing into the control is seamlessly integrated with other JADE controls on the form. If the .NET control has more than one subcontrol that can have focus, tabbing now moves through those subcontrols in .NET TabIndex order. When the TAB key is pressed on the last such subcontrol, focus moves to the next JADE control in the JADE **tabIndex** order. (SHIFT+TAB performs the same function in reverse **tabIndex** order.)

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 Chapters 5, 6, and 7, respectively.)

Chapter 1	.NET and JADE requirements
Chapter 2	JADE .NET object management
Chapter 3	Sample client-server application
Chapter 4	Introduction to JADE .NET tutorial
Appendix A	Mapping JADE primitive types to Common Language Runtime (CLR) data types
Appendix B	Developing applications to store, edit, and query spatial information directly through the JADE .NET API

Ad Hoc Indexes

Ad hoc indexes, which enable you to create indexes suitable for optimizing ODBC ad hoc queries without requiring database reorganization, are defined as a subclass of **DynaDictionary**, with instances mapped to a RootSchema map file called **_sindexes.dat**. The **_sindexes.dat** database file is partitioned (for example, **_sindexes_part0000000001.dat**), with each ad hoc index instance created in a new partition to facilitate a fast drop index operation, which uses the existing **dropPartition** operation. The classes that represent index meta data are mapped to the **_sindexdefs.dat** RootSchema database file. (The **_sindexes.dat** and **_sindexdefs.dat** files are of **Kind_Utility**.)

The **Ad Hoc Index Controller** application reads from the main database (or from journals when catch-up is required), and creates and maintains the user-defined index. The ODBC driver function that performs ODBC queries can then use a combination of ad hoc indexes and collections in the main database to satisfy queries.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

The JADE development environment Ad Hoc Index Browser enables you to create or modify ad hoc index definitions, and to drop unwanted indexes. As the ad hoc index definition does not define any meta data, the data is not included when the schema is extracted. The Ad Hoc Index Browser enables you to save an ad hoc index to an Extensible Markup Language (XML) file and to load a saved index. For details, see "Maintaining Ad Hoc Indexes", in Chapter 9 of the *JADE Development Environment User's Guide*. For details about using the **jadclient** executable to create indexes suitable for optimizing ad hoc queries without requiring database reorganization, see "Ad Hoc Index Batch Interface", in Chapter 1 of the *JADE Runtime Application Guide*.

Adding User Classes at Run Time

Your user applications can add user classes at run time. While these user-defined classes are visible in the JADE development environment, they are not considered as part of the JADE model, you cannot reference them directly 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	addUserCollectionSubclass	Creates a user collection class as a subclass of the specified superclass in the receiving schema
Schema	addUserSubclass	Creates a user class as a subclass of the specified superclass in the receiving schema
Schema	deleteUserSubclass	Deletes a user class from the specified superclass in the receiving schema
JadeUserCollClass	addExternalKey	Adds an external key definition to a user class at run time
JadeUserCollClass	addMemberKey	Adds a member key definition to a user class at run time
JadeUserCollClass	clearKeys	Clears existing key definitions
JadeUserCollClass	endKeys	Indicates the end of a single or multiple key definition
JadeUserCollClass	setLength	Sets or changes the element length for an array
JadeUserCollClass	setMembership	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.

Pseudo Type Variable Support (PAR 61634)

The JADE editor AutoComplete feature has been enhanced as follows.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- The JADE pseudo types of **InstanceType**, **MemberType**, and **SelfType** are now handled, and offer AutoComplete suggestions as required.

  Note **KeyType** parameters are not handled.

- The displayed method signature handles a **ParamListType** pseudo type parameter when highlighting the current parameter in the displayed method signature.

  Note, however, that if the first parameter of the method being called is a **ParamListType**, if the terminating bracket of the call has not yet been entered, it is not possible to accurately determine which parameter is currently being entered when there is more than one parameter.

Application Class *isValidObject* Method (PAR 52806)

The performance of the **Application** class **isValidObject** method has been improved. This method now takes into account the presence of the object in the cache if automatic cache coherency is enabled.

Application Programming Interface (PAR 60679)

The JADE C Application Programming Interface (API) has been extended, as follows. (For details, see Chapter 3 and Chapter 6, respectively, of the *JADE Object Manager Guide*.)

- The JADE API **jomcalls.h** file now provides the following Application Programming Interface (API) call.
  - jomGetNodeContextHandle
- The existing header file **joscalls.h** in the **include** directory on the JADE release medium now contains the following new functions.
  - josDskDateToGregorian
  - josDskTimeStampToGregorianHMSm
  - josDskTimeStampCompare
  - jomDecimalFromReal
  - jomDecimalFromInt64Scale
  - jomDecimalFromString
  - jomDecimalToReal
  - jomDecimalToString
  - jomDecimalCompare
- The **include** directory on the JADE release medium now contains the new header file **jombuild.h**, which provides defines that enable you to perform build and runtime version checking. Each JADE Dynamic Link Library (DLL) exports a function named using the define **MAKEMODULEVERSIONFUNC**; for example, the function name exported by **jomutil.dll** in the 7.1.03 version of JADE is **"moduleVersion_jomutil_7_1_03"**.

  Static importers of the DLL can use the following to ensure that the importing DLL or executable, the **jomutil.lib**, and the **jomutil.dll** are consistent.

  Declaration:

```
extern "C" DllImport int JOMAPI MAKEMODULEVERSIONFUNC(jomutil)();
```

  Invocation:

```
int jomutilpatch = MAKEMODULEVERSIONFUNC(jomutil)();
```

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

If the **jombuild.h** version included by the user sources does not match the **jomutil.lib**, the compile-time linker raises an error; for example:

```
mysource.obj : error LNK2019: unresolved external symbol __imp_moduleVersion_
jomutil_7_0_7 referenced in function MyVersionChecker
```

At run time, if the importer version and the **jomutil.lib** version do not match **jomutil.dll**, the dynamic linker raises the following.

```
The procedure entry point moduleversion_jomutil_7_0_8 could not be located in
the dynamic library jomutil.dll.
```

Class Number Extensions

JADE has extended the number of user classes permitted to just under one million. The:

- First user class number is 2,048.
- Maximum user class number is 999,999.
- Number of user classes is 997,952.
- Maximum number of exclusive collections (subobjects), binary large objects (blobs), and string large objects (slobs) in a class is 65,535.

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;
```

create Statement Handling (PAR 62019)

JADE now handles **create** statements in which the **as** expression is a static class name. (If the **as** expression is not a static class name, the **create** statement is ignored because the type returned by the expression cannot be determined.)

The type of the **create** statement is treated as transient if the **create**:

- Qualifier is **transient**
- Is not qualified and the class to be created is defined as defaulting to transient

Disk Write-Cache Messages (PAR 60863)

Prior to JADE release 7.0.09, 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 non-volatile write-cache is not enabled on 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.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

DynaDictionary Methods (PAR 58349)

The **DynaDictionary** class can now contain the following methods, which enable you to specify in the **sortOrder** parameter the locale identifier of the locale used for comparison. A value of zero (**0**) indicates the **Binary** sort order.

- The **addExternalKeyWithSortOrder** method, which adds an external key specification to a dynamic dictionary, has the following signature.

```
addExternalKeyWithSortOrder(keyType:         Type;
                            keyLength:       Integer;
                            descending:      Boolean;
                            caseInsensitive: Boolean;
                            sortOrder:       Integer);
```

- The **addMemberKeyWithSortOrder** method, which adds a member key specification to a dynamic dictionary, has the following signature.

```
addMemberKeyWithSortOrder(propertyName:    String;
                          descending:      Boolean;
                          caseInsensitive: Boolean;
                          sortOrder:       Integer);
```

For details, see Volume 1 (that is, **EncycloSys1.pdf**) of the *JADE Encyclopaedia of Classes*.

Dynamic Clusters and Properties

There are now two types of dynamic properties: runtime and design-time.

- Runtime dynamic properties are essentially unchanged from JADE 7.0, with the only difference being that the color of property text (magenta) in the hierarchy browser. Runtime dynamic properties cannot be maintained (created, changed, or deleted) in the JADE development environment.

  For details, see "Runtime Dynamic Properties", in Chapter 21 of the *JADE Developer's Reference*.

- Design-time dynamic properties are new in JADE 7.1. Design-time dynamic properties are the same as static properties, except that they:

  - Are stored in a binary large object (blob) associated with the owner dynamic cluster.
  - Generally do not require a reorganization to add, change, or delete them.
  - Are slightly slower to access (similar to static blob access).
  - Can be maintained only from the JADE development environment or the Schema Load utility.
  - Are displayed in the development environment hierarchy browser with a default color of maroon.

  For details, see "Dynamic Clusters and Properties", in Chapter 4 of the *JADE Development Environment User's Guide*.

The JADE development environment supports dynamic properties as follows.

- Dynamic property display

  The **Window** sheet of the Preferences or JADE Installation Preferences dialog now enables you to change the color with which the text of dynamic design-time and runtime property names are displayed in the Properties List of a hierarchy browser. By default, design-time dynamic properties are displayed in maroon and runtime dynamic properties in magenta.

- Dynamic property cluster maintenance

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

The Classes menu now provides the **Dynamic Property Clusters** command, which accesses the Dynamic Clusters dialog that enables you to view the clusters defined for the class, add a new cluster, delete an existing cluster, and view a list of the properties defined in a cluster. (This command is enabled only when the selected class is allowed dynamic clusters.)

- Design-time dynamic property maintenance

  The Properties menu now provides the **Add Dynamic Attribute** and **Add Dynamic Reference** commands.

Entity Name Maximum Length (NFS 61179)

The 30-character limit for schema entity names has increased to the maximum of 100 characters, to ensure, for example, that class, method, and property names integrated with .NET are unique. (See also "String Arrays and Dictionary Keys" under "Jade 7.1 Changes that May Affect Your Existing Schemas".)

The names of the following entities now have a maximum length of 100 characters.

- ActiveX component
- Ad hoc index
- Application
- C# exposure
- Class
- Constant (global, class, or local)
- Dynamic property
- Dynamic property cluster
- Form
- HTML document
- Identifier in a method (that is, a property or method of the receiver, a parameter, a local variable, a constant, or a class or primitive type name)
- Imported .NET object
- Interface
- **JadeDynamicObject** property
- Map file
- Mask queue in the messaging framework
- Menu item
- Method
- Method view
- Non-GUI client application service
- Package (imported *and* exported)
- Primitive type
- Property
- Relational Population Service (RPS) mapping
- Schema

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- Server node
- Translatable string
- Web service application
- Web service exposure definition
- XAML document

In addition, the:

- **SystemLimits** category now provides the **Max_Identifier_Length** global constant, which has an **Integer** value of **100**.
- Control name maximum length is now 86 characters.

External Function and External Method Entry Point Length (PAR 61412)

External function and external method entry points can now be up to 255 characters in length. If they are longer than 100 characters, they must be enclosed in single quote (**'**) or double quote (**""**) characters or exception 6007 (*Token too long*) is raised.

The schema extract process encloses all entry point names in quote characters.

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)*.

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.

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
```

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

JADE Debugger

This section describes the JADE debugger changes in this release.

Attaching the JADE Debugger to a Running Application (NFS 2413)

JADE now enables you to dynamically attach the JADE debugger to a running application, provided that the JADE development environment is also being run by that client.

The JADE User Interrupt menu now provides the **Attach Debugger** command for each application, which enables you to initiate the JADE debugger. The application execution halts in the debugger at the next debugged method logic statement.

By default, only methods that begin execution after the JADE debugger is attached can be debugged, because the JADE Interpreter executes methods using two different execution engines: normal execution and debugger execution. Normal execution is significantly faster and cannot be debugged. As a result, any methods already in execution will not halt on breakpoints or allow debug break and step. The exceptions to this are:

- Running the application in Run in Debug Ready mode. The JADE Run Application dialog now contains the **Run in Debug Ready Mode** check box, which runs the application in debug mode but without the debugger being initiated when this is checked. Attaching to that application then allows debugging of any method already on the execution stack. (Note that if the application is idle and no modal dialogs are displayed, attaching the debugger is equivalent to running in Debug Ready mode.)
- If the debugger was attached to an application, the debugger is closed and then subsequently re-attached.

If the application is in a loop, attaching the debugger breaks only if the application is debug-ready or if the logic begins execution of another user-defined method after the attach action.

If you use the JADE User Interrupt menu **Break Application** command and the logic being executed is debug-ready, it causes a debugger break if logic is being executed.

debugApplicationWithParameter Method (NFS 61035)

The **Application** class can now contain the **debugApplicationWithParameter** method, which has the following signature.

```
debugApplicationWithParameter(schemaName:      String;
                              applicationName: String;
                              passedObject:    Object);
```

Executing this method initiates the application specified in the **applicationName** parameter requested application in JADE debug mode, passing the value specified in the **passedObject** parameter to the **initialize** method defined in the application. The **initialize** method must expect an object parameter; otherwise an exception is raised.

The JADE debugger stops on the first logic statement executed in the user application.

The conditions that apply to the **Application** class **debugApplication** method apply to the method; that is:

- To debug the application, the JADE development environment must be running.
- You must have a defined user profile in the JADE development environment.
- The schema specified in the **schemaName** parameter must exist.
- The application must be defined in that schema or a superschema.

Displaying the Current Exception Handler Stack (NFS 46856)

The View menu of the JADE debugger now contains the **Show Exception Handler Stack** command, which displays the exception handlers armed by the receiving process in the current node.

The exception handler stack displays:

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- The name of the application that is arming the exceptions.
- For each armed exception handler, the following information is displayed.
  - The stack depth (the handlers at the top of the stack are invoked first)
  - Whether the exception handler is a global or a local exception handler
  - The name of the class and method that armed the exception (this applies to local exception handlers only)
  - The name of the exception class
  - The name of the class and method of the exception handler method
  - The name of the class of the exception handler receiver

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 "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.

CustomRuntimeUpgrade Parameter (NFS 61726)

The [JadeAppServer] section only of the JADE initialization file on each application server can now contain the **CustomRuntimeUpgrade** parameter, which defaults to **false**, indicating that the normal thin client upgrade process occurs. The **CustomRuntimeUpgrade** parameter is read when the application server starts up.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Set this parameter to **true** to allow the downloading of an alternative environment-type set of thin client binaries; for example, when 64-bit thin clients have been installed but you want to leave specific clients running 32-bit versions. When this parameter is set to **true** and a thin client connects to the application server, the environment-type value of each host specified in a **CustomRuntimeUpgrade[*computer-name*]** parameter variant in the [JadeAppServer] section of the initialization file is checked. (These computer-name-specific parameter variants are ignored when the **CustomRuntimeUpgrade** parameter is set to **false**.)

The value of the **CustomRuntimeUpgrade[*computer-name*]** parameter variant is the environment-type (that is, the *<hardware-type>-<vendor>--<operating-system-version>*-ansi|unicode) set of binaries; for example:

```
[JadeAppServer]
CustomRuntimeUpgrade=true
CustomRuntimeUpgrade[alfreda]=i686-msoft-win32-ansi
CustomRuntimeUpgrade[cecilb]=x64-msoft-win64-unicode
CustomRuntimeUpgrade[jemimac]=armv4i-msoft-wce50-unicode
```

Note Setting the required environment type for a computer name overrides the use of the **UpgradeRuntimeTo64bit** parameter for that specific client.

The **CustomRuntimeUpgrade[*computer-name*]** parameter variant allows control on a per-thin client basis which thin clients upgrade to an alternative type of binaries. (For details about supported architecture and build-type combinations, see "Overview", in Appendix B of the *JADE Thin Client Guide*.)

Tip The *[computer-name*] value is the name of the thin client machine passed to the application server. This name, which is case-sensitive, can be seen in the application server monitor window or in the **jommsg.log** file in the "Thin client connected and signed on: host-name (IP-address) schema..." message.

For more details, see "Upgrading 32-bit Thin Client Binaries to 64-bit (NFS 61726)".

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 Ad Hoc Index [JadeAdHocIndex] Section

The JADE initialization file can now contain the [JadeAdHocIndex] section, which enables you to specify options for the worker applications that build, drop, and delete an ad hoc index, and for the controller application that starts worker applications when there is an ad hoc index maintenance operation to be performed.

For details about the **BuildCommitPeriod** and **MaxBuildWorkers** parameters, see "JADE Ad Hoc Index Section [JadeAdHocIndex]", in the *JADE Initialization File Reference*.

JADE Application Server Upgrade Parameters

The [JadeAppServer] section of the JADE initialization file on each application server now provides the following parameters, which enable you to upgrade 32-bit presentation (thin) client binaries to 64-bit binaries when the client is using a 64-bit operating system.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

In addition, you can control the specific architecture of the binaries downloaded to a specific presentation client.

- **UpgradeRuntimeTo64bit**
- **CustomRuntimeUpgrade**
- **CustomRuntimeUpgrade[*computer-name*]**

For details, see "Upgrading 32-bit Thin Client Binaries to 64-bit (NFS 61726)" under "Thin Client Downloads" and "Upgrading a 32-bit Presentation Client Connecting to a 64-bit Application Server", elsewhere in this document.

JADE Sentinel Configuration

The **EnableSentinel** parameter in the [FaultHandling] section and the **ExcludeDiskCache** parameter in the [JadeSentinel] section of the JADE initialization file enable you to configure the **jadesentinel.exe** program.

The JADE Sentinel runs as a debugger attached to its parent process, looks for specific exceptions, performs out-of-process process dumps, and terminates the program, as appropriate. For more details, see "Monitoring Your Active Process Using JADE Sentinel", later in this document.

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)

Parameter Value 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).

UpgradeRuntimeTo64bit Parameter (NFS 61726)

The [JadeAppServer] section only of the JADE initialization file on each application server can now contain the **UpgradeRuntimeTo64bit** parameter, which defaults to **false**, indicating that the normal thin client upgrade process occurs. The **UpgradeRuntimeTo64bit** parameter is read when the application server starts up. Set this parameter to **true** to allow the downloading of 64-bit binaries to clients that are running 32-bit JADE binaries on a 64-bit operating system (WoW64). When this parameter is **true** and a thin client connects to the application server, if the client is running 32-bit binaries on a 64-bit operating system, a download of the JADE 64-bit binaries will occur instead of 32-bit binaries.

The exception to this is if the value of the **CustomRuntimeUpgrade** parameter is **true** and the specific client connecting has a **CustomRuntimeUpgrade** parameter indicating the environment type of the binaries required, in which case the **UpgradeRuntimeTo64bit** parameter is ignored for this client.

Note For the **UpgradeRuntimeTo64bit** parameter to take effect, the presentation client must first be upgraded with the version of JADE that supports this parameter. The previous thin client versions do not pass their operating system type in the initial handshake that is used to determine whether a download may be required. As a result, to get clients upgraded to 64-bit requires two downloads: one to update to JADE 7.1, and a second to upgrade to 64-bit.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

For more details, see "Upgrading 32-bit Thin Client Binaries to 64-bit (NFS 61726)".

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.

Detecting a Collection Object with Multiple Keys (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.

JADE Remote Node Access Utility

This section describes the JADE Remote Node Access utility (**jadrap**) changes in this release.

See also "JADE Remote Node Access (PAR 60364)", under "JADE 7.1 Changes that May Affect Your Existing Schemas".

Menu Item Changes (PAR 57911)

The Network menu and the **Threads** command in the Options menu have been removed from the Remote Node Access utility window. (Values are updated by using the appropriate parameters in the [JadeServer] section of the JADE initialization file.)

In addition, the **Show Configuration** and **Show Connections** commands have been added to the File menu, to enable you to display server network and thread configuration details and current client connections information, respectively. (You can clear these details, by selecting the **Clear Display** command in the File menu.)

Termination Message Box

The JADE Remote Node Access utility has an optional new style TERMINATION message box, which is controlled by using the **TerminationMsgbox** parameter with a value of **TryAgain** in the [JadeServer] section of the JADE initialization file. If you do not specify this value or the parameter has a **<default>** value, the existing behavior applies (that is, the message box displays **Yes** and **No** buttons only).

When the **TerminationMsgbox** parameter has a value of **TryAgain**, the message box displays the following buttons.

- **Cancel**

  Closes the message box.

- **Try Again**

  Rechecks to see if there are any active nodes. If there are no active nodes, **jadrap** terminates. If there are active nodes, the message box is redisplayed.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

- **Continue**

  Shuts down **jadrap**, even if active nodes are attached.

JADE Version Information Utility

This section describes the JADE Version Information utility changes in this release.

Commit Limit Values (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.

datmask and sysmask Optional Parameters (PAR 61797)

The JADE Version Information utility (**jverinfo**) can now contain the optional **datmask** and **sysmask** parameters if you want to specify alternative file mask patterns for user data files and system files, respectively.

The **jverinfo** syntax is now as follows.

```
jverinfo [binpath=binary-directory-name] [binmask=binary-file-type]
         [datpath=user-data-directory] [datmask=user-data-file-type]
         [out=output-file-name|stdout] [syspath=system-file-directory]
         [sysmask=system-file-type] [noThinClients] [noExtraDirs]|
         [extraDirs=[path][directory-name]]
```

The default user data file type is ***.dat** and the default system file type is **_*.bin**.

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.

Monitoring your Active Process Using JADE Sentinel

JADE now provides JADE Sentinel (**jadesentinel.exe**), which monitors your active process. If, for any reason, a fault or problem occurs in this active process, JADE Sentinel takes a diagnostic copy in case the main process is unable to do so itself. No matter what happened to the process, JADE Sentinel ensures that you still get all of the information you need.

The **jadesentinel.exe** is started by any JADE-built or supplied executable. It is a small executable that links to **jomos.dll**, in which all of the work occurs. It runs as a debugger attached to its parent process, looks for specific exceptions, performs out-of-process process dumps, and terminates the program, as appropriate.

The **jadesentinel.exe** is started automatically when the JADE-built executable starts. If it is necessary to start **jadesentinel.exe** manually, it has the following command line syntax.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

```
jadesentinel.exe pid=pid ini=path\jade-initialization-name [prefix=executable-name]
```

The **pid** value specifies the process identifier of the process to monitor (that is, the parent process), the **ini** value specifies the path and name of the JADE initialization file, and the optional **prefix** value specifies the prefix to put on the dump file (for example, **jadrap**, **jade**, **jadload**, and so on).

The [FaultHandling] section of the JADE initialization file contains the following parameter.

```
EnableSentinel=<default>
```

The **EnableSentinel** parameter, which is **true** by default, is read when a JADE executable starts up. This controls starting the **jadesentinel.exe** program that monitors the executing parent, and when required to do so, does a process dump. (Process dumps are output to the directory specified in the **ProcessDumpDirectory** parameter of the [FaultHandling] section of the JADE initialization file.)

The JADE initialization file now contains the [JadeSentinel] section, which provides the following parameter.

```
ExcludeDiskCache=<default>
```

The **ExcludeDiskCache** parameter, which is **true** by default, is read when the process dump is about to occur, to exclude disk cache from the process dump.

Note You should set these values to **false** only when requested to do so by JADE Support.

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.

OpenSSL Library

This section describes the OpenSSL library changes in this release.

Application Server OpenSSL Cipher Suite (NFS 61541)

In earlier JADE releases, only the **TLSv1** protocol was supported.

JADE now supports TLS 1.1 and TLS 1.2. You can now specify the **TLSv1.1** or **TLSv1.2** value in the **SSLMethodName** parameters in the [JadeThinClient] and [JadeAppServer] sections of the JADE initialization file. In addition, the **methodType** property in the **JadeSSLContext** class now supports the new **MethodTLSv1_1** and **MethodTLSv1_2** constants defined in the **JadeSSLContext** class.

Library Version Upgrade (PAR 61368)

The OpenSSL library has been upgraded to version 1.0.1h.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Recompiling All Methods (PAR 61988)

The **jadclient** executable enables you to automate the running of the **JadeRecompileAllMethods** non-GUI client application to force a recompile of *all* methods; that is, methods in error, methods not compiled, and successfully compiled methods.

The **JadeRecompileAllMethods** application now has the following syntax.

```
jadclient path=database-path
          [ini=jade-initialization-file]
          schema=RootSchema
          app=JadeRecompileAllMethods
          endJade
          [command-line-arguments]
```

The list of command line arguments that you can define after the **endJade** parameter in ***command*=*value*;** format are as follows.

- allMethods=true|false
- inError=true|false
- notCompiled=true|false

The following example recompiles only those methods that are in error.

```
jadclient.exe path=c:\jade\system ini=c:\jade\system\testjade.ini schema=RootSchema
app=JadeRecompileAllMethods endJade notCompiled=false
```

The default action is to recompile all methods that are in error or are not compiled (that is, the **inError** and **notCompiled** arguments default to **true**).

As the **allMethods** argument defaults to **false**, specifying **allMethods=true** on the command line now allows all methods to be recompiled, including methods currently compiled without error, methods that have been compiled but are in error, and methods that have not been compiled.

The **allMethods=true** argument overrides the **inError** and **notCompiled** arguments.

References Browser (NFS 61215)

The References Browser for class, property, method, class constant, global constant, and translatable string entities now includes the number of references to the entity when there are two or more references to that entity; for example:

- `MySchema::ListTest::listBox1_displayRow (2)`

  This indicates that there are two references to the entity in the method **ListTest::listBox1_displayRow**.

- `MySchema::ListTest::listBox2_displayEntry`

  This indicates that there is one reference only to the method (that is, there is no number in parentheses).

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.

REST-Based Web Services

Earlier JADE releases supported only the Simple Object Access Protocol (SOAP) and WSDL-based Web services.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

JADE now implements the Representational State Transfer (REST) stateless architecture style as a simpler alternative to SOAP Web services. Mainstream Web 2.0 service providers such as Google, Salesforce, and Facebook have endorsed this easier-to-use, resource-oriented model to expose their services. REST-based Web services, implemented using HTTP, offer a light-weight alternative to the Web services available in earlier releases.

REST works with resources that are identified with a uniform resource identifier (URI). A resource represents a static or dynamically-generated Web page. REST resources are named with nouns as part of the URI rather than verbs; for example, **/customers** rather than **/getCustomers**.

To use REST services, a client sends an HTTP request using the **GET**, **POST**, **PUT**, or **DELETE** verb.

The traditional HTTP error messages (for example, **200 - OK** and **404 - Not found**) can be used to indicate whether a request is successful. If a request is successful, information can be returned in Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

Session handling is not performed, so there is no timeout of connections. Additionally, information is not retained between requests from a client. If that was required, it would need to be provided by the application developer.

For details, see Chapter 11, "Building Web Services Applications", of the *JADE Developer's Reference*. See also the **JadeRestService** class, in volume 1 of the *JADE Encyclopaedia of Classes*.

Snapshot Database Recovery

You can now recover a database backup that was captured and restored by a third-party snapshot tool; for example, VMWare or Amazon Elastic Block Store (EBS) snapshots.

For details about external third-party snapshot backups, see "Non-JADE Backups" in the *Developing a Backup Strategy White Paper*.

The third-party tools are responsible for detecting any errors that were raised during the movement of data. A restored snapshot is considered to be database data exactly as it was copied when the snapshot was created. No JADE backup or restore verification checks can be performed.

A roll-forward recovery must be performed to establish database integrity.

A snapshot recovery requires the snapshot processing to happen after the snap begin call, completes before the snap end call, and all journals generated between these two points are required to restore file-level integrity.

When performing a snapshot recovery:

- If a database is accidentally shut down during snapshot processing, do not restart it until the snapshot processing is complete. When restarted, the database will perform a snapshot recovery.
- Ensure that changing the access mode to **DB_SNAPSHOT** occurs before snapshot processing begins.
- Ensure that snapshot processing completes before restoring access mode to **DB_DEFAULT**.

The **pMode** parameter of the **jomChangeAccessMode** Application Programming Interface (API) call, documented in Chapter 3 of the *JADE Object Manager Guide*, now has the **DB_SNAPSHOT**, **DB_ARCHIVE**, and **DB_ DEFAULT** mode constants, which conditions the database for external snapshot backup, quiesces the database for archive backup, and restores the database to initial mode and usage values, respectively.

The **JadeDatabaseAdmin** class now provides the **Mode_Exclusive** constant (which requests exclusive access to the database) and the **Mode_Snapshot** constant, which can be specified in the mode parameter of the **changeDbAccessMode** method to condition the database for external snapshot backup.

The **jdbadmin** program enables you to specify **StartSnapshot** and **EndSnapshot** database actions, to condition the database for recovery from a non-JADE backup and to take the database out of snapshot mode, respectively. (For details, see "StartSnapshot" and "EndSnapshot", respectively, in Chapter 2 of the *JADE Database Administration Guide*.)

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

StringUtf8Array Elements (PAR 60605)

The maximum length of elements in a user-defined **StringUtf8Array** is 8,000 UTF8 characters.

System Class isValidProcess Method (PAR 60191, 61755)

The **System** class now provides the **isValidProcess** method, which has the following signature.

```
isValidProcess(process: Process): Boolean;
```

This method returns **true** if the process specified by the **process** parameter represents a signed-on application. A **Process** instance without a corresponding signed-on application can exist under certain circumstances; for example, if an error occurs during process sign off that prevents the **Process** instance from being deleted. The **isValidProcess** enables you to identify these *zombie* **Process** instances.

When a zombie process is encountered in a monitor operation, the instance is deleted; for example, an interrupt or force off user, call stack request, and so on.

Thin Client Downloads

This section describes the thin client download changes in this release.

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 **jaddinst.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.

Upgrading 32-bit Thin Client Binaries to 64-bit (NFS 61726)

JADE enables you to upgrade 32-bit presentation (thin) client binaries to 64-bit binaries when the client is using a 64-bit operating system. In addition, you can control the specific architecture of the binaries downloaded to a specific presentation client.

Note Visual C++ runtimes are always upgraded (that is, the **x64** version is installed) as part of the upgrade process.

The [JadeAppServer] section only of the JADE initialization file on each application server can now contain the following parameters, which default to **false**, and that are read when the application server starts up.

- **UpgradeRuntimeTo64bit**
- **CustomRuntimeUpgrade**

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

When the **UpgradeRuntimeTo64bit** parameter is set to **true**, presentation clients that are running a 32-bit version of **jade.exe** on a 64-bit operating system cause a download and installation of 64-bit binaries to that client (with the exceptions described later in this section). The default value of **false** indicates that 32-bit version presentation clients continue to download 32-bit binaries when required. Presentation clients that are running a 32-bit version of **jade.exe** on a 32-bit operating system continue to download 32-bit binaries when required.

When the **CustomRuntimeUpgrade** parameter is set to **true**, each connecting presentation client looks for a **CustomRuntimeUpgrade[*computer-name*]** variant of this parameter in the [JadeAppServer] section based on the machine name of the client machine name. This parameter variant, which specifies the architecture of the binaries to be downloaded to the client when required, has the following format.

```
CustomRuntimeUpgrade[computer-name]=architecture-name
```

The **[*computer-name*]** value is the name of the thin client machine passed to the application server. This name, which is case-sensitive, can be seen in the application server monitor window or in the **jommsg.log** in the "Thin client connected and signed on: *host-name* (*IP-address*) *schema*..." message. For details about supported architecture and build-type combinations, see "Overview", in Appendix B of the *JADE Thin Client Guide*.

The following example would cause 32-bit binaries to be used for any required download to the presentation client with a machine name of **pc101a**.

```
CustomRuntimeUpgrade[pc101a]=i686-msoft-win32-ansi
```

If the value of the **CustomRuntimeUpgrade** parameter is **true** and a corresponding parameter is found for the presentation client machine name, the **UpgradeRuntimeTo64bit** parameter is ignored.

The **CustomRuntimeUpgrade[*computer-name*]** parameter allows control on a per-thin client basis which thin clients upgrade to an alternative type of binaries.

If the client is not running a JADE version that is using Visual Studio 2005 runtime libraries, the JADE thin client download will not revert to a version that uses Visual Studio 2005 runtime libraries.

Note The **UpgradeRuntimeTo64bit** parameter applies only after the *next* thin client download has occurred, because any currently deployed **jade.exe** currently does not communicate the operating system type to the application server. To get 32-bit clients to upgrade to 64-bit binaries therefore requires two download and install processes: one to upgrade the **jade.exe** so that it will identify the operating system type to the application server, and the second to upgrade from 32-bit to 64-bit binaries.

Transient Leak Detection (NFS 60726)

The **Search for creates of selected class in selected schema** option button on the Find Possible Transient Create Leaks dialog (accessed from the **Find Possible Transient Leaks** command in the Schema menu) has been changed as follows.

- The caption for the option button is now **Search for creates of selected class in all schemas**.
- The **Select the schema above to get its list of classes** combo box beneath the **Search for creates of selected class in all schemas** option button enables you to select the class defined in the schema selected in the combo box of the **Search selected schema for all creates** option.

When you click the **Search** button after selecting the **Search for creates of selected class in all schemas** option button and selecting a class in the **Select the schema above to get its list of classes** combo box, all creates of that class in *all* schemas are searched; that is, the search is performed in the schema in which the class is defined and all subschemas. If the class is a **RootSchema** class, all schemas are searched.

In earlier releases, creates were searched for in the selected schema only.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved

Unaudited Database Files and Partitions

When batch-loading a database file or when generating ad hoc indexes, for example, there is significant auditing overhead. (See also "Ad Hoc Indexes", earlier in this document.) Until the index build is complete, the database file or partition is known to be building (application preserved state) and if the build is restarted, the file or partition is first dropped.

From this release, you can update database files and partitions with auditing disabled, to eliminate journal disk space use and I/O overhead when loading data.

When auditing is re-enabled for a file or partition, a copy of the file or partition is compressed by default, and inserted into the journal. During database roll-forward or replay, the file at the database location is replaced by the file reconstructed from the journal. Subsequent audited updates therefore replay correctly.

Caution Disable the auditing of database files and partitions only when restart recovery is not required.

Unaudited operations cannot be used when you have RPS secondaries, as this would result in SQL inconsistencies.

The following table summarizes the methods used to manage unaudited operations.

Class	Method	Description
DbFile	disableAuditing	Disables the auditing associated with object operations performed against the file
DbFile	drop	Removes the file and marks it as deleted
DbFile	enableAuditing	Re-enables the auditing associated with object operations performed against the file
DbFile	isAuditing	Returns **true** if auditing associated with object operations performed against the file is enabled and returns **false** when auditing has been disabled
JadeDatabaseAdmin	doQuietpoint	Attempts to establish a database quiet point
JadeDbFilePartition	disableAuditing	Disables auditing associated with object operations performed against the partition
JadeDbFilePartition	drop	Removes the partition and marks it as deleted
JadeDbFilePartition	enableAuditing	Re-enables the auditing associated with object operations performed against the partition
JadeDbFilePartition	isAuditing	Returns **true** if auditing associated with object operations performed against the partition is enabled and returns **false** when auditing has been disabled

In addition, the **DbFile** class now provides the **EnableAudit_NoCompress** class constant.

For details, see Volume 1 of the *JADE Encyclopaedia of Classes*.

Copyright 2015, Jade Software Corporation Ltd. All rights reserved