



J A D E TM

Upgrading to JADE Consolidated Release Information

VERSION 6.3.09



Copyright © 2011
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 © 2011 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

JADE Release Support	6
Deimplementations and Deprecations in JADE 6.3	6
JADE Portable Graphical User Interface (GUI) Client	6
PKWare Compression.....	6
Real[10] Parameters in External Function Calls.....	7
RPS SQL Script Execution.....	7
Advanced Notice of JADE 7.0 Changes.....	7
Compact JADE Single User Mode	7
JADE on Linux	7
Exposure Deimplementation.....	7
JADE 7.0 Runtime Binaries.....	8
Upgrading a 32-Bit Presentation Client Connecting to a 7.0 Application Server.....	8
JADE Dump and Load Utility Deimplementation.....	9
Platform Requirements for JADE 7.0.....	9
Rational Rose Deimplementation	9
Accessing Details about Faults Fixed in Releases	10
How to Locate PARs Fixed in a Specific Release.....	10
Upgrading to JADE 6.3.09 from a JADE 6.2 or Earlier JADE 6.3 Release	11
Upgrading to JADE 6.3 from a Windows JADE 6.2 or Earlier JADE 6.3 Release.....	11
Running Two Windows Releases of JADE on the Same Workstation.....	14
Upgrading to JADE 6.3 from a Linux JADE 6.2 or Earlier JADE 6.3 Release	14
JADE Thin Client Upgrade	15
Upgrading a Synchronized Database Environment (SDE).....	16
Upgrading an RPS Node from JADE 6.2 to 6.3	16
Upgrade Validation.....	17
Reorganization.....	17
Hot Fix Releases	17
Changes in JADE Release 6.3.09	18
Adding an Existing Property or Method to an RPS Table (NFS 55461, NFS 57061).....	18
Changing the Size of Text in a JadeTextEdit Window (NFS 56953).....	18
Date Parsing Routines for Two-Digit Years (PAR 56832)	18
Dock Controls (PAR 56859)	19
External Database Access (NFS 30972).....	19
JADE Debugger (PAR 56580)	19
JadeHTTPConnection Class.....	19
JadeHTTPConnection Class Constants.....	20
JadeHTTPConnection Properties.....	21

Changes in JADE Release 6.3.09, continued

JadeHTTPConnection Methods	23
configureProxy	23
getHeader	24
getHttpPage	24
getHttpPageBinary	24
open	25
queryConnectionIsClose	25
queryContentLength	26
queryContentType	26
queryDate	26
queryInfo	26
queryStatusCode	27
readBody	27
sendRequest	27
setAccept	27
setContenttype	28
setKeepAlive	28
setReferer	28
setReload	28
setSoapAction	29
setUserAgent	29
JadeProfiler::report Method Output (PAR 55984)	29
Journal Rate Analysis Sampling (PAR 57125)	29
Portable GUI Thin Client Upgrade (PAR 56161)	29
Recompiling Methods after Upgrading from a 6.2 to a 6.3 Release (PAR 56522)	30

Changes in JADE Release 6.3.08**31**

Recompiling Methods after Upgrading from Release 6.2 to 6.3.07 (PAR 55301)	31
.NET Requirements	31
Changing a Control Type to a Superclass or Subclass (PAR 54738)	32
Class Lifetimes	32
Persistence when Extracting and Loading Schemas (NFS 55284, PAR 53319)	32
Customized Deployment	33
Customized Deployment Upgrade (PAR 53457)	33
Setup.ini File	33
Dates Entered in Text Boxes	33
Install Shield Version	33
ODBC User Method Exception Logging (PAR 55105, 52344)	33
Relational Population Service (RPS)	33
Many-to-Many Junction Tables Naming Convention (PAR 55186)	33
Mapping Many-to-Many Collections (PAR 55352)	34
Removing a Method in a Delta from an Exported Interface (PAR 55256)	34
Report Writer	34
Changing the Feature or Path Name of an Alias (NFS 54335, NFS 54664)	34
Removing Invalid View Features (NFS 38293)	34

Upgrading to JADE Release 6.3.09

This document covers the following topics.

- [JADE Release Support](#)
 - [Deimplementations and Deprecations in JADE 6.3](#)
 - [Advanced Notice of JADE 7.0 Changes](#)
- [Accessing Details about Faults Fixed in Releases](#)
- [Upgrading to JADE 6.3.09 from a JADE 6.2 or Earlier JADE 6.3 Release](#)
 - [Upgrading to JADE 6.3 from a Windows JADE 6.2 or Earlier JADE 6.3 Release](#)
 - [Upgrading to JADE 6.3 from a Linux JADE 6.2 or Earlier JADE 6.3 Release](#)
 - [JADE Thin Client Upgrade](#)
 - [Upgrading a Synchronized Database Environment \(SDE\)](#)
 - [Upgrading an RPS Node from JADE 6.2 to 6.3](#)
 - [Upgrade Validation](#)
 - [Reorganization](#)
 - [Hot Fix Releases](#)
- [Changes in JADE Release 6.3.09](#)
- [Changes in JADE Release 6.3.08](#)

Refer to [RelInfo6307.pdf](#) in your JADE **documentation** directory for details about JADE release 6.3.07, 6.3.06, 6.3.05, and 6.3.04 changes that may affect your JADE 6.2 existing schemas and changes in these JADE 6.3 releases. Refer to [RelInfo6303.pdf](#) in your JADE **documentation** directory for details about JADE release 6.3.03 changes that may affect your JADE 6.2 existing schemas and changes in JADE release 6.3.03.

Tip For information about using Acrobat Reader to view JADE documents, see “[JADE Product Information Library in Portable Document Format](#)”, in Chapter 2 of your *JADE Development Environment User’s Guide* in your JADE **documentation** directory. The *JADE Product Information Library* document ([JADE.pdf](#)) provides a summary of JADE product information library documents and navigation to them.

If you want to develop your own installation process:

- For Windows, the JADE install and upgrade steps are documented in the **ReadmeInstallSteps.txt** file in the **\documentation** directory.
- For Linux, the steps are as executed in the **pre_i** and **post_i** scripts in the **bin** directory in the **/opt/jade** subdirectory for the release.

To customize the deployment upgrade on Windows, see Appendix A, “[Customizing the Deployment Upgrade Process](#)”, in your *JADE Runtime Application Guide*.

JADE Release Support

Support for JADE 6.2.20 (the final JADE 6.2 release) will cease in October 2012.

As earlier JADE 6.3 release information advised, support for JADE 6.1.15 (the final JADE 6.1 release) ceased in October 2010. All versions of JADE 6.1 are no longer supported.

For details about the JADE release policy, go to:

http://www.jade.co.nz/downloads/jade/JADE_ReleasePolicy.pdf

For details about the JADE release schedule, go to:

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

Microsoft support for Windows 2000 and Windows XP Service Pack 2 (SP2) ended on July 13, 2010. If you are still using one of these operating systems, you should migrate to a later operating system version, as JADE is no longer supported on Windows 2000 and Windows XP SP2 now that Microsoft support for them has ended. (Windows XP SP3 continues to be supported. In addition, JADE is now supported on Windows 7.)

From JADE 7.0, JADE will no longer be supported on Windows Server 2003.

Deimplementations and Deprecations in JADE 6.3

This section contains the deimplementations and deprecations in JADE 6.3.

JADE Portable Graphical User Interface (GUI) Client

The JADE portable GUI client is deprecated for both Linux and Windows platforms.

PKWare Compression

As notified in JADE 6.2 release information, PKWare compression is not supported in this release.

Before upgrading to JADE 6.3:

1. Change existing code in your JADE 6.2 applications to use the **compressToBinary** method of the **Binary**, **String**, and **StringUtf8** primitive types and the **uncompressToBinary**, **uncompressToString**, and **uncompressToStringUtf8** methods of the **Binary** primitive type.
2. If you have persistent data that has been compressed using the PKWare compression libraries, update that data by uncompressing it using the appropriate **uncompress**, **uncompressString**, or **uncompressStringUtf8** decompression method, and then recompress it using the **compressToBinary** method using a compression option from the **Binary** primitive **Compression_Zlib**, **Compression_ZLibFast**, or **Compression_ZLibSmall** constant.

Note JADE strongly recommends that you make the changes necessary to transition to the use of zlib compression.

However, if you elect to continue to use PKWare compression, you should be aware that before you can upgrade to a 64-bit version of JADE, coding changes to use new methods and data recompression are necessary. For details about using PKWare compression in 32-bit editions of JADE 6.3, see “Compression and Decompression Methods Deimplemented”, in the JADE 6.3.03 Release Information document (that is, [RelInfo6303.pdf](#)).

Real[10] Parameters in External Function Calls

As notified in JADE 6.2 release information, external function calls with **Real[10]** parameters are no longer supported.

The upgrade validation process checks external functions for **Real[10]** parameters, logs any detected usages, and the upgrade fails. You must change the usages to **Real[8]** and re-run the validation.

RPS SQL Script Execution

In this release, **sqlcmd** has replaced the Open Database Connectivity (ODBC) interface as the default mechanism for SQL script execution on an RPS node.

SQL scripts are used to create or alter table definitions and to load data. These scripts can be executed from the RPS Manager utility or automatically by the **Datapump** application.

The SQL Server **sqlcmd** utility is the preferred mechanism for SQL script execution. In order to use **sqlcmd**, it must be installed on the machine hosting the RPS node. The SQL Server instance name is specified from the RPS manager node configuration dialog.

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

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

Use of the ODBC interface for script execution is supported in JADE release 6.3. However, it will be deimplemented in JADE 7.0.

Advanced Notice of JADE 7.0 Changes

This section contains advanced notice of changes in JADE 7.0.

Compact JADE Single User Mode

In the JADE 7.0 product release, the Compact JADE single user node will not be available; that is, JADE 7.0 will not provide support for JADE databases running on Windows Mobile devices.

However, the Compact JADE thin client will continue to be available on these devices in JADE 7.0.

JADE on Linux

JADE 7.0 will support Windows only, which means that in the JADE 7.0 product release, Linux distributions (for both Red Hat and SUSE) will no longer be available.

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

Exposure Deimplementation

Notice is given of the intent to deimplement JADE's ActiveX exposure feature, starting from JADE release 7.0. This means that from JADE 7.0, the ActiveX exposure feature will become unavailable.

The ActiveX exposure enables you to expose selected features of your JADE system to application development tools such as Microsoft's Visual Basic and C++ languages through ActiveX technologies. JADE implements ActiveX generation through the wizard feature of Microsoft's Visual Studio 6. However, this product is no longer supported by Microsoft. In addition, new users of JADE may be unable to source a copy of Visual Studio 6.

In recent years, ActiveX technologies have been replaced by .NET. From JADE 6.3, you can now generate exposures using these .NET technologies. This provides a more modern, flexible, and easier to develop mechanism than that provided by ActiveX.

It is recommended that, where required, you re-write ActiveX exposures using the new JADE .NET exposure. For details, see [Chapter 19](#) of the *JADE Development Environment User's Guide*.

JADE 7.0 Runtime Binaries

Notice is given of the intent that JADE 7.0 will be built using Microsoft Visual Studio 2010, which will require the installation of appropriate C++ runtime binaries.

If you require 64-bit presentation clients, you must install Visual Studio 2010-based C++ runtime binaries.

For a level of backwards compatibility, system administrators will be able to choose to support 32-bit presentation clients using the same (Visual Studio 2005) C++ runtime binaries currently required by JADE 6.3.07, 6.3.08, and 6.3.09.

Upgrading a 32-Bit Presentation Client Connecting to a 7.0 Application Server

JADE 6.3 presentation clients connecting to a JADE 6.3 application server behave as they did in earlier JADE 6.3 releases, downloading the required files from the appropriate *jade-program-data-directory/i686-msoft-win32-ansi/download/* directory structure.

The JADE 7.0 release has two more sets of thin client binaries (32-bit ANSI and Unicode Visual Studio 2005 binaries, in addition to the expected sets of thin client 32-bit ANSI and Unicode Visual Studio 2010 and the 64-bit ANSI and Unicode Visual Studio 2010), 32-bit ANSI or Unicode binaries must be downloaded from different locations.

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

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

When a JADE 6.3 Windows 32-bit ANSI or Unicode presentation client connects to a JADE 7.0 application server, the presentation client is upgraded to version 7.0 Microsoft Visual Studio 2010 binaries, by default. However, if you want to support 32-bit presentation clients using the same (Visual Studio 2005) C++ runtime binaries that are required by JADE 6.3 releases, you must:

1. Install the Visual Studio 2005 presentation clients as an alternate architecture directory on each application server.
2. Set the value of the **NoCRTRuntimeUpgrade** parameter in the [JadeAppServer] section of the JADE initialization file on each application server to **true**. This parameter, which is new on the application server in JADE 7.0, determines whether the presentation client should be running the same Visual Studio C++ runtime binaries as the application server or it uses the Visual Studio C++ runtime binaries needed by JADE 6.3.

Setting this parameter to **true** in causes a JADE release 6.3 presentation client to download the alternate architecture binaries rather than the standard Visual Studio 2010-based presentation client files when it connects to a JADE 7.0 application server.

The following environment types are supported when the parameter value is **true**.

- i686-msoft-win32_vs2005-ansi
- i686-msoft-win32_vs2005-unicode

When the value of the **NoCRTRuntimeUpgrade** parameter is set to the default value of **false**, the normal presentation client download rules apply.

If a JADE 6.3 presentation client has an architecture type of **i686-msoft-win32-ansi**, it will be remapped to **i686-msoft-win32_vs2005-ansi**. Similarly, **i686-msoft-win32-unicode** is remapped to **i686-msoft-win32_vs2005-unicode**. When a download is required, a different [*environment-type*] section of the JADE initialization file is therefore used to locate the files to download.

In JADE 7.0, the 32-bit version of presentation client binaries must be installed on the application server, in the *jade-program-data-directory/i686-msoft-win32_vs2005-ansi/download/* directory structure.

If you require a 64-bit presentation client, you must manually install it. Once installed, it will automatically upgrade with 64-bit binaries.

JADE Dump and Load Utility Deimplementation

Notice is given of the intent to deimplement JADE's Dump and Load utility (**jddlutl**), starting from JADE release 7.0. This means that from JADE 7.0, Dump and Load utility feature will become unavailable.

Platform Requirements for JADE 7.0

JADE 7.0 will require the following.

- JADE presentation clients will require Windows XP Professional with SP3 or higher.
- Server and client nodes in JADE 7.0 will require Windows NT6 or later version (Vista or Server 2008, or better).
- Database server nodes in JADE 7.0 will require a 64-bit operating system.

For client nodes, the default configuration also requires 64-bit binaries. However, if 32-bit client nodes are required (for example, to allow for the use of 32-bit third-party DLLs), these can be used with some loss of performance and functionality.

Rational Rose Deimplementation

In the JADE 7.0 release, the JADE interface for the Rational Rose modeling tool will no longer be available.

If you want to use third-party design tools (for example, Enterprise Architect), you can use the JADE XMI interface to import the model into your database. (For details, see Chapter 26, "XML Metadata Interchange (XMI) Support", of the *JADE Developer's Reference*.)

Accessing Details about Faults Fixed in Releases

To access the complete documentation about the Product Anomaly Reports (PARs) fixed in this release, you can directly access **Parsys**, our Fault Managements and Customer Contact system. This enables you not only to view the progress of your own contacts but also to view all of the PARs fixed in a specific release.

If you have any queries about **Parsys**, please direct them to JADE Support in the first instance.

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

http://www.jade.co.nz/Jade6/jade6_parsys.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 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 PARs fixed in a specific release.

➤ To perform an advanced search

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

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

2. On the **Advanced Search Criteria III (Optional)** sheet:
 - In the **Closed** list box of the Releases group box, select the release whose fixed PARs you want to locate (for example, the **6.3.0** list item).
3. Click the **Search** button.

Upgrading to JADE 6.3.09 from a JADE 6.2 or Earlier JADE 6.3 Release

This section covers the following topics.

- [Upgrading to JADE 6.3 from a Windows JADE 6.2 or Earlier JADE 6.3 Release](#)
 - [Running Two Windows Releases of JADE on the Same Workstation](#)
- [Upgrading to JADE 6.3 from a Linux JADE 6.2 or Earlier JADE 6.3 Release](#)
- [JADE Thin Client Upgrade](#)
- [Upgrading a Synchronized Database Environment \(SDE\)](#)
- [Upgrading an RPS Node from JADE 6.2 to 6.3](#)
- [Upgrade Validation](#)
- [Reorganization](#)
- [Hot Fix Releases](#)

Upgrading to JADE 6.3 from a Windows JADE 6.2 or Earlier JADE 6.3 Release

You can upgrade from a 6.2 release to the 64-bit edition of JADE only if patch versioning has never been used in the JADE system. If patch versioning has ever been used, you must first upgrade to the 32-bit edition before you can upgrade to the 64-bit edition.

If you are upgrading to the 32-bit edition of JADE, the Microsoft Windows C++ 2005 Redistributable Package (x86) called **vc80_x86.exe** is required. If you are upgrading to the 64-bit edition, the Microsoft C++ 2008 SP1 Redistributable Package (x64) called **vc90_x64.exe** is required. (These executables are supplied on the JADE distribution media.)

Note Installing this Microsoft redistributable package requires administration privileges. If possible, deploy this package to all workstations *before* upgrading to JADE 6.3, using the appropriate techniques that allow for privileged installations.

If the required package is not already installed, it will be installed during the JADE installation.

The JADE Setup program enables you to upgrade your binary and database files to JADE 6.3 from a JADE 6.2 or earlier 6.3 release on Windows, by performing the following actions.

1. Ensure that your JADE environment is JADE release 6.2 or 6.3.
2. If you are upgrading from JADE 6.2, uninstall any **jadrap** database services (that is, JADE Remote Node Access services) that you want to upgrade to JADE 6.3.

This is required because of changes in the underlying registry entries that are expected or created in JADE 6.3.

3. If you are upgrading to JADE release 6.3 under Windows Server 2008, Windows 7, or Vista, ensure that you have the appropriate privileges or capabilities to install applications.

The configuration of Windows Server 2008, Windows 7, or Vista's User Account Control (UAC) and your current user account privileges may affect the behavior of the upgrade to JADE 6.3. For details about Windows Server 2008, Windows 7, or Vista UACs, standard user accounts, and administrator accounts, see:

<http://technet2.microsoft.com/WindowsVista/en/library/00d04415-2b2f-422c-b70e-b18ff918c2811033.msp?mfr=true>

[http://technet.microsoft.com/en-us/library/cc709691\(ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc709691(ws.10).aspx)

4. 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 6.3 and upgrade your existing data.

5. To start the JADE Setup program, invoke the **setup.exe** program from the **Jade63** release medium or execute the executable program downloaded from the JADE Web site.
6. The Microsoft Windows C++ Redistributable Package is installed, if not already 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 upgrade that you require. By default, the **Development** option is selected. If you do not want development files upgraded, select the **Application Runtime**, **Presentation Client**, **Jade Client**, or **SDS/RPS Database Server** option button, as required.

Note The **Custom** type applies only to a **Fresh Copy** installation type, and is not relevant when upgrading binary and database files. The **SDS/RPS Database Server** option applies only to the **Feature Upgrade** installation type.

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\jadetest\system\jadetest.ini
```

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

The **Database Directory** text box enables you to explicitly specify the location in which the database (system) files are installed.

When installing on a non-Windows Server 2008, Windows 7, or Vista operating system or on Windows Server 2008, Windows 7, or Vista when the destination folder is not **\Program Files**, the database destination defaults to **system** under the install folder (for example, if you specify **c:\Jade63** in the **Install Directory** text box, the database directory defaults to **c:\Jade63\system**).

If the installation directory is a subdirectory of the programmatically determined location of **\Program Files** on Windows Server 2008, Windows 7, or Vista, 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\Jade63** in the **Install Directory** text box, the default database location is **c:\ProgramData\Jade63\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 **JADE JIT Debugger** folder is displayed. The JADE Just-In-Time (JIT) debugger is required to reliably acquire a dump and crash log when an exception occurs in a system running with the Microsoft Visual C++ run time.

Note It is recommended that you install this on all machines hosting JADE nodes.

13. 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.
14. 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.
15. A warning message box is then displayed, advising you that Dynamic Link Libraries (DLLs) may need to be recompiled. Click the **OK** button, to continue.
16. A warning message may be displayed if the upgrade validation process has not completed. If so, check the **jadeupgrade.log** file for details about what needs to be modified in your user schemas to pass the validation and enable application execution.
17. 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.

18. Click the **Finish** button to end the JADE upgrade process.
19. Install any **jadrap** database services (that is, JADE Remote Node Access services) you had set up in JADE 6.2. (For details, see “[Running the Server Node as a Service](#)”, in the *JADE Remote Node Access Utility User’s Guide*.)

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

Some obsolete files are deleted from the JADE directories when upgrading from JADE 6.2. If you require these files for your JADE system, you must save them before you upgrade or restore them from the original JADE 6.2 release medium.

Running Two Windows 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.

Upgrading to JADE 6.3 from a Linux JADE 6.2 or Earlier JADE 6.3 Release

You can upgrade from JADE 6.2 to the 64-bit edition of JADE only if patch versioning has never been used in the JADE system. If patch versioning has ever been used, you must first upgrade to the 32-bit edition before you can upgrade to the 64-bit edition.

To upgrade from an existing JADE release to JADE release 6.3 on UNIX servers under SUSE Linux Enterprise Server 10.0 or Red Hat 5.0 or higher, perform the following actions.

1. Ensure that your JADE environment is JADE release 6.2 or earlier JADE 6.3 release.
2. Take a full backup copy of your existing JADE directories, first ensuring that your database is not in recovery mode.
3. Install the required JADE Red Hat Package Manager (RPM) file directly by using the standard Linux RPM install tools. This puts the files in the **/opt/jade** directory.

For details, see “Installing JADE”, in Chapter 3 or Chapter 4 of the *JADE Installation and Configuration Guide*.

4. To upgrade your JADE installation, use the **jadeinstall -U** parameter, as shown in the following example.

```
/opt/jade/sbin/jadeinstall -i /home/jade -U -v 6.3.09.000 --all
```

The parameter values are as follows.

- **-i <dir>** is the previously installed JADE directory
- **-v <version>** is the new JADE version to be installed
- **--all** indicates that all components previously installed into the directory specified in the **-i** parameter will be upgraded

The upgrade process copies over the new binaries and required system map files, resets timestamps, and performs any other steps necessary to complete the upgrade.

Caution As with any JADE release, you may need to recompile any external method libraries or external programs using the JADE Object Manager APIs with the new JADE `/include` and `/lib` 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*.)

For more details, see “Installing JADE on a UNIX Server under Linux” and “Parameters for the `jadeinstall` Command”, in Chapter 3 or Chapter 4 of the *JADE Installation and Configuration Guide*.

JADE Thin Client Upgrade

Note If your applications allowed for two UTC timestamp adjustments when `TimeStampOffset` local variables were initialized, you must change you application before you upgrade to this release. For details, see “TimeStampOffset Local Variable Initialization”, in the [RelInfo6307.pdf](#) document in your JADE **documentation** directory.

A JADE 6.3 presentation client upgrade:

- Rejects a presentation client upgrade from 5.2.08 or earlier. You must handle a presentation client download from JADE 6.1 by first upgrading JADE on the presentation client to release 6.0, 6.1, or 6.2.
- Cannot handle a reversion to JADE 6.1 or earlier.
- Rejects a reversion to JADE 6.0 or 6.1 if the JADE 6.0 or 6.1 application server attempts to download files to the **DownloadDirectory2** directory. The only reversion that is guaranteed is from JADE release 6.3 to JADE release 6.2.
- If you are upgrading presentation clients to JADE release 6.3 under Vista, ensure that you have the appropriate privileges or capabilities to install applications. The configuration of Vista’s User Account Control (UAC) and your current user account privileges may affect the behavior of the upgrade to JADE 6.3. For details about Vista UACs, standard user accounts, and administrator accounts, see:

<http://technet2.microsoft.com/WindowsVista/en/library/00d04415-2b2f-422c-b70e-b18ff918c2811033.aspx?mfr=true>

When running JADE in thin client mode under Windows Vista or Windows XP, if the presentation client is installed:

- Under the **\Program Files** directory (or the **\Program Files (x86)** directory on a 64-bit machine with 32-bit JADE binaries), when an automatic presentation client upgrade occurs, you must have administrator rights or know the user name and password of a user with administrator rights. (This is a normal Microsoft requirement that updating of files under the **Program Files** directory requires administrative rights.)

If the Vista 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 logon and password of a user with administrative privileges to enable the upgrade to succeed.

- Outside of the **\Program Files** directory (or the **\Program Files (x86)** directory), privilege elevation to perform an automatic thin client upgrade is not requested. For more details, see Appendix B, “Upgrading Software on Presentation Clients”, in the *JADE Thin Client Guide*.

Upgrading a Synchronized Database Environment (SDE)

To upgrade a Synchronized Database Service (SDS) installation from JADE release 6.2.12 or higher to JADE 6.3, perform the following actions.

1. Upgrade the primary system, as specified in earlier sections of this document.

Notes Upgrading a primary database causes a special **version check** trigger record to be written to the database journal to mark the boundary where conversion from JADE 6.2 to 6.3 occurred.

A 32-bit secondary cannot communicate with a 64-bit primary, or the reverse.

When upgrading an SDS database from a JADE 6.2 release, you must do so using the 32-bit software. When all SDS participants have been upgraded, you can then upgrade them to use 64-bit JADE software, if required.

2. Connect any JADE 6.2 native or RPS secondary database servers to the JADE 6.3 primary so that any remaining JADE 6.2 database journals are transferred and applied.

When the upgrade **version check** record is replayed, the following messages are recorded in the **jommsg.log** and tracking is halted.

```
SDS: Secondary upgrade version mismatch: tracking will now halt
SDS: Upgrade to the same software release level as the primary and
restart server
```

3. When database tracking halts at the upgrade trigger point, shut down the server and upgrade the secondary system by performing one of the following actions.
 - On Windows, use the automated InstallShield script provided in this release and select the **Feature Upgrade** option on the Installation Type dialog.
On the Setup Type dialog that is then displayed, select the **SDS/RPS Database Server** option.
 - Copy the:
 - i. Binary files (including **_sys*.bin** and **_jad*.bin** files)
 - ii. Monitor and dmpload map files
 - iii. Reset the timestamps

Upgrading an RPS Node from JADE 6.2 to 6.3

On upgrade from 6.2 to 6.3, the database type on the RPS node is set to **SQL Server 2000**, since SQL Server 2000 data types only were used in JADE 6.2.

If the RPS mapping database type was set to **SQL Server 2005** in the 6.2 system and you want the RPS mapping to use the SQL Server 2005 types, after the upgrade to 6.3 has completed, execute the following steps.

1. Start the RPS Manager on the RPS node.
2. Stop the **Datapump** application.

3. In the Configure RPS Node dialog, change the database type to **SQL Server 2005**.
This causes SQL scripts to be created and run to modify the columns to use the **SQL Server 2005** database types.
4. Start the **Datapump** application.

Upgrade Validation

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

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

Reorganization

Updates from user applications to user data files during reorganization offline processing are now prohibited and will encounter error *3116 – Database file is locked for reorganization*.

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. You do not need to reset the timestamps.

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 for Windows and **\$JADEHOME/runtime** for Linux).

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

Changes in JADE Release 6.3.09

This section contains changes in JADE release 6.3.09.

For details about changes in JADE:

- Release 6.3.08, see “[Changes in JADE Release 6.3.08](#)”, later in this document.
- Releases 6.3.07, 6.3.06, 6.3.05, and 6.3.04, see [RelInfo6307.pdf](#), in your JADE **documentation** directory.
- Release 6.3.03 (the first general release of JADE 6.3), see [RelInfo6303.pdf](#), in your JADE **documentation** directory.

Adding an Existing Property or Method to an RPS Table (NFS 55461, NFS 57061)

The new **DropHistoricalTableOnAddExisting** parameter in the [JadeRps] section of the JADE initialization file controls the behavior when you add an existing property or method to a historical table in an RPS mapping. The default value of this parameter is **true**; that is, the historical table is dropped if an existing property or method is added.

To modify the historical table (rather than drop the table) when an existing property or method is added, set the value of this parameter to **false**; which will then result in an **ALTER TABLE <> ADD** being used to modify the table when an existing property or method is added. Any existing rows will have a column value of **NULL**.

This parameter is read when the alter table script is created.

Caution If you have applied hot fix 6.3.08.024 but not hot fix 6.3.08.28 and you are using the new functionality provided by the **DropHistoricalTableOnAddExisting** parameter value of **false** (introduced by NFS 55461), you must set the **DropHistoricalTableOnAddExisting** parameter value before applying this upgrade.

Changing the Size of Text in a JadeTextEdit Window (NFS 56953)

Reminder You can use the following keystroke combinations to change the size of the text in a **JadeTextEdit** window.

- CTRL+numeric keyboard + (plus), to increase text size by one point
 - CTRL+numeric keyboard – (minus), to decrease text size by one point
 - CTRL+numeric keyboard / (divide), to set text to the default size
-

Date Parsing Routines for Two-Digit Years (PAR 56832)

The **JadeEditMask** class now uses the Windows Control Panel setting to convert a two-digit year into a four-digit year for a two-digit edit mask year of 'yy' when the value of the **EnhancedLocaleSupport** parameter in the [JadeEnvironment] section of the JADE initialization file is set to **true**.

By default, years:

- 00 through 29 become 2000 through 2029
- 30 through 99 become 1930 through 1999

If the value of the **EnhancedLocaleSupport** parameter is **false** (the default), the year is calculated using the current century.

Dock Controls (PAR 56859)

The drawing of dock control drag rectangles was changed from drawing on the desktop because such drawing under Windows 7 was slow and problematic (Windows can erase the drawing without warning).

If a dock container and the MDI client window share the same parent and overlap, when drawing on the dock container, the MDI client window causes that drawing to clip out its own window area.

JADE now recognizes the situation where the drawing is over a sibling MDI client window and it reverts to drawing on the desktop in that situation. This may then result in the drawing flickering and leaving drawing remnants, because of Windows interference.

External Database Access (NFS 30972)

The [ExternalDb] section of the JADE initialization file can now contain the **SQLConnectOption** parameter, which controls the **DriverCompletion** option sent to the ODBC **SQLDriverConnect** call.

You can set this parameter to one of the values listed in the following table.

Value	DriverCompletion option is set to...
Complete	SQL_DRIVER_COMPLETE (the default)
Prompt	SQL_DRIVER_PROMPT
NoPrompt	SQL_DRIVER_NOPROMPT
Complete_Required	SQL_DRIVER_COMPLETE_REQUIRED

For more details about the **DriverCompletion** options for the ODBC **SQLDriverConnect** call, see the Microsoft ODBC documentation.

JADE Debugger (PAR 56580)

Inspection of properties from within the JADE debugger (either directly or via the bubble help inspector) will invoke user mapping methods, if defined.

Caution You should be aware that debugging may fail if the mapping method performs actions not suitable for situations where the execution of the user application is suspended. For example, the creation and display of a form, showing a form modally, display of a message box, deletion of an object in use by the current call stack, and so on.

JadeHTTPConnection Class

JADE now provides the **JadeHTTPConnection** class, which inherits from the **Object** class.

The **JadeHTTPConnection** class enables applications to access the standard Internet protocol HTTP. For ease of use, this class abstracts this protocol into a high-level interface. The underlying implementation uses the Microsoft WinHTTP library or the WinINET library (using the respective **EnableWinHTTP** or **EnableWinINET** parameter in the [JadeEnvironment] section of the JADE initialization file).

If you want to use this feature on the server or in a service, you will need to use the WinHTTP library option.

A basic understanding of the HTTP protocol is required to use this feature. The two Requests for Comments (RFCs) that define this protocol are:

- RFC 1945, Hypertext Transfer Protocol - HTTP/1.0
- RFC 2616, Hypertext Transfer Protocol - HTTP/1.1

JadeHTTPConnection Class Constants

The constants provided by the **JadeHTTPConnection** class are listed in the following table.

Name	Type and Value	Description
Default_FTP	Integer = 21	Default FTP port
DefaultPort_HTTP	Integer = 80	Default HTTP port
DefaultPort_HTTPS	Integer = 443	Default HTTPS port
DefaultPort_Invalid	Integer = -1	Default invalid port
DefaultPort_Socks	Integer = 1080	Default SOCKS port
HeaderType_Client	Integer = 1	Client HTTP header type
HeaderType_Server	Integer = 2	Server HTTP header type
Header_Accept	String = "Accept"	Accept header
Header_ContentLength	String = "Content-Length"	Content-length header
Header_ContentType	String = "Content-Type"	Content-type header
Header_Host	String = "Host"	Host header
Header_KeepAlive	String = "Proxy-Connection"	Proxy-connection header
Header_ReasonPhrase	String = "Reason-Phrase"	Reason-phrase header
Header_Referer	String = "Referer"	Referer header
Header_SoapAction	String = "SOAPAction"	SOAPAction header
Header_StatusCode	String = "Status-Code"	Status-code header
Header_UserAgent	String = "User-Agent"	User-agent header
ProtocolVersion_1_0	String = "HTTP/1.0"	HTTP Protocol 1.0
ProtocolVersion_1_1	String = "HTTP/1.1"	HTTP Protocol 1.1
ProxyConfig_Direct	Integer = 1	Resolves all host names locally
ProxyConfig_Preconfig	Integer = 0	Retrieves the proxy or direct configuration from the registry
ProxyConfig_Preconfig_NoAuto	Integer = 4	Retrieves the proxy or direct configuration from the registry and prevents the use of a start-up Microsoft JScript or wpad.dat file
ProxyConfig_Proxy	Integer = 3	Passes requests to the proxy
Scheme_DIRECT	String = "jadehttp.tcp"	JADE Direct scheme
Scheme_DIRECT_6_2	String = "jadehttp.tcp2"	JADE 6.2 Direct scheme
Scheme_FILE	String = "file"	File scheme; not supported
Scheme_FTP	String = "ftp"	FTP scheme; not supported

Name	Type and Value	Description
Scheme_HTTP	String = "http"	HTTP scheme
Scheme_HTTPS	String = "https"	HTTPS scheme
State_Failure	Integer = 8	Connection failed
State_NeedConnection	Integer = 0	Not connected
State_NeedRequest	Integer = 1	Connected and waiting for request
Verb_GET	String = "GET"	The GET operation
Verb_POST	String = "POST"	The POST operation

JadeHTTPConnection Properties

The properties defined in the **JadeHTTPConnection** class are listed in the following table.

Property	Description
abs_path	Sets or retrieves a string that contains the path portion of the url property, where the URL is made up of the following components. “scheme://[user[:password]@]host[:port]/path[?query][#fragment]”
connectTimeout	Integer value that specifies the timeout value, in milliseconds, to use for server connection requests. If a connection request takes longer than the specified timeout value, the request is cancelled. The initial value is 120,000 (that is, 120 seconds).
fragment	Sets or retrieves a string that contains the fragment portion of the url property, where the URL is made up of the following components. “scheme://[user[:password]@]host[:port]/path[?query][#fragment]”
hostname	Sets or retrieves a string that contains the name of the host to which to connect. This string is passed when the connection is requested.
httpVersion	Sets or retrieves a string that contains the version of the HTTP protocol. Its value can be one of the following JadeHTTPConnection class constants. <ul style="list-style-type: none"> ■ ProtocolVersion_1_0, for HTTP Version 1.09 ■ ProtocolVersion_1_1, for HTTP Version 1.1 (the default value)
password	Sets or retrieves a string that contains the password portion of the url property, where the URL is made up of the following components. “scheme://[user[:password]@]host[:port]/path[?query][#fragment]” If the string is not null, this URL is passed when the connection is requested.
port	An Integer value that specifies the port number to use when the connect operation is requested.
proxyConfig	Read-only Integer value that defines the required type of access. Its value can be one of the following JadeHTTPConnection class constants. <ul style="list-style-type: none"> ■ ProtocolConfig_Direct, which resolves all host names locally, and does not use a proxy ■ ProxyConfig_PreConfig, which retrieves the proxy or direct configuration from the registry (the default value) ■ ProxyConfig_PreConfig_NoAuto, which retrieves the proxy or direct configuration from the registry and prevents the use of a start-up Microsoft JScript or wpad.dat file ■ ProxyConfig_Proxy, which passes requests to the proxy

Property	Description
proxyHostname	Read-only String property that specifies the name of the proxy server or servers to use when proxy access is specified by setting the value of the proxyConfig property to ProxyConfig_Proxy . If proxyConfig is not set to ProxyConfig_Proxy , this value will be null.
proxyPassword	Read-only String property that specifies the password for the proxy server authentication, if required. The value is set using the configureProxy method. Some proxy servers require a user name and password before connecting to the target host.
proxyUser	Read-only String property that specifies the user name for the proxy server authentication, if required. The value set using the configureProxy method. Some proxy servers require a user name and password before connecting to the target host.
query	Sets or retrieves a string that contains the queryent portion of the url property, where the URL is made up of the following components. “scheme://[user[:password]@]host[:port]/path[?query][#fragment]”
receiveTimeout	An Integer value that specifies the timeout value, in milliseconds, to receive a response to a request. If a response takes longer than the specified timeout value, the request is cancelled. The initial value is 120,000 (that is, 120 seconds).
responseBody	Read-only Binary value that contains the response message.
responseHeaders	Read-only String value that contains the HTTP headers from the response message.
scheme	Sets or retrieves a string that contains the scheme. Its value can be one of the following JadeHTTPConnection class constants. <ul style="list-style-type: none"> ▪ Scheme_DIRECT, for direct JADE to JADE Web services earlier than JADE 6.2 ▪ Scheme_DIRECT_6_2, for direct JADE to JADE Web services from JADE 6.2 or later ▪ Scheme_FILE; known scheme, but not supported by JADE ▪ Scheme_FTP; known scheme, but not supported by JADE ▪ Scheme_HTTP, for the HTTP scheme (the default) ▪ Scheme_HTTPS, for the HTTPS scheme
sendTimeout	An Integer value that specifies the timeout value, in milliseconds, to use for sending requests. If sending a request takes longer than the specified timeout value, the send is cancelled. The initial value is 120,000 (that is, 120 seconds).
state	Read-only Integer value that defines the state of the connection. Its value can be one of the following JadeHTTPConnection class constants. <ul style="list-style-type: none"> ▪ State_Failure, that defines the open or connect failed setting options ▪ State_NeedConnection, indicating that the connection is not open or needs to be opened ▪ State_NeedRequest, indicating that the connection is ready and waiting for a request
url	Sets or retrieves a string that contains the URL of the page being requested, where the URL is made up of the following components. “scheme://[user[:password]@]host[:port]/path[?query][#fragment]”

Property	Description
usePresentationClient	Boolean value that when set to true , uses WinHTTP (or WinINET) from the machine from which the presentation client is running. The initial value is false .
user	Sets or retrieves a string that contains the user portion of the url property value, where the URL is made up of the following components. “scheme://[user[:password]@]host[:port]/path[?query][#fragment]” If the string is not null, the value of this property is passed when the connection is requested.

JadeHTTPConnection Methods

The methods defined in the **JadeHTTPConnection** class are documented in the following subsections.

configureProxy

The **configureProxy** method, which sets up the proxy server configuration, has the following signature.

```
configureProxy (config: Integer;
               host: String;
               user: String;
               password: String): Boolean updating;
```

The **config** parameter, which defines the type of access required, can be one of the **JadeHTTPConnection** class constants listed in the following table.

Value	Description
ProxyConfig_Direct	Resolves all host names locally, and does not use a proxy server.
ProxyConfig_PreConfig	Retrieves the proxy or direct configuration from the registry (the default value).
ProxyConfig_PreConfig_NoAuto	Retrieves the proxy or direct configuration from the registry and prevents the use of a start-up Microsoft JScript or wpad.dat file.
ProxyConfig_Proxy	Passes requests to the proxy.

The **host** parameter specifies the name of the proxy server or servers to use when proxy access is specified by setting the value of the **config** parameter to **ProxyConfig_Proxy**. If the value of the **proxyConfig** property is not set to **ProxyConfig_Proxy**, the value of the **host** parameter should be null.

If the proxy server requires authentication, the **user** parameter contains the user name provided to the proxy server. If the **proxyConfig** property is not set to **ProxyConfig_Proxy**, the value of the **user** parameter should be null.

If the proxy server requires authentication, the **password** parameter contains the password provided to the proxy server. If the **proxyConfig** property is not set to **ProxyConfig_Proxy**, the value of the **password** parameter should be null.

The **configureProxy** method returns **true** if the method call was successful in setting up the proxy server configuration; otherwise it returns **false**.

getHeader

The **getHeader** method, which retrieves an HTTP header, has the following signature.

```
getHeader(type: Integer;
          key: String): String;
```

The **type** parameter, which specifies the type of header to retrieve, can be one of the **JadeHTTPConnection** class constants listed in the following table.

Value	Description
HeaderType_Client	Client header
HeaderType_Server	Server header

The **key** parameter specifies the header key; for example, **Accept**, **Content-Type**.

The **getHeader** method returns the value of the key if a header with the specified type and key exists; otherwise it returns null.

getHttpPage

The **getHttpPage** method, which returns the requested page, has the following signature.

```
getHttpPage(pVerb: String,
            pServerName: String;
            pUrlAddress: String;
            pMessage: String;
            pContentType: String): String updating;
```

The **pVerb** parameter specifies the HTTP verb to use in the request. The value can be **"GET"** (the default), **"POST"**, or null.

The **pServerName** parameter specifies the scheme, host, and other optional parameters. This parameter value is specified as **"scheme://[user[:password]@]host[:port]"**, where:

- The **scheme** value can be the **"http"** or **"https"**
- The **user** and **password** values are optional site security
- The **host** value is a host name or an IP number
- The optional **port** value defaults to 80 for HTTP and to 443 for HTTPS

If the value of the **pServerName** parameter is null, the host information portion of the **url** property is used (that is, **[user[:password]@]host[:port]**).

The **pUrlAddress** parameter specifies the relative URI from the server (that is, **[/path]?query][#fragment]**). If this value is null, the value of the **url** property is used.

The optional **pMessage** parameter specifies the body of the message for POST requests.

The **pContentType** parameter, which specifies the content type, has a default value of **"text/xml; charset=utf-8"**.

The **getHttpPage** method returns a string representing the message response or it returns null if the request failed.

getHttpPageBinary

The **getHttpPageBinary** method returns the complete requested page in binary format, with no text encoding conversion performed.

This method has the following signature.

```
getHttpPageBinary(pVerb: String,
                  pServerName: String;
                  pUrlAddress: String;
                  pMessage: String;
                  pContentType: String): Binary updating;
```

The **pVerb** parameter specifies the HTTP verb to use in the request. The value can be **"GET"** (the default), **"POST"**, or null.

The **pServerName** parameter specifies the scheme, host, and other optional parameters. This parameter value is specified as **"scheme://[user[:password]@]host[:port]"**, where:

- The **scheme** value can be the **"http"**, **"https"**, or **"jadedirect"**
- The **user** and **password** values are optional site security
- The **host** value is a host name or an IP number
- The **port** value is optional for HTTP (defaults to 80) and HTTPS (defaults to 443), and is required for the **jadedirect** scheme

If the value of the **pServerName** parameter is null, the host information portion of the **url** property is used (that is, **[user[:password]@]host[:port]**).

The **pUrlAddress** parameter specifies the relative URI from the server (that is, **[/][path]?query][#fragment]**). If this value is null, the value of the **url** property is used.

The optional **pMessage** parameter specifies the body of the message for POST requests (that is, SOAP requests, and so on).

The **pContentType** parameter specifies the content type. For SOAP 1.2, it is set to **"application/soap+xml"** as well as an optional **action** parameter. If the value of the **pContentType** parameter is null, it defaults to **"text/xml; charset=utf-8"**.

The **getHttpPageBinary** method returns the complete page or it returns null if there is a problem. Use the **queryStatusCode** method to determine the HTTP error.

open

The **open** method, which opens an HTTP connection, has the following signature.

```
open(closeFirst: Boolean): Boolean updating;
```

If the HTTP connection is currently open and you want it closed before a new connection is opened, set the **closeFirst** parameter to **true**.

Note The **url** property must be set before the connection can be opened.

This method returns **true** if the open operation was successful; otherwise it returns **false**. The value of the **state** property is set to **State_NeedRequest** if the open action was successful; otherwise it is set to **State_Failure** if it was unsuccessful.

queryConnectionIsClose

The **queryConnectionIsClose** method, which checks if the HTTP connection is closed, has the following signature.

```
queryConnectionIsClose(): Boolean updating;
```

This method returns **true** if the connection is closed; otherwise it returns **false**.

You can call this method to check the "Connection" header state before calling the **queryInfo** method, for example.

queryContentLength

The **queryContentLength** method, which returns the length of the response message, has the following signature.

```
queryContentLength(): Integer updating;
```

This method returns the length of the response message in bytes; or it returns zero (**0**) if the response message is empty.

You can call this method to check the "Content-Length" header state before calling the **queryInfo** method, for example.

queryContentType

The **queryContentType** method, which returns the content type of the response message, has the following signature.

```
queryContentType(): String updating;
```

This method returns the content type of the response message (for example, "**text/html**"); or it returns null if the response message is empty.

You can call this method to check the "Content-Type" header before calling the **queryInfo** method, for example.

queryDate

The **queryDate** method, which returns the timestamp at which the response message originated, has the following signature.

```
queryDate(): TimeStamp updating;
```

This method returns a timestamp representing the date and time of origin on the response message (for example, **Mon, 27 Jun 2011 23:43:27 GMT**); or it returns null if the response message is empty.

You can call this method to check the "Date" header before calling the **queryInfo** method, for example.

queryInfo

The **queryInfo** method, which retrieves header information associated with an HTTP request, has the following signature.

```
queryInfo(key: String,  
          index: Integer io): String updating;
```

The **key** parameter specifies the HTTP header key that is to be retrieved; for example, **Accept**, **Date**.

The **index** parameter enumerates multiple headers with the same key. When calling the function, this parameter is the index of the specified header to return (which is usually 0). When the function returns, this parameter is the index of the next header. If the next index cannot be found, a null value is returned.

queryStatusCode

The **queryStatusCode** method, which retrieves the HTTP status code returned by the server, has the following signature.

```
queryStatusCode(): Integer updating;
```

This method returns an Integer value representing the HTTP status code returned by the server or it return null if the status code is empty.

readBody

The **readBody** method, which retrieves the response message from an HTTP request, has the following signature.

```
readBody(length: Integer): Binary updating;
```

The **length** parameter, if specified, pre-allocates space for the **responseBody** property. The default value is zero (0).

Note For improved performance, we recommend that you use the content length if it is known or it can be estimated.

This method returns a Binary value representing the message response of an HTTP **"POST"** of the passed URL returned from the server, by reading data in **Default_Size_NetworkRead** chunks (that is, 4K bytes). The **responseBody** property is set to this returned value.

sendRequest

The **sendRequest** method, which sends the request to the HTTP server, has the following signature.

```
sendRequest(verb: String,
            additionalHeaders: String;
            optionalPostPutData: String): Boolean updating;
```

This method sends the specified request to the HTTP server and allows the client to specify additional headers to send along with the request. In addition, it allows the client to specify optional data to send to the HTTP server immediately following the request headers. This feature is generally used for write operations such as PUT and POST.

The **verb** parameter specifies the HTTP verb to use in the request. The value can be **"GET"** (the default), **"POST"**, or null. A null value implies a **"GET"** request.

The **additionalHeaders** parameter specifies a string containing the additional headers to be appended to the request. If there are no additional headers to be appended, the value of this parameter can be null.

The **optionalPostPutData** parameter specifies a string containing any optional data to be sent immediately after the request headers. This parameter is generally used for POST and PUT operations. The optional data can be the resource or information being posted to the server. If there is no optional data to send, the value of this parameter can be null.

This method returns **true** if the send request was successful; otherwise it returns **false**.

setAccept

The **setAccept** method, which sets "Accept" HTTP headers, has the following signature.

```
setAccept(value: String,
          index: Integer io) updating;
```

The **value** parameter specifies the string value for the Accept key; for example, "**text/xml**".

If you specify a value for the **index** parameter and the Accept key exists at the specified position in the list of headers, the corresponding entry is removed from the list if the value is null. If the value is non-null, the corresponding value is replaced. If there is no Accept key in the specified position, the key-value pair is added at the position.

setContentTypes

The **setContentTypes** method, which sets "Content-Type" HTTP headers, has the following signature.

```
setContentTypes(value: String,  
               index: Integer io) updating;
```

The **value** parameter specifies the string value for the Content-Type key; for example, "**text/xml; charset=utf-8**".

If you specify a value for the **index** parameter and the Content-Type key exists at the specified position in the list of headers, the corresponding entry is removed from the list if the value is null. If the value is non-null, the corresponding value is replaced. If there is no Accept key in the specified position, the key-value pair is added at this position.

setKeepAlive

The **setKeepAlive** method, which sets the "Proxy-Connection" HTTP header, has the following signature.

```
setKeepAlive(flag: Boolean) updating;
```

Set the value of the **flag** parameter to **true** to generate the "Proxy-Connection : Keep-Alive" header; or set the value to **false** to remove the "Proxy-Connection : Keep-Alive" header if it exists.

setReferer

The **setReferer** method, which sets the "Referer" HTTP header, has the following signature.

```
setReferer(value: String) updating;
```

Set the **value** parameter to the string value for the Referer key; for example:

```
"http://www.w3.org/hypertext/DataSources/Overview.html"
```

If the value of this parameter is null and the Referer key exists in the list of HTTP headers, the corresponding entry is removed from the list. If the value is non-null, the corresponding value is replaced.

setReload

The **setReload** method, which sets the "Pragma" HTTP header, has the following signature.

```
setReload(flag: Boolean) updating;
```

Set the value of the **flag** parameter to **true** to generate the "Pragma: no-cache" HTTP header; or set the value to **false** to remove the "Pragma: no-cache" header if it exists.

setSoapAction

The **setSoapAction** method, which sets the "SOAPAction" HTTP header, has the following signature.

```
setSoapAction(value: String) updating;
```

Set the **value** parameter to the string value for the "SOAPAction" header. If the value of this parameter is null and the SOAPAction key exists in the list of headers, the corresponding entry is removed from the list. If the value is non-null, the corresponding value is replaced.

setUserAgent

The **setUserAgent** method, which sets the "User-Agent" HTTP header, has the following signature.

```
setUserAgent(value: String) updating;
```

Set the **value** parameter to the string value for the "User-Agent" header. If the value of this parameter is null and the User-Agent key exists in the list of headers, the corresponding entry is removed from the list. If the value is non-null, the corresponding value is replaced.

JadeProfiler::report Method Output (PAR 55984)

The output of the **JadeProfiler** class **report** method now includes the schema name with the class and method names, as shown in the following example.

Time	Count	Minimum	Maximum	Average	%	Method
107	1	107	107	107.01	22.72	ErewhonInvestmentsViewSchema::FormShopSaleItems::zBuildCartTable

Journal Rate Analysis Sampling (PAR 57125)

The sampling interval granularity has been changed from seconds to milliseconds. The default sampling interval remains one minute, expressed as **60000** milliseconds.

Analysis, generation of summary and total values, and interval snapshot data have been enhanced, by the transaction count now being the number of active transactions seen derived from the transaction IDs of the update records processed, rather than the number of **BEGIN_TRANSACTION** records seen.

An additional sample column containing the number of **COMMIT_TRANSACTION** records seen has been added. The file summary and totals summary have also been enhanced to report on the transaction and commit activity.

Portable GUI Thin Client Upgrade (PAR 56161)

After upgrading a Linux Portable GUI thin client from JADE 6.2 to JADE 6.3.08 or higher, runtime error 1461 is raised by the application server when the client tries to start up if **en_US** is the only installed locale, as this locale is a pseudo locale identifier (LCID) that has no details in the Linux **locales.cat** file.

You can now specify the LCID number in the new **UserDefaultLCID** parameter in the [JadeEnvironment] section of the JADE initialization file, which will remap the user default LCID to the LCID of your choice. The default value is the system LCID (that is, **1033**).

Recompiling Methods after Upgrading from a 6.2 to a 6.3 Release (PAR 56522)

After upgrading from a JADE 6.2 release to a JADE 6.3 release, any methods that reference the following methods in the **Process** class must be manually recompiled.

- `getCallStackInfo`
- `sendCallStackInfo`
- `sendTransientFileAnalysis`
- `sendTransientFileInfo`

Changes in JADE Release 6.3.08

This section contains changes in JADE release 6.3.08.

For details about changes in JADE:

- Releases 6.3.07, 6.3.06, 6.3.05, and 6.3.04, see [RelInfo6307.pdf](#), in your JADE **documentation** directory.
- Release 6.3.03 (the first general release of JADE 6.3), see [RelInfo6303.pdf](#), in your JADE **documentation** directory.

Recompiling Methods after Upgrading from Release 6.2 to 6.3.07 (PAR 55301)

After upgrading from a JADE 6.2 release to JADE 6.3.07 or higher, any methods that contain **create object as variable** must be recompiled, to fix the type of persistence used when the object is created.

The **Schema** class now provides the undocumented **_fixCreateAs** system method, which locates any JADE methods that use **create object as variable** in any of your schemas and recompiles them. This method creates the **FixCreateAs.log** file in the system log directory, which lists all of these methods that successfully compiled and all methods that failed to compile.

The file output has a format like that shown in the following example.

```
Checking for methods that use create as variable...
Checking for create as <variable> statements complete, errors: 0
  methods with create as <variable> statements:                1
  methods recompiled successfully:                             1
  methods with compile errors:                                  0

The following methods were successfully compiled:
  Contact55301::IFMessageProcessorController::doActionMessage
```

To run the **_fixCreateAs** method, create and run a **JadeScript** method as follows.

```
fixCreateAs();
vars
begin
  currentSchema._fixCreateAs();
end;
```

Manually execute this method in the upgrade script if you are upgrading to JADE 6.3.07 or higher from a JADE 6.2 release.

.NET Requirements

Chapter 1 of the *JADE Installation and Configuration Guide* states that you must install .NET 3.0 or higher, to generate and use the .NET library. However, .NET 4.0 is not compatible with .NET 3.0 and .NET 3.5.

Installing .NET 4.0 does not satisfy the requirements of .NET 3.0 or .NET 3.5 to run .NET components in JADE 6.3; that is, you require .NET 3.0 or 3.5 to use JADE 6.3 external components.

In addition, you cannot import a .NET 4.0 component into JADE 6.3.

Changing a Control Type to a Superclass or Subclass (PAR 54738)

User-implemented event methods are linked back to the event method of the control so that the method signature of the user event method is consistent with that of the control.

In earlier releases, an event named **xyz** on two different control types can have totally different parameters, which JADE tries to handle by ensuring that the user event signature always matches the event signature of the control. However, when dealing with events on controls that are subclasses of each other, these events must have the same parameters.

From this release, if the type of a control is changed to a superclass or subclass of the original type, event method headers are left untouched.

Note, however, that although these methods are syntactically correct and would compile without errors, they *may* fail at runtime. For example, if you create a **ListBoxSub** subclass of **ListBox**, add **ListBoxSub** to a form, implement the **displayRow** method and change its signature so that the first parameter is of type **ListBoxSub**, and change the control type from **ListBoxSub** to **ListBox** in the JADE Painter, the **displayRow** event method remains valid. However, when the event occurs at run time, the first parameter will be of type **ListBox** rather than **ListBoxSub**, so an invalid parameter exception will be raised.

Class Lifetimes

This section contains class lifetime changes in this release.

Persistence when Extracting and Loading Schemas (NFS 55284, PAR 53319)

If a class already is defined in the database as a *normal* class that that can have both persistent and transient instances (that is, the **transientsOnly** class lifetime is set to **false** so that the class number is less than 20,480) and the definition in the schema or class file is as a transients-only class (that is, **transientsOnly** is **true** and the class number is 20,480 or higher), the JADE compiler now allows the loading of a schema or class if the class already defined in the database does not have the **persistentAllowed** class lifetime set to **true**.

Notes If the **persistentAllowed** class lifetime is set to **true**, the load will not be allowed, even if there are no persistent instances of the class.

If a class is defined in the database as a **transientsOnly** class and the definition in the schema or class file is as a normal class, the compiler will allow the load and all methods referencing the class will be (potentially versioned and) recompiled.

In earlier releases, exception 6404 (*Cannot change class transientsOnly setting*) was always raised.

In addition:

- Subclass lifetimes are now retained during a schema file load. All of the subclasses are updated to reflect the change and the default lifetime is changed, if required.
- A new compiler error 6435 (*Cannot remove subclassPersistentAllowed option as subclass instances exist*) can be reported.

Customized Deployment

This section contains the customized deployment changes in this release.

Customized Deployment Upgrade (PAR 53457)

The customized deployment upgrade now sets the default value for the **ShowLicenseKey** parameter to **No** if no license key is required for an upgrade or to **Yes** if a license key is required.

For JADE 6.3, no license key is required for the upgrade from JADE 6.2.

Setup.ini File

The **AppName** parameter in the [Startup] section of the **setup.ini** file has been renamed to the **Product** parameter.

Dates Entered in Text Boxes

The “Locale Handling Enhancements” section in the JADE 6.3.07 Release Information stated that the **TextBox** class **getTextAsLongDate** and **getTextAsShortDate** methods raised an exception if the text box is empty or the text is not a valid long or short date.

However, if the text box is empty, null is returned. If the text is not a valid long date or short date value, respectively, an exception is raised.

Install Shield Version

The JADE 6.3.08 installations have been created using InstallShield 2011.

If you install multiple copies of JADE 6.3.08 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.

ODBC User Method Exception Logging (PAR 55105, 52344)

Exceptions raised in user methods will be logged if the **LogUserMethodExceptions** parameter in the [JadeOdbc] section of the JADE initialization file is set to **true**.

Tip For methods that are mapped to ODBC columns, it is more efficient to handle exceptions within your user code and return a valid value rather than propagating the error out to the ODBC driver code.

Relational Population Service (RPS)

This section contains the RPS changes in this release.

Many-to-Many Junction Tables Naming Convention (PAR 55186)

When a many-to-many junction table is created in an RPS mapping and the class of both collections is the same type, the second column in the junction table is created with the name **class-name_oid2**.

The default column name for a junction table column is **class-name_oid**.

Mapping Many-to-Many Collections (PAR 55352)

The **Many To Many Table Mapping** sheet of the RPS Wizard now provides the new **Map SubClasses to this Class Map** command in the popup menu, which enables you to include subclasses that contain many-to-many inverse references in the same junction table class map, and the **Class** column in the table, which provides the name of the class from which the junction table is derived for information purposes (that is, it is read-only, as you cannot change it).

Alternatively, you can map the subclasses separately.

Removing a Method in a Delta from an Exported Interface (PAR 55256)

The **Remove** command in the Methods menu is now disabled when a method from an exported interface is checked out, so that neither the original method nor the checked out method or methods can be removed.

Before you can remove the method, you must check it in or undo the checking out of the method.

Report Writer

This section contains JADE Report Writer changes in this release.

Changing the Feature or Path Name of an Alias (NFS 54335, NFS 54664)

You can now change the feature name or path name for an alias:

- When loading a reporting view
- In the **Selected Features** pane of the **Types & Features** sheet of the JADE Report Configuration window, double-click on the feature name or path name that you want to change and then specify the new feature name or path name.

If the following requirements are not met, the name is not changed.

- The feature name or path name must be valid
- The type of the new feature name or path name must match the original definition

Removing Invalid View Features (NFS 38293)

The GUI (**jadload**) and batch (**jadloadb**) JADE Schema Load utilities now enable you to specify whether the JADE Report Writer view file being loaded should replace any existing view with the same name.

The **jadloadb** executable now provides the **reportReplaceView** command line argument, which is **false** by default.

When you the **More Options** button on the Load Schema dialog of the JADE Schema Load utility (**jadload**), the additional load options now provide the **Replace Report View** check box.

If the **reportReplaceView** command line argument is set to **true** or the **Replace Report View** check box is checked and a reporting view with the same name already exists, view items (types, features, root collections, joins, and script methods) that are not in the incoming view extract file and are not used in any existing report definitions are removed. Those items that are used in existing reports are retained.

If the parameter is set to **false** or the check box is unchecked (default) and a reporting view with the same name already exists, the views are merged.