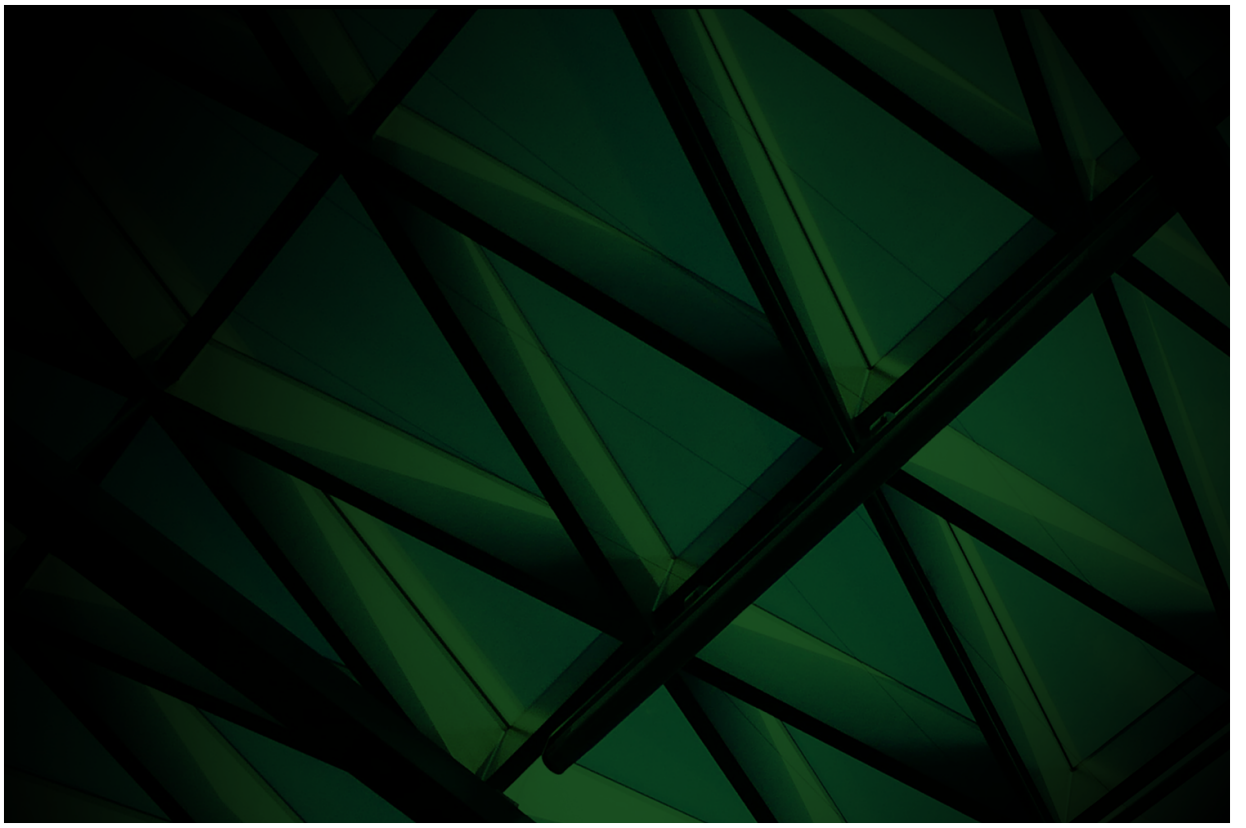

Upgrading to the Jade 2025 Release

VERSION 2025.0.01

jade platform

Copyright©2025 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 © 2025 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 2025, Jade Software Corporation Limited All rights reserved

Copyright 2025, Jade Software Corporation Limited All rights reserved

Upgrading to Jade Release 2025

This document covers the following topics.

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

The *Product Information Library* document (**Jade**) provides a summary of contents of documents in the Jade Platform 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 *Runtime Application Guide*.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Jade Release Support

For details about the:

- Jade Platform release policy, see https://www.jadeplatform.com/developer-centre/support/release-policy.
- Jade Platform release schedule, see https://www.jadeplatform.com/developer-centre.

Jade 2025 is built using Microsoft Visual Studio 2017, which requires the installation of the Microsoft Visual C++ 2017 Redistributables.

For details about the deimplementations and deprecations in this release, see the following subsection.

Deimplementations and Deprecations

This section contains the deimplementations and deprecations in this release.

Collection Concurrency Method Deprecations

As part of the Collection Concurrency Phase 2 changes, the following methods are deprecated.

Class	Deprecated Method	Method to Use Instead *
Collection	includesWithDeferred	includes
Dictionary	getAtKeyWithDeferred	getAtKey
	includesKeyWithDeferred	includesKey

For details, see "Collection Concurrency Phase 2 (JAD-I-423)", later in this document.

Jade JSON Web Token Time Format Setting (PAR 69837)

The Jade initialization file [WebOptions] section **JwtExpectUtc** parameter has been deprecated in this release.

All Jade JSON Web Token (JWT) timestamps are now processed as Coordinated Universal Time (UTC), which is the JSON Web Token standard (https://datatracker.ietf.org/doc/html/rfc7519).

If you set the **JwtExpectUtc** parameter to specify that JWT timestamps are to be processed as local time, this setting is now ignored.

JadeJsonWebToken Class Constants Deprecated

The following **JadeJsonWebToken** class constants have been deprecated.

- Error_SecretTooShort
- MinSecretLength

If any methods reference these constants, your REST web service authentication could fail. For details, see "JSON Web Tokens Minimum Length Increase (PAR 69953)", later in this document.

Process Class Lock Call Stack Method Deprecations

The following table lists the **Process** class methods that have been deprecated in this release, and the new methods to use instead.

Deprecated Method	New method to use instead...
addLockCallStackFilter	addCallStackFilter

Copyright 2025, Jade Software Corporation Limited All rights reserved

Deprecated Method	New method to use instead...
clearLockCallStackFilter	clearCallStackFilter
getLockCallStackFilter	getCallStackFilter
getSaveLockCallStack	getSaveCallStack
setSaveLockCallStack	setSaveCallStack

Note The new methods completely replace the deprecated methods.

The first parameter in each of the new methods is an Integer value that represents the operation to which the call stack applies; that is, create, delete, or lock. To replace the deprecated methods in the previous table, the operation to specify in the new method is the **Process** class constant **CallStack_Lock** (value **4**).

The **getLockCallStack** method of the **Object** class is also deprecated in this release. Use the new **getCallStack** method of the **Process** class, instead.

The deprecated methods will be deleted in a future Jade release.

For details about the new methods, see "Process Class Methods to Control and Retrieve Call Stacks", later in this document.

Server Thread Initialization File Parameters Deimplemented

The following parameters in the [JadeServer] section of the Jade initialization file have been deimplemented in this release.

- MinShortThreads
- MaxShortThreads
- ConcurrentShortThreads
- MinLongThreads
- MaxLongThreads
- ConcurrentLongThreads

For details about the **MinServerThreads** and **MaxServerThreads** parameters that replace them, see "Minimum and Maximum Server Threads", in "Jade 2025 Changes that May Affect Your Existing Systems", elsewhere in this document.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Highlights in this Release

The highlights in Jade release 2025, which help you to deliver high-performance, interoperable applications on Windows for both 32-bit and 64-bit platforms, are as follows.

- Orb web framework. For details, see "Orb Web Framework", later in this document.
- New browsers in the development environment that display additional information about outgoing method calls, explore method call hierarchies, where a property is being updated in the code, and what unit tests cover a method. For details, see "Supplementary Browsers Available", later in this document.
- Provision of the **switch** instruction, which provides a structured way to execute one block of statements from multiple alternative statements, based on the value of an expression. See also "switch Instruction (JAD-I-96)", later this document.
- Jade now supports the .NET Core extension to the .NET Component and .NET Import functionality, which enables you to execute .NET Core runtime functionality. See also "Using the .NET Core Run Time", later in this document.
- You can now perform a number of RPS Manager functions using the command line, including the creation of a new RPS database, multithreading RPS database creation, and converting a primary database backup into an RPS node. For details, see "RPS Enhancements", later in this document.

Copyright 2025, Jade Software Corporation Limited 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 Management and Customer Contact system. This system also enables you to view the progress of your own contacts.

If you have any queries about **Parsys**, please direct them to Jade Support in the first instance, at support@jadeplatform.com.

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

https://www.jadeplatform.com/developer-centre/support

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 **2025** list item).
3. Click the **Search** button.

   Tip To display more than the default 100 entries returned by the search process, select the **User Preferences** command from the File menu to open the User Preferences dialog, select the **Search Defaults** command from the Searching submenu, and in the Maximum hits returned (latest search mode) group box, select the **All** option button or select the **No more than** option button and then increase the value (for example, to **300**) in the adjacent text box.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Upgrading to Jade 2025

This section covers the following topics.

For details about the Jade 2025 software requirements, which differ from those of earlier releases, see "Software Requirements", in Chapter 1 of the *Installation and Configuration Guide*.

Caution Before you upgrade to Jade 2025, refer to "Jade 2025 Changes that May Affect Your Existing Systems", elsewhere in this document.

As with any Jade release (for example, upgrading to a minor release or to a major feature release from an earlier Jade version), you must 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 *Object Manager Guide*.)

Upgrading to Jade 2025 from Jade 2022

Caution In Jade 2022 SP3, the minimum Visual C++ Redistributables have been updated and you will need to install them. For details, see "Visual C++ Redistributables Updated", later in this document.

If you want to develop your own upgrade process, refer to the Jade install and upgrade steps documented in the **ReadmeInstallSteps.pdf** document in the documentation directory.

Note Example files are not part of the installation. They must be downloaded from the appropriate link (for example, **JADE-Erewhon** or **JADE-ATCG**) at https://github.com/jadesoftwarenz.

The Jade Setup program enables you to upgrade your binary and database files to Jade 2025 from Jade 2022, by performing the following actions.

1. On the Jade *2022* 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 the Jade Database utility (**jdbutil.exe** or **jdbutilb.exe**), to ensure that the system is structurally correct. (For details, see Chapter 1 of the *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 *Object Manager Guide*.)

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

   Note If you are unsure how to interpret the information output by the certify process, first refer to "Logical Certifier Errors and Repairs", in Chapter 5 of the *Object Manager Guide*, and if you are still unsure, contact Jade Support (support@jadeplatform.com) for advice.

2. Use the Jade Database utility to take a full backup of your existing Jade 2022 database.

   Caution If the upgrade should fail, you will need to restore this backup and then retry the upgrade process when all of the conditions that caused the failure have been addressed.

Copyright 2025, Jade Software Corporation Limited All rights reserved

3. Installing the Jade ODBC drivers and the Microsoft Visual C++ redistributable packages requires administrator rights, so ensure that you have the appropriate privileges.

4. Run the Jade 2025 installer, available from https://www.jadeplatform.com/developer-centre/downloads.

   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.

5. 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. If the validation needs to be run again, see the **ReadmeInstallSteps.pdf** file in the documentation directory for instructions.

6. 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. The **ReadMe.txt** file contains late-breaking important information not possible to publish in this document.

7. Use the Jade Database utility to take a full backup of your Jade 2025 database.

Running Two Releases of Jade on the Same Workstation

You can have any number of releases of Jade installed on the same workstation. If ODBC is installed, only the last installation of the Jade ODBC driver is available from the ODBC Data Source Administrator.

Jade Thin Client Upgrade

When upgrading a presentation client to Jade release 2025, 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 2025. 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 (x86)** 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 *Thin Client Guide*.

Upgrading an SDS Native or RPS Secondary System

SDS secondary databases can be upgraded. For details about how to do this, see the **ReadmeInstallSteps.pdf** file in the **\documentation** directory.

Note RPS secondary tracking will be paused due to a binary mismatch only when the primary database is upgraded to a major Jade version; not during upgrade to later service packs.

You should not operate in mixed binary mode between primary and secondary databases at any time.

Copyright 2025, Jade Software Corporation Limited All rights reserved

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 Platform development environment, asking if validation should be re-run.

To perform the validation from the command line, see the **ReadmeInstallSteps.pdf** file in the **\documentation** directory.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Jade 2025 Changes that May Affect Your Existing Systems

This section describes only the changes in the Jade 2025 release that may affect your existing systems. Some changes may result in compile errors during the load process, or cause your Jade release 2025 systems to behave differently.

Change to Security Hooks for the Unit Test Runner Form

The task name used in the security hook that is called when launching the Unit Test Runner form from within the menu options of the Jade Platform development environment has been changed. This enables finer-grained control, specifically allowing administrators to prohibit access to the Unit Test Runner interface.

Previously, when opening the Unit Test Runner form, a security hook was called with one of the following task names, which depended on whether the Unit Test Runner form was started from the Jade Debugger.

- **executeIt**
- **debugExecute**

These hooks were also used for executing **JadeScript** and Workspace methods, making it impossible to distinguish between script execution and launching the Unit Test Runner form.

From this release, the Unit Test Runner form now invokes the security hook using one of the following dedicated task names.

- **runUnitTests**, when opening the Unit Test Runner form normally.
- **runUnitTestsDebug**, when opening the Unit Test Runner form under the Jade Debugger.

Additionally, the entity name passed to the hook now reflects the context from which the Unit Test Runner form was opened, as shown in the following table.

Opened From	Entity Name	Format Example
Schema Browser	*Schema*	Accounting
Class within a schema	*Schema::class*	Accounting::Invoice
Specific method	*Schema::class::method*	Accounting::Invoice::calculateTax

After the Unit Test Runner form is opened, you can execute any unit test methods that are displayed in the Unit Test Runner form.

Granular control over individual unit test execution is not supported through hooks. You cannot block execution of specific tests after the Unit Test Runner form has been launched.

However, using the new **runUnitTests** and **runUnitTestsDebug** task names, you can now block users from opening the Unit Test Runner form from the Jade Platform development environment menu options without blocking access to the execution of **JadeScript** and Workspace methods.

InstanceType Pseudo Type Error Instead of Warning (PAR 69245)

Using the **InstanceType** pseudo type outside of methods defined on the **Class** class or subclasses will now cause a 6155 (*Invalid use of pseudotype*) compilation error instead of a 10015 (*Invalid Use of InstanceType*) warning.

Copyright 2025, Jade Software Corporation Limited All rights reserved

ISO 8601 Format of Timestamps

The format for an ISO timestamp is **HH:MM:SS.sss** (where **sss** is the 3-digit millisecond count).

In earlier releases, the format produced by the **JadeJson** and **Jade RestService** classes was incorrectly specified as **HH:MM:SS:sss** ; that is, with a colon (**:**) character used as the separator between seconds and milliseconds. Note the difference is that the period (**.**) is the correct character separating seconds and milliseconds, *not* the colon (**:**). See https://www.iso.org/iso-8601-date-and-time-format.html and https://en.wikipedia.org/wiki/ISO_8601.

Caution If your Jade system or any applications with which it interacts consumes data generated from the **JadeJson** class or by Jade REST application responses that rely on the colon (**:**) character being used as a separator between seconds and milliseconds for ISO 8601 timestamps, make changes *before* you upgrade your Jade system to address this so that the period character (**.**) separator is used.

This release corrects an error in the ISO 8601 format for timestamps, as follows.

- For output generated by the **JadeJson** class **generateJson** and **generateJsonFile** methods when the value of the **useISO8601** property is set to **true**, if you use any of the following primitive type or object instances as the source parameter, the format will be corrected. (The **generateJsonDynamic** method is unaffected.)
  - **TimeStamp**, **Time**, and **TimeStampOffset** primitive types
  - Any objects that have properties of the **TimeStamp**, **Time**, and **TimeStampOffset** primitive types
  - Any objects that have references to these objects directly or indirectly (that is, through multiple references)
- Responses generated by Jade REST applications for which the **Use ISO 8601 Time** check box is selected on the **Web Options** sheet of the Define Application dialog.
- **JadeRestService** class methods that return any of the above primitive type or object instances will have their format corrected.

The output has been corrected to the ISO timestamp format; that is, **HH:MM:SS.sss**.

JSON Web Tokens Minimum Length Increase (PAR 69953)

The length requirements for the secret strings used to encode and validate JSON Web Tokens have been updated to match the JSON Web Algorithms standard (https://datatracker.ietf.org/doc/html/rfc7518#section-3.2).

If your secret string is too short, this change could cause your REST web service authentication to fail. To avoid this, you must regenerate the JSON Web Tokens with a new stricter secret length; that is:

- 256-bit secret for the HS256 algorithm
- 384-bit secret for the HS384 algorithm
- 512-bit secret for the HS512 algorithm

The following table lists the **JadeJsonWebToken** class methods that have been updated with a longer minimum length for the secret string.

method	New Minimum Length (characters)	Old Minimum Length (characters)
encodeHS256	32	16
encodeHS384	48	16
encodeHS512	64	16

In addition, the **JadeJsonWebToken** class **Error_SecretTooShort** and **MinSecretLength** constants have been deprecated as each method now has a specific minimum length.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Minimum and Maximum Server Threads

The **MinServerThreads** and **MaxServerThreads** parameters in the [JadeServer] section of the Jade initialization file now enable you to specify minimum, default, and maximum values as follows.

- The **MinServerThreads** parameter now has the following values.
  - Minimum value is **20**
  - Default value is **100**
  - Maximum value is **4,095**
- The **MaxServerThreads** parameter now has the following values.
  - Minimum value is **20**
  - Default value is **500**
  - Maximum value is **4,095**

Note In earlier releases, the minimum number of server threads in the **MinServerThreads** parameter was **1** and it is now **20**, so you may want to change your values of those parameters, as threads in the Jade database pool on the server grow dynamically.

These parameters replace those documented in "Server Thread Initialization File Parameters", under "Deimplementations and Deprecations", earlier in this document.

OpenSSL Upgrade from Version 1.0.2 to 3.0.8 (PAR 69321)

The major version of OpenSSL that the Jade Platform uses for the **TcpIpConnection** class and application server to presentation client encryption is no longer supported by the OpenSSL Project. This release upgrades OpenSSL to 3.0.8.

There should be no effect to the majority of systems; however, there are some cases where changes may need to be made. If:

- You are using a certificate that has a weak, unsupported cipher

  This would be likely only if you use old self-signed certificates, in which case you would have to regenerate the certificate. The new certificate will work for Jade without the upgrade so you can test the new certificate (if you require one) before upgrading Jade.

- You have configured Jade to use ciphers that are no longer supported

  You must edit this custom configuration as there is no way to enable weak, unsupported ciphers. The **jommsg.log** file displays the name of the unsupported cipher you are attempting to use.

  An example of this configuration would be the **SSLCipherNames** option in the **jade.ini** file that can be set to a value of **<default>** for maximum compatibility.

- You have defined external functions to **ssleay32.dll** or **libeay32.dll**

  The names of these OpenSSL libraries have changed to **libssl-3-x64.dll** and **libcrypto-3-x64.dll** for 64-bit and to **libssl-3.dll** and **libcrypto-3.dll** for 32-bit respectively, so you must change the library name for these definitions in Jade.

Output Parameter Initialization (PAR 68649)

Output parameters of type **Date**, **Time**, **TimeStamp**, and **TimeStampOffset** are initialized to the current date, time, or date and time value when the called method begins execution.

Copyright 2025, Jade Software Corporation Limited All rights reserved

When passing a **Date**, **Time**, **TimeStamp**, or **TimeStampOffset** property into an output parameter of the same type or **Any** type in earlier releases, the default assigned value differed between Jade method calls and external method calls if the called method did not explicitly assign anything to the parameter before returning.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Changes and New Features in Jade Release 2025.0.01

This section summarizes the product and documentation changes and new features in Jade release 2025.0.01. For details about the changes in Jade release 2025 that may affect your existing systems, see "Jade 2025 Changes that May Affect Your Existing Systems", earlier in this document.

Additional File Extension Appended to webFileName Property (PAR 69988)

A change in the logic in Jade 2022.0.01 caused images with a **webFileName** property to have an additional file extension appended, depending on the value specified in the **ImageType** parameter in the [WebOptions] section of the Jade initialization file; for example, where the **ImageType** parameter value is **.png**, a **webFileName** property of **myImage.gif** would be assigned the name **myImage.gif.png**.

The logic now ensures that the value specified for the **webFileName** property is honored.

A file extension is appended only if a valid one is not specified for the **webFileName** property. The appended file extension is derived from the **ImageType** parameter value.

In addition, the **ImageType** parameter now supports **png** (as well as **.png**) as a valid value and no longer supports **.gif** (Graphics Interchange Format).

API Optimization

This section describes the API optimization enhancements in this release.

Dynamic Worker Pool Scaling

Jade configuration settings affect the operation of the dynamic worker pool used by web server applications such as REST servers and SOAP servers. These web applications can be configured in either the traditional HTTP access (via IIS and the **jadehttp.dll** ISAPI extension) or the newer direct HTTP mode.

You can configure a pool of workers to be dynamically scaled in response to changes in the incoming communication load. When the incoming rate of requests exceeds the response capacity of the server, the Jade application detects this situation and adds processing capacity by spawning another worker application.

An incoming request is added to the request queue if there are no workers available to process the request. If the number of queued requests continuously remains above the queue depth threshold for the configured period, an internal **JadeMultiWorkerTcpManager** class, which manages TCP connections to a Jade application, adds a worker to the pool if the number of current workers is less than the configured maximum number of workers. Each *worker* in the pool is a complete single-threaded Jade application, running on the same Jade node as the original Jade application that spawned it.

If a worker process is idle for more than the configured timeout period, it is removed from the worker pool if the total number of workers is greater than the configured minimum number of workers.

The scaling of the pool of workers is controlled by the following initialization file settings.

- The [WebOptions] section of the Jade initialization file (**jade.ini**) can now contain the following parameters for Jade applications to directly control the scaling of the dynamic worker pool.
  - MaximumInUse
  - MinimumInUse
  - QueueDepthLimit
  - QueueDepthLimitTimeout
  - WorkerIdleTimeout

Copyright 2025, Jade Software Corporation Limited All rights reserved

- The **MaxInUse** parameter in the [*application-name*] section of the **jadehttp.ini** file affects the dynamic worker pool scaling. This setting does not control the scaling, but it can prevent the scaling from working if it is set too low.

  This parameter is relevant only for web applications configured for traditional HTTP access (via IIS and the **jadehttp.dll** ISAPI extension).

- The specified number of application copies in the **Application Copies** text box on the **Web Options** sheet of the Define Application dialog provides the Jade node with the starting number of worker instances.

For more details, see "Scaling Dynamic Worker Pool Connections", in Chapter 2 of the *Installation and Configuration Guide*.

JadeAnyArray Class

Jade Platform now provides the **JadeAnyArray** class, which is an **Array** subclass.

JadeJson Class Formatting Options for Generated Output

From this release, you can specify whether to apply indented formatting to **JadeJson** class dynamically generated output.

The **JadeJson** class now provides the properties listed in the following table.

Property	Description
generationStyle	Specifies what style to apply to dynamically generated output
generationStyleIndentCharacter	Defines the indent character used when formatting dynamically generated output
generationStyleIndentDepth	Defines how many indent characters are used to represent each indent when formatting dynamically generated output

The formatting styles that can be applied to the dynamically generated output are represented by the **JadeJson** class constants listed in the following table.

Constant	Description	Integer Value
Style_Compact	Compact (that is, unformatted) output	0
Style_Indented	Indented output	1

For details about additional new properties for the **JadeJson** class, see "JadeJson Class modifyReadOnlyProperties Property (PAR 69760, 69896)", later in this document.

JadeJson Class modifyReadOnlyProperties Property (PAR 69760, 69896)

The **JadeJson** class now provides the **modifyReadOnlyProperties** property, which specifies whether read-only properties are populated with their values from the JSON text.

By default, the **JadeJson** class **parse** method does not modify read-only and protected properties of the object type of the data class that is being created; that is, the value of the **modifyReadOnlyProperties** property is **false**, by default.

The **readOnly** property is honored in **JadeRestDataModelProxy** classes generated using the OpenAPI External Component Browser, but these properties are not populated when returning them from **JadeRestResourceProxy** class endpoints unless you set the value of the **modifyReadOnlyProperties** property in the **JadeJson** class to **true**.

Note To modify existing generated source code in your schema or schemas, re-generate the OpenAPI proxy classes when this property is set to **true**.

Copyright 2025, Jade Software Corporation Limited All rights reserved

For details about additional new properties for the **JadeJson** class, see "JadeJson Class Formatting Options for Generated Output", earlier in this document.

JadeJson Class parseAllowComments Property

The **JadeJson** class now provides the **parseAllowComments** property, which specifies whether comments are allowed when parsing JSON using the **parseDynamic** method. The default value is **false**.

If the value of the **parseAllowComments** property is set to **true** when parsing JSON using the **parseDynamic** method, JSON input with comments is accepted as valid. Comments are ignored by the parser and will not change the output of the method.

JadeJsonObject and JadeJsonObjectIterator Classes

The following classes have been added.

- **JadeJsonObject**, which stores an accurate representation of a JSON data structure within a Jade system. The class is transient-only and each instance of the class is valid only for the node on which it was originally created.

  The **JadeJsonObject** class has the methods listed in the following table.

Method	Description
createIterator	Returns an implementation of the **JadeIteratorIF** interface that can be used to iterate through the property values. The returned object that implements the **JadeIteratorIF** interface is guaranteed to be of type **JadeJsonObjectIterator**, which can be used to iterate through the property names and also the property values with its **currentPropertyName** method.
getJsonProperty	Gets the value of the specified JSON property.
getJsonPropertyArray	Gets the value of the specified JSON **array** property.
getJsonPropertyBoolean	Gets the value of the specified JSON **boolean** property.
getJsonPropertyCount	Returns the number of JSON properties on the JSON object.
getJsonPropertyNull	Returns **true** if a null-type JSON property with a name matching the **name** parameter is found.
getJsonPropertyNumber	Gets the value of the specified JSON **number** property.
getJsonPropertyObject	Gets the value of the specified JSON **object** property.
getJsonPropertyString	Gets the value of the specified JSON **string** property.
setJsonPropertyArray	Sets the value of the specified JSON **array** property.
setJsonPropertyBoolean	Sets the value of the specified JSON **boolean** property.
setJsonPropertyNull	Sets the JSON property with the name specified in the name parameter to a null value.
setJsonPropertyNumber	Sets the value of the specified JSON **number** property.
setJsonPropertyObject	Sets the value of the specified JSON **object** property.
setJsonPropertyString	Sets the value of the specified JSON **string** property.

Copyright 2025, Jade Software Corporation Limited All rights reserved

- **JadeJsonObjectIterator**, which implements the **JadeIteratorIF** interface and allows you to iterate through the property names and values of its matching object.

  The **JadeJsonObjectIterator** class has the methods listed in the following table.

Method	Description
current	Retrieves the current JSON property value in the iterable sequence of JSON properties.
currentPropertyName	Retrieves the current JSON property name in the iterable sequence of JSON properties.
next	Advances the iterator to the next JSON property in the sequence and retrieves the value of that property.

The following table lists methods have been added to the **JadeJson** class.

Method	Generates...
parseDynamic	JSON based on the hierarchy of **JadeJsonObjects** and **JadeAnyArrays** of which the passed in **source** parameter is the root.
generateJsonDynamic	A series of linked **JadeJsonObject** objects and **JadeAnyArray** arrays that accurately represent the passed in JSON data structure.

The **Any** primitive type now provides the **hasValue__** method, which returns **true** if a value has been assigned to the receiver.

JadeRestDataModelProxy Class _additionalProperties Property (PAR 69760, 69933)

The **_additionalProperties** property of the **JadeRestDataModelProxy** class contains additional JSON properties that are intended for the generated **JadeRestDataModelProxy** subclass but which did not exist in the OpenAPI specification.

Note The additional properties were not defined in the imported OpenAPI specification but are supplied in JSON response text when you use your OpenAPI client.

JadeRestObjectArrayProxy Class

Jade Platform now provides the **JadeRestObjectArrayProxy Class** class, which is an **ObjectArray** subclass.

The **JadeRestObjectArrayProxy Class** class is a grouping class for auto-generated proxy classes that model the collections of an imported REST API specification.

The **objectsToBeDeleted** property is now automatically added to subclasses generated for the **JadeRestResourceProxy** class. When you delete an OpenAPI definition, this property enables Jade to remove associated subclasses of the **JadeRestObjectArrayProxy** class in addition to all associated classes.

Objects within the **objectsToBeDeleted** collection property are deleted when the **JadeRestResourceProxy** class is deleted. Objects returned by generated methods on your **JadeRestResourceProxy** classes are added to this collection to help manage the lifetime of these objects.

JSON Web Tokens Minimum Length Increase (PAR 69953)

The length requirements for the secret strings used to encode and validate JSON Web Tokens have been updated to match the JSON Web Algorithms standard (https://datatracker.ietf.org/doc/html/rfc7518#section-3.2).

Copyright 2025, Jade Software Corporation Limited All rights reserved

If your secret string is too short in the **encodeHS256**, **encodeHS384**, or **encodeHS512** methods, this change could cause your REST web service authentication to fail. For details, see "JSON Web Tokens Minimum Length Increase (PAR 69953)", earlier in this document.

Application Class generateUuidDefault Method

The **Application** class now provides the **generateUuidDefault** method, which has the following signature.

```
generateUuidDefault(): Binary;
```

This method generates and returns a binary Universally Unique Identifier (UUID) value, which can be used as an attribute of an object so that it can be exposed to third-parties.

This method generates the recommended UUID variant defined in the **generateUuid** method, which is **VariantMicrosoft**.

Note The recommended UUID variant has changed from **VariantDce** to **VariantMicrosoft**, which is more unique.

Azure SQL Support for RPS Nodes (JAD-I-525, PAR 69475)

From this release, RPS supports targeting managed Azure SQL Database and Azure SQL Managed Instance services. Consult the Microsoft Azure documentation for the versions and features that are supported.

See also "SQL Server 2022 Support for RPS Nodes (PAR 69259)" and "Loading Bulk Data into an Azure SQL Database", elsewhere in this document.

Collection Concurrency Phase 2 (JAD-I-423)

Phase 2 of the collection concurrency enhancements continues to make it easier to write code that maintains persistent collections either directly or as a side-effect of inverse maintenance in a manner that minimizes contention and avoids deadlocks.

Prior to this release, deferred add and remove operations were not visible to the calling process until after the enclosing transaction was committed. For example, if an object was added to a collection using a deferred operation, the **includes** method returned **false** until the transaction was committed. After the commit, the **includes** method returned **true**, as expected. Similarly, if a collection was iterated, using a **foreach** instruction or an iterator deferred operations were not taken in to account.

From this release, you can change this behavior so that the effects of deferred operations on collections will be visible before the updates are committed to the database. This allows any subsequent actions dependent on those collection updates in the transaction to proceed successfully.

The Jade initialization file can now contain the **DefaultIteratorsIncludeDeferredOperations** parameter in the [JadeClient] section, which enables you to specify at the system level whether collection methods and iterators take into account the effects of deferred operations on collections. This parameter defaults to **false**; that is, the current behavior is retained.

The **Process** class now provides the **iteratorsIncludeDeferredOperations** method, which enables you to override the value of the **DefaultIteratorsIncludeDeferredOperations** parameter for the current process.

Calling a method that does not take into account deferred operations on a collection with deferred updates raises exception 1479 (*Operation not supported when Collection has deferred updates*). This applies to the methods listed in the following table.

Class	Method
Collection	deleteIfEmpty
Array	rebuild
Dictionary, Set	rebuild

Copyright 2025, Jade Software Corporation Limited All rights reserved

Class	Method
	indexNear
	indexNear64
	indexOf
	indexOf64
DictIterator, SetIterator	startAtIndex
	startNearIndex

As part of this change, the following methods are deprecated.

Class	Deprecated Method	Method to Use Instead
Collection	includesWithDeferred	includes
Dictionary	getAtKeyWithDeferred	getAtKey
	includesKeyWithDeferred	includesKey

Caution When you use a method listed in the **Method to Use Instead** column instead of the deprecated method, ensure that you enable visibility to the effects of deferred collection operations by using the **DefaultIteratorsIncludeDeferredOperations** initialization file parameter or the **iteratorsIncludeDeferredOperations** method.

commandFile Loader Argument New Commands (JAD-I-457)

The following new commands are available to the **commandFile** argument for the **jadloadb** batch Schema Load utility and non-GUI **JadeSchemaLoader** application.

- Delete Library *schema-name*::*library-name*
- Rename Application *schema-name*::*existing-application-name new-application-name*
- Delete Application *schema-name*::*application-name*
- Rename Package *schema-name*::*existing-package-name new-package-name*
- Delete ExportedType *schema-name*::*exported-package-name*::*exported-type-name*
- Delete ExportedFeature *schema-name*::*exported-package-name*::*exported-type-name*::*exported-feature-name*
- Delete Exposure *schema-name*::*exposure-name*

Configuring a Secondary RPS Database for Azure (PAR 70353)

The [JadeRPS] section of the Jade initialization file can now contain the following parameters to configure a secondary RPS database for Azure.

- **UseAzureSQLDatabase**, which specifies whether the RPS database is an Azure SQL database. The default value is **false**; that is, the RPS database is an Azure SQL database.
- **AzureAuthenticationMethod**, which specifies what authentication method should be used in the generated Bulk Copy Program (BCP) scripts. The default value is **ActiveDirectoryIntegrated** that is, Active Directory (AD) authentication.

Copyright 2025, Jade Software Corporation Limited All rights reserved

- **DSNName**, which specifies the name of the Data Source Name (DSN) if the **AzureAuthenticationMethod** parameter is set to **DSNDefined**.
- **AzureBcpMaxErrors**, which specifies what value to use for the **-m** flag in the BCP script. The default value is zero (**0**).

The initialization file parameters in the following example generate BCP scripts that use Azure Active Directory (AD) authentication (that is, **-G**), set the max row level errors to **5**, and output a **bcp_errors.log** file.

```
[JadeRPS]
UseAzureSQLDatabase=true
AzureAuthenticationMethod=ActiveDirectoryIntegrated
AzureBcpMaxErrors=5
```

The initialization file parameters in the following example generate BCP scripts that use the authentication method defined in the DSN (that is, **-D**), set the max row level errors to zero (**0**), and output a **bcp_errors.log** file.

```
[JadeRPS]
UseAzureSQLDatabase=true
AzureAuthenticationMethod=DSNDefined
DSNName=SystemDSN
```

Note The **bcp_errors.log** file is output if the conditions specified above are met. If there are no errors, the file is still output, but is zero (**0**) bytes in size.

The documentation for the flags that have been added to the BCP scripts (that is, **-D**, **-e**, and **-m**), can be found at the following URL.

- https://learn.microsoft.com/en-us/sql/tools/bcp-utility?view=sql-server-ver17&tabs=windows

The new BCP flags are set only if the **UseAzureSQLDatabase** initialization file parameter has been set to **true**; this is to avoid changing the behavior of any deployed or existing RPS systems.

Converting CCD Files to CSV Files (JAD-I-763)

The **JadeProfiler** class now provides the **codeCoverageCcdToCsv** method, which converts code coverage files (.**ccd**) to comma-separated values (.**csv**) format files.

The **codeCoverageCcdToCsv** method has the following signature.

```
codeCoverageCcdToCsv(ccdPath: String;
                     csvPath: String): Boolean typeMethod;
```

Default Replayable Option Removed from Load Schema Source Dialog (PAR 69828)

The **Default** option has been removed from the **Replayable** combo box on the Load Schema Source dialog. This option was removed to reduce errors. The default value of the **Replayable** combo box is now **True**.

Accessibility Enhancements

This section describes the accessibility enhancements in this release.

List Box and Table Accessibility Enhancements (PAR 69511)

The [Jade] section of the Jade initialization file can now contain the **AccessibilityUIAEnabled** parameter, which specifies whether Jade enables UIA (Microsoft UI Automation) accessibility features for **ListBox** and **Table** controls. The parameter is set to **false** by default.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Menu Accessibility Enhancements (PAR 69691)

Menu bars on forms and menu items have been updated to support accessibility. The Jade Platform has been changed so that menu bars now expose their location for MDI frame forms and non-MDI forms. In addition, the name, location, and state are now exposed for each menu item in the hierarchy of a menu bar.

The MenuBar is now exposed as a direct child of a Form. Menu items are children of the menu bar. In multi-hierarchical menus, a menu item may have multiple menu item children.

This feature is enabled by setting the **EnhancedAccessibilityEnabled** parameter to **true** in the [Jade] start-up section of the Jade initialization file.

Note The **AccessibilityEnabled** parameter also must be set to **true**.

The following restrictions apply.

- This feature has been implemented using Microsoft Active Accessibility (MSAA). Modern accessibility clients must support MSAA to be able to view the exposed menu bar and menu items.
- System menus, such as Microsoft Window list items and the top-left system menu of a form are not supported.

Scroll Bar Accessibility Enhancements (PAR 69511)

Scroll bars and spin-style (**Style_SpinBox**) combo boxes have been updated to support accessibility. The Jade Platform has been changed so that scroll bars and spin-style (**Style_SpinBox**) combo boxes now expose each of their child elements including names, locations, and values.

Table Accessibility Enhancements (PAR 69426)

Tables have been enhanced to better support accessibility and automation. The Jade Platform has been changed so that:

- Table cells will now expose **"Cell (*row*, *column*)"** as the value of the **Name** property and their content as the **Value** property.
- Table cells will now expose their correct location within the table.
- Table rows will now expose **"Row (*row*)"** as the value of the **Name** property and space-delimited cell content as the **Value** property.
- Table rows will now expose their correct location within the table.
- Combo box list items (including spin box controls) now expose the correct location.
- Table row and cell selection now more-closely resembles Jade selection logic, as follows.
  - If the value of the **Table** class **selectMode** property is **SelectMode_CurrentRow** or **SelectMode_WholeRows**, a change in row selection will result in the entire row gaining focus.
  - When a row has gained focus however, changing the column or clicking on the same cell will then cause that cell to gain focus, which allows for the selection of a single cell.
- If a cell in a fixed column gains focus and the value of the **Table** class **selectMode** property is one of the following values, the entire row gains focus.
  - SelectMode_Default
  - SelectMode_FixedColumn
  - SelectMode_CurrentRow
  - SelectMode_WholeRows

Copyright 2025, Jade Software Corporation Limited All rights reserved

Tips It is recommended that if accessibility is to be activated and remain able to be tested, **Table**, **ComboBox**, and **ListBox** controls that may contain over approximately 1,000 entries should be populated with the **Collection** class **displayCollection** method. This ensures that the initial accumulated cost of querying children is limited to the number of entries displayed.

External applications that utilize accessibility (such as the Windows Inspect tool) and call Jade's accessibility framework may result in an accessibility object being created for each child. This can have a negative impact on the performance of the external application.

Note This table accessibility feature affects only accessibility focus and selection; the underlying Jade logic has not changed.

Direct SOAP HTTP Access

A Jade SOAP application is a Jade application with the **Web-Enabled** or **Web-Enabled, Non Gui** application type. These applications expose SOAP endpoints by adding methods to a user implemented subclass of the **JadeWebServiceProvider** class.

Before the implementation of the Direct SOAP feature, access to web service applications was available using only Internet Information Server (IIS). Client applications would make HTTP/1.1 web service requests to IIS, and then IIS would communicate with the Jade web service application to serve responses to these requests. The communication between the IIS and the Jade web service application was performed by the Jade Internet Server Application Programming Interface (ISAPI) extension Dynamic Link Library (DLL) called **jadehttp.dll**. This DLL implemented a proprietary Jade protocol between IIS and the web service applications.

The Direct SOAP feature adds support for the HTTP/1.1 communication protocol to SOAP web service applications. This means that Jade web service applications no longer require IIS to enable communication with other HTTP-compliant applications (for example, SOAP clients, web browsers, reverse proxies, web servers, or TLS termination servers).

This allows a greater degree of freedom for the deployment of Jade SOAP web service applications and it also reduces the system requirements for the development of SOAP web service applications.

Event Streaming

The following enhancements have been made to the Event Stream Producer (ESP) and associated Change Data Capture (CDC) mechanisms.

- A **HealthMonitor** section can now be configured in the ESP configuration to log or report health check data by using an event topic or REST endpoint.
- Comments are now supported in the ESP configuration JSON files.
- A separate Change Capture specification file enables you to specify which classes should be included in the change capture.
- Support for streaming String Large Objects (slobs) and Binary Large Objects (blobs).
- Bulk load capability to produce events matching the current database state.
- Higher throughput by multithreading the ESP and the batching of events.
- More authentication types are supported for Azure, Confluent, and Kafka.

Jade Initialization File

This section describes the Jade initialization file changes in this release. See also, "Configuring a Secondary RPS Database for Azure" and "Minimum and Maximum Server Threads", elsewhere in this document.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Disabling Accessibility Enhancements (PAR 69709)

The [Jade] section of the Jade initialization file can now contain the **EnhancedAccessibilityEnabled** parameter, which specifies whether Jade enables enhanced accessibility features; for example, table and scroll bar accessibility.

Set the parameter value to **false** to disable enhanced accessibility features and retain your current behavior.

Note To use enhanced accessibility features, the value of the **AccessibilityEnabled** parameter in the [Jade] section of the initialization file must also be set to **true**.

Disk Cache Parameters in the [PersistentDb] Section

In earlier releases, the maximum and minimum memory was specified in terms of the number of segments allocated. Jade 2025 now automatically allocates the number of segments as close as possible and not exceeding the specified values of the new **DiskCacheMaxMemory** and **DiskCacheMinMemory** parameters in the [PersistentDb] section of the Jade initialization file.

All previous settings of the legacy **DiskCacheMaxSegments** and **DiskCacheMinSegments** parameters from earlier Jade releases are automatically converted to the equivalent **DiskCacheMaxMemory** and **DiskCacheMinMemory** parameter values using the **DiskCacheBlocksPerSegment** parameter calculation.

Because the memory settings are not present when the Jade initialization file is first read in Jade 2025, Jade calculates the equivalent memory values from existing segments values, and writes them into the Jade initialization file. The **DiskCacheMinMemory** and **DiskCacheMaxMemory** parameters are always created if they were not already present. If the segments settings were not present or they had the **<default>** value, the new memory settings will be written as **<default>**.

Logging Normal TCP/IP Disconnections

The default value of the **LogTcpNormalDisconnects** parameter in the [FaultHandling] section of the Jade initialization file has changed from **1** to zero (**0**), so that the logging of normal TCP/IP disconnections is now suppressed.

Using the .NET Core Run Time

Earlier Jade Platform releases supported the .NET Framework only. Jade now supports the .NET Core runtime extension to the .NET Component and .NET Import functionality.

The [Jade] section of the **jade.ini** file now provides the **UseDotnetCoreRuntime** parameter, which enables you to execute .NET Core runtime functionality, including:

- Extending your applications by adding .NET controls to your forms
- Using the .NET Regex search engine
- Using existing .NET libraries for parsing and modifying Word documents, and so on

When the parameter is set to **false**, the .NET Framework is used. When it is set to **true**, libraries are imported into Jade using the .NET run time Reflection and **JadeDotNetType** proxies use the .NET runtime.

If you have set the **UseDotnetCoreRuntime** parameter to **true**, you can use .NET Framework 4.x components or any .NET (.NET Core) components up to and including .NET 8.

Using Dedicated Application Client Connections

When using TCP/IP for nodes connecting to the Jade Remote Access program (**jadrap**), the **UseDedicatedApplicationConnections** parameter in the [JadeClient] section of the Jade initialization file allows each Jade application to have its own dedicated TCP/IP connection rather than being shared by all applications on the node. This parameter is set to **true** by default.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Tip This can remove a bottleneck in performance during high activity on the node, because one application only can use the connection at any time.

Web Process Timeout Enhancements (PAR 68982)

The [WebOptions] section of the Jade initialization file can now contain the **TimeoutSentinel** parameter, which specifies whether the timeout sentinel is enabled for web applications.

If this parameter is set to **true**, when a web application is started, the web timeout sentinel application will be started if an instance of the application is not already running in that schema. The web timeout sentinel application acts as an arbiter for web requests and interrupts requests that exceed the configured timeout limit.

The default value of this parameter is **false**, which is the pre-existing behavior.

Jade Platform Development Environment

This section describes the Jade Platform development environment changes in this release. (See also "Accessing Call Stack Information in the Development Environment", elsewhere in this document.)

ATCG Thin Client Method Broadcast Support (PAR 69314)

When running in thin client mode, the Automated Test Code Generator (ATCG) did not record a subset of unimplemented control and form event methods. This subset is defined by internal method masks. For thin clients, the event methods are optimized and are not sent to the application server for execution if unimplemented; ATCG remained unaware that they were called so could not record them.

You can now specify which event methods to omit from this optimization when running ATCG in thin client mode; that is, you can force the broadcasting of unimplemented event methods to the application server so that ATCG can record them.

The following new **Application** class methods enable you to force the broadcasting of unimplemented event methods when using ATCG in thin client mode.

- **isThinClientMethodBroadcastForced**, which returns **true** if the broadcast has been forced for the method specified in the **eventMethodName** parameter. This method has the following signature.

```
isThinClientMethodBroadcastForced(cls: Class;
                                  eventMethodName: String): Boolean
```

- **forceThinClientMethodBroadcast**, which marks a single event method to start broadcasting to the application server, even when unimplemented. This method has the following signature.

```
forceThinClientMethodBroadcast(cls: Class;
                               eventMethodName: String): Boolean
```

- **resetThinClientMethodBroadcast**, which clears the internal collection of broadcasting event methods. This method has the following signature.

```
resetThinClientMethodBroadcast()
```

Notes You must invoke a method in the previous list prior to the creation of a form you want it to affect.

The methods have no effect on existing forms.

If you have been using the ATCG thin client kit provided by Jade Support, you should use the kit to uninstall the methods that were generated by the kit.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Editor Shortcut Key Category Changes

The following changes have been made to the keystroke and actions of the editor shortcut key category.

- F11

  The F11 keystroke **ShowSymbolInfo** action behavior has been extended so that if the symbol on which the caret is positioned is:

  - A method, a new Class Browser is opened with the method selected in the Methods List (unless the **Reuse Same Method Source Window For All** check box is not checked on the **Source Management** sheet of the User Preferences dialog).
  - Not a method, information is displayed, which includes the creation timestamp and patch version of the entity.

  The previous behavior (which displayed information for the current symbol) is now associated with the **LegacyShowSymbolBehaviour** action and has the Alt+F11 keystrokes.

  Note To retain the previous behavior, rebind the previous behavior (that is, the **LegacyShowSymbolBehaviour** action) to F11 in the **Short Cut Keys** sheet of the User Preferences dialog.

- Shift+F11

  The Shift+F11 keystroke is now associated with the new **OpenBrowserAndNavigateToRootDefinition** action, which opens the item under the caret in the appropriate browser; for example, if the item is a:

  - Class, a new Class Browser is opened with the class selected in the Class List.
  - Class attribute property, a new Class Browser is opened with the attribute selected in the Properties List. Press F11 to open the attribute type in the Primitive Types Browser.
  - Reference property, a new Class Browser is opened with the reference selected in the Properties List. Press F11 to open the reference type in the Primitive Types Browser.
  - Translatable string, the String Browser is opened with the translatable string selected.
  - Function, the External Functions Browser is opened with the function selected.

  The previous behavior (which displayed a new Class Browser) is now associated with the **LegacyOpenBrowserBehaviour** action and has the Alt+Shift+F11 keystrokes.

  Note To retain the previous behavior, rebind the previous behavior (that is the **LegacyOpenBrowserBehaviour** action) to Shift+F11 in the **Short Cut Keys** sheet of the User Preferences dialog.

Extracting Compact DDX Files

You can now extract and load compact DDX files. Compact DDX files reduce the size of the extracted file and improve loading performance. The compact DDX extract files exclude properties with a null value for applications, forms, standard controls, and RPS mappings.

As part of this change, the **Schema Options** sheet is now available on the Extract dialog when extracting RPS mappings.

Note Compact DDX files extracted in Jade 2022.0.02 and higher will not load into Jade 2022.0.01 or any earlier Jade releases.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Compact DDX files include the **Compact="true"** tag and value in the second line of the DDX file; for example:

```
<schema name="Schema1" JadeVersionNumber="2022.0.02" Compact="true"
JadePatchNumber="0" CompleteDefinition="true">
```

The **Schema Options** sheet of the Extract dialog now displays the **Compact DDX** check box, which is unchecked by default. Check this check box to extract a compact DDX file. The **Compact DDX** check box is enabled only if you are extracting a DDX-format file.

The **JadeBatchExtract** application in the **jadclient** non-GUI client application now enables you to extract compact DDX files by including the **<compactddx>** option, as shown in the following example.

```
jadclient path=c:\jade\system ini=c:\jade\system\jade.ini schema=JadeSchema
app=JadeBatchExtract startAppParameters CL "c:\temp\WidgetForm.cls"
"c:\temp\WidgetForm.ddx" Sales Widget "<ddxFormat,compactddx,utf8Encoding>"
```

Note The **<compactddx>** option has no effect when extracting DDB (legacy) format files.

Extracting Properties and Constants (JAD-I-775)

You can now extract properties and constants from some browsers in the Jade Platform development environment.

The **Extract** command is available in the:

- Properties menu when you select a property in the Properties List of the Class Browser.

  Note If the extracted property has inverse references, these details are also extracted.

- Constants menu when you select a constant in the **Constants** folder in the Properties List of the Class Browser or Primitive Types Browser.

The property or constant is then extracted to a .**cls** file in the specified location.

The task names listed in the following table are available to the **jadeDevelopmentFunctionSelected** function.

Task Name	Entity Name	Description
extractConstant	*Schema-name::type-name::constant-name*	Extracts a constant
extractProperty	*Schema-name::class-name::property-name*	Extracts a property

Supplementary Browsers Available

The following new supplementary browsers are available in the Jade Platform development environment.

- Call Hierarchy Browser, which enables you to view all outgoing method calls in a hierarchical manner for a method selected in the Methods List of the Class Browser, Interface Browser, or Primitive Types Browser.
- Property Updates Browser, which enables you to view where a property selected in the Properties List of the Class Browser is being updated in the code.
- Related Tests Browser, which enables you to view what unit tests cover a method, class, interface, or primitive type selected in the Class Browser, Interface Browser, or Primitive Types Browser.

When opened, the browsers are displayed to the right of the editor pane in the current browser.

Copyright 2025, Jade Software Corporation Limited All rights reserved

The Call Hierarchy Browser, Related Tests Browser, and Property Updates Browser have a Browser Options pane that enables you to configure preferences and filter settings for searching within those browsers.

Regular Expressions in Search Dialogs

The following dialogs now enable you to use regular expression (Regex) pattern matching in searches.

- Global Search And Replace
- Find/Replace
- Search/Replace the Methods in the Methods List
- Search Translatable Strings

Check the new **Regex** check box, to specify a Regex pattern in the search combo box. If you check this check box, the **Separate Lines** check box is enabled. By default, the **Regex** check box is unchecked.

If you want to treat the search text as if it is broken up into separate lines, check the **Separate Lines** check box; for example, if you want to search for a specified string at the start or end of a line, check this check box. When this check box is checked, the search breaks the text into individual lines at each **CrLf** end-of-line sequence. By default, the **Separate Lines** check box is unchecked.

The following list shows examples of regular expressions that you can use in a Regex search. To search for:

- At least a 4- or 5-digit number assignment ignoring whitespace; for example, an error code

```
\s*:=\s*\d{4,5}
```

- Transaction methods

```
(abortTransaction|beginTransaction|commitTransaction)
```

- **unloadForm** in an epilog

```
epilog[\s\S]*unloadForm
```

- A variable declaration named **userName**

```
vars[\s\S]*\buserName\b[\s\S]*begin
```

Schema Extract File Format Change

When extracting a schema that contains an RPS mapping, a .**ddxx** format file is now output if you are using the default DDX format. If you are using the legacy DDB format, a .**ddbx** format file is output (the contents of this file are in XML format).

The .**ddxx** format was introduced as part of the DDX extract format changes in Jade release 2022.0.01.

Jade Report Writer Formatting Options for XML and Delimited Files (PAR 69967)

From this release, fields are formatted by default in Jade Report Writer (JRW) reports output to Extensible Markup Language (XML) file or delimiter-separated files.

On the **Output** sheet of the Report Properties dialog, the **Numbers in Printed Format** and **Dates in Printed Format** check boxes are now checked by default.

Note There is no change to any formatting options for JRW reports created before this release.

Copyright 2025, Jade Software Corporation Limited All rights reserved

The **JadeReportWriterReport** class now provides the following methods to format JRW reports for extraction to an XML file or delimiter-separated file.

- **setDelimitedOutputFormatOptions**, which has the following signature.

```
setDelimitedOutputFormatOptions(formatBooleans, formatNumerics, formatDates,
                                formatTimes, formatTimeStamps: Boolean);
```

  This method specifies whether to format each of the types (**Boolean**, **Date**, **Time**, **TimeStamp**, **Integer**, and **Integer64**) when running the report for extraction to a delimiter-separated file.

- **setXmlExtraOptions**, which has the following signature.

```
setXmlExtraOptions(reportHeaderTag:  String;
                   pageHeaderTag:    String;
                   formatDates:      Boolean;
                   formatNumerics:   Boolean);
```

  This method sets the values of the **reportHeaderTag** and **pageHeaderTag** parameters, and specifies whether to format the date and numeric fields when running the report for extraction to an XML file.

- **setXmlOutputFormatOptions**, which has the following signature.

```
setXmlOutputFormatOptions(formatBooleans, formatNumerics, formatDates,
                          formatTimes, formatTimeStamps: Boolean);
```

  This method specifies whether to format each of the types (**Boolean**, **Date**, **Time**, **TimeStamp**, **Integer**, and **Integer64**) when running the report for extraction to an XML file.

Jade Sentinel

This section describes the Jade Sentinel changes in this release.

Jade Sentinel Exit Values (PAR 69589)

Sentinel now returns the exit values listed in the following table.

Value	Description
0	Success, no error
1	Incorrect command line
2	Manual dump failed

Manually Initiating a Process Dump (PAR 69229)

You can now manually initiate Jade Sentinel to take a process dump of a specified process. For details, see "Jade Sentinel", in Chapter 2 of the *Installation and Configuration Guide*.

JadeTestCase Class

This section describes the **JadeTestCase** class changes in this release.

expectAbort Method

The **JadeTestCase** class now provides the **expectAbort** method, which enables you to register that a test method is expected to abort execution.

Copyright 2025, Jade Software Corporation Limited All rights reserved

expectTerminate Method

The **JadeTestCase** class now provides the **expectTerminate** method, which enables you to register that a test method is expected to terminate.

Loading Schemas

This section describes the schema load changes in this release.

Loading Transaction Agent Framework (TAF) Partial Extract Files

Chapter 3 of the *Development Environment User's Guide* now provides an overview of the Transaction Agent Framework (TAF) that is incorporated into the *Erewhon Demonstration System Reference* and instructions if you want to utilize the example TAF from the Erewhon demonstration system and load partial extract schemas into your own new schemas.

The Transaction Agent Framework is a recommended approach to persisting objects using the Jade Platform to create, update, and delete functionality in your own Jade applications.

loadOrder Argument (PAR 69801)

The **loadOrder** argument is now available to the GUI and non-GUI versions of the Schema Load utility (**jadload** and **jadloadb**, respectively).

The following **loadOrder** argument options enable you to specify the load order of schema files in a command script.

- **minimizeReorgs** (the default value) loads all .**scm** files before any .**ddb** or .**ddx** files.
- **verbatim** loads the schema files in the order specified by the .**mul** file. If a .**scm** and .**ddx** or .**ddb** pair are specified on one line, they are treated as a pair of files.

Note There is an interlock with the **allowCircularPackages** argument. If the **allowCircularPackages** argument is set to **true**, the **loadOrder=verbatim** option is not valid and the default **loadOrder** argument value is used. This is because loading circular packages often requires reordering and retrying .**scm** files before any .**ddb** or .**ddx** files are loaded.

As part of this change, the **Load Order Verbatim** check box has been added to the Advanced Load Options dialog of the Jade Schema Load Utility.

Suppressing Reorganization prior to Loading Changed DDB and DDX Files

In earlier Jade releases, if you ran the **jadload** or **jadloadb** executable with the **suppressReorg** argument set to **true**, exception 8525 (*Reorganisation is required prior to DDB load*) was raised if a reorganization of user controls was required before the form and data definition (.**ddb** or .**ddx**) files could be loaded and then reorganized.

The way in which user controls are loaded has changed in Jade 2025 to reduce the circumstances in which this exception can be raised.

A reorganization is not required before form definitions are loaded if user control definitions have changed. (A reorganization *is* required if the form definitions include new user control definitions.)

The **jadloadb** batch Schema Load utility and non-GUI **JadeSchemaLoader** application now provide the **avoidReorgForFormsLoad** argument, which is set to **true** by default. Set this argument to **false** if you want to use the behavior of earlier Jade releases.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Alternatively, you can now uncheck the **Avoid reorg when loading form definitions** check box (which is checked by default) on the Jade development environment Advanced Load Options dialog if you want the behavior of earlier Jade releases; that is, the schema must be reorganized before the forms definition .**ddb** or .**ddx** files are loaded and then reorganized.

Local Exception Handler Disarming (PAR 69667)

Local exception handlers stay armed only until the method in which the handler was armed terminates, or until the handler is explicitly disarmed.

Caution Care should be taken when disarming a local exception handler. The search for a handler for exceptions of the specified class is not limited to the current method. If a suitable exception handler is not found in the current method, the exception handlers in calling methods are also checked. The **getExceptionHandlerStack** method of the **Process** class can be used to determine what exception handlers are currently armed.

Logging Direct Web Services HTTP Messages from the Jade Monitor

In the **Users** view of the Jade Monitor, you can now enable or disable logging for Direct web services HTTP messages. Right-click on a process and then select:

- **Turn HTTP Logging Off**, to turn off HTTP message logging.
- **Turn HTTP Logging On**, to turn on HTTP message logging. By default, HTTP messages are not logged.

Note Logging should be used only in development environments to help with debugging.

Logical Certifier Errors 44 and 45 (PAR 69602)

In Jade 2020, a check was added to the Meta Certifier to detect orphan event methods that had not been deleted when the owning control or menu was deleted. The fix that was generated was not applied correctly and this resulted in the invocation property of the method being set to an invalid value. This has no impact on the behavior of the application but has been fixed to address the inconsistency in the method meta data.

A new Logical Certifier error has been implemented to identify and repair event methods that have had the meta data updated incorrectly. The new error is number 45, and when the fix is applied, the meta data will be restored correctly.

Orb Web Framework

The *Orb Web Framework Reference* is now provided from this release.

Orb is an HTML-over-the-wire web framework that is native to the Jade Platform. It enables you to create rich web-based applications and provides the features of a modern web framework such as session handling and templating.

Orb is an optional extension to the Jade Platform. To get the Orb components, download the zip file from the Orb Web Framework tile on the **Extensions** page of the Jade Platform Developer Centre; that is:

https://www.jadeplatform.com/developer-centre/extensions

Presentation Client Single Sign-On (JAD-I-711)

The **JadeSoftware.Identity.Client.Desktop** component is a .NET assembly that can be imported into a Jade application to provide Single Sign-On (SSO) authentication. The component makes use of the Microsoft Authentication Library (MSAL) to provide the required functionality. The MSAL makes use of various .NET constructs such as asynchronous method calls that cannot be called directly in Jade. The **JadeSoftware.Identity.Client.Desktop** library is a wrapper with Jade-callable methods that can readily integrate into the standard Jade Platform security extension points.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Download the **JadeSoftware.Identity.Client.Desktop** component from the Jade Platform developer centre **Extensions** page at the following URL.

- https://www.jadeplatform.com/developer-centre/extensions

The **JadeSoftware.Identity.Client.Desktop** component is compatible with Jade releases 2020 and higher.

Note Jade supports Microsoft Entra ID (previously known as Azure Active Directory) SSO authentication only.

Additional information about installing and using the **JadeSoftware.Identity.Client.Desktop** component and the .NET API methods is available in the **README.MD** file that is included in the downloaded .**zip** file.

Note Ensure that you copy the folder and files in the **Library** directory of the downloaded .**zip** file into the **bin** directory of your Jade installation.

Relational Population Service (RPS)

This section describes the Relational Population Service (RPS) changes in this release.

Loading Bulk Data into an Azure SQL Database (JAD-I-525)

Relational Population Service (RPS) support has been extended to enable loading bulk data into an Azure SQL database using the new **rpsExtractDataAzure** method in the **JadeDatabaseAdmin** class. This method starts the RPS **Datapump** application on the server node to extract data for all tables in a format appropriate for loading into an Azure SQL database from the client side using a **bcp** script, and it returns the process of the asynchronous process that is generating the data extract scripts. For details about the method parameters, see "rpsExtractDataAzure" in Volume 1 of the *Encyclopaedia of Classes*.

To extract and load data, run the **rpsExtractDataAzure** method on the secondary database as a GUI or non-GUI application. For details, see "Extracting Data and Loading it into an Azure SQL Database", in Chapter 2 of the *Synchronized Database Service (SDS) Administration Guide*.

See also, "Configuring a Secondary RPS Database for Azure", elsewhere in this document.

RPS Enhancements

Historically, creating and setting up an RPS node required the use of the RPS Manager, which required a GUI interface.

You can now perform a number of RPS functions without requiring the use of the RPS Manager. These include:

- Multithread the creation of the RPS database, using the **JadeDatabaseAdmin** class **createRpsDatabaseMultithreaded** method
- Perform a number of RPS Manager functions using the command line, including the creation of a new RPS database, multithreading RPS database creation, and converting a primary database backup into an RPS node
- Converting a backup copy of the primary database into an RPS node without bringing up the RPS Manager

Primitive Type Methods

The **isOneOf** method defined in primitive types other then the **Any** and **Point** types has now been published. This method returns **true** if the receiver matches exactly any of the possible values provided in the **l** parameter list or it returns **false** if no match is found. You must specify two or more comma-separated alternatives against which to match.

In addition, the **isOneOfExtended** method defined in the **Character**, **String**, and **StringUtf8** primitive types is also now published. This method is an extended form of the **isOneOf** method, and it is available only for character-based primitive types. It allows more flexible case- and locale-aware matching for this subset of types.

Copyright 2025, Jade Software Corporation Limited All rights reserved

These methods (which are useful for unit testing, for example) take a variable number of arguments of the same primitive type as the receiver. The number of arguments is limited to the Jade Platform limit for a method of 129, but at least two possible matches must be provided.

REST Services

This section describes the Jade REST services changes in this release.

Controlling Quotes in REST String Responses

Currently any REST service method that returns a string type (that is, a **String**, **Character**, or **StringUtf8** primitive type) is enclosed in double quote (**""**) characters. Any existing quotes in the string response are escaped, as are other special characters.

A new boolean property called **encodeResponseString** has been added to the **JadeRestService** class. It defaults on creation to the previous hard-coded behavior of **true**. If the **encodeResponseString** property is set to **false**, the returned string will be returned unmodified in the REST response.

This property provides you with flexibility, to prevent any string encoding. When set within individual REST service methods, it enables REST servers to support both behaviors side by side.

Direct REST HTTP Access

A Jade REST application is a Jade application with the **Rest Services** or **Rest Services, Non Gui** application type. These applications expose REST endpoints by adding methods to a user implemented subclass of the **JadeRestService** class.

Before the implementation of the Direct REST feature, access to REST applications was available using only Internet Information Server (IIS). Client applications would make HTTP/1.1 REST requests to IIS, and then IIS would communicate with the Jade REST application to serve responses to these requests. The communication between the IIS and the Jade REST application was performed by the Jade Internet Server Application Programming Interface (ISAPI) extension Dynamic Link Library (DLL) called **jadehttp.dll**. This DLL implemented a proprietary Jade protocol between IIS and the REST applications.

The Direct REST feature adds support for the HTTP/1.1 communication protocol to REST applications. This means that Jade REST applications no longer require IIS to enable communication with other HTTP-compliant applications (for example, REST clients, web browsers, reverse proxies, web servers, or TLS termination servers).

This allows a greater degree of freedom for the deployment of Jade REST applications and it also reduces the system requirements for the development of REST applications.

See also "Direct REST Reference Architecture White Paper", in the following subsection.

Direct REST Reference Architecture White Paper

The *Direct REST Reference Architecture White Paper* is now provided from release 2022.0.05 (Service Pack 4).

Direct REST allows Jade REST applications to utilize any technology or cloud service that uses HTTP to facilitate communication. This white paper provides suggestions for how Jade Direct REST services should be deployed in production environments. However, you can use any HTTP-capable technology to fulfill this role.

While an intermediate gateway is required for external web connections, this does not preclude internal connections being made directly to the Jade REST application. This gateway can be the existing IIS web server without the use of the ISAPI module, as the web server itself does not suffer the limitations inherent when utilizing the jadehttp module. It is expected that the gateway will also provide features such as TLS termination, which are essential in a modern web environment.

Copyright 2025, Jade Software Corporation Limited All rights reserved

JadeRestService Class httpStatusCode Property (PAR 69625)

If the **httpStatusCode** property of the **JadeRestService** class is set to any value in the **1*xx*** range (Informational), **204** (No Content), or **304** (Not Modified), no message body is sent with the response.

This is to comply with RFC 7230 section 3.3 (https://www.rfc-editor.org/rfc/rfc7230#section-3.3), which mandates:

```
All 1xx (Informational), 204 (No Content), and 304 (Not Modified) responses do
not include a message body.
```

Sending Binary Payloads in Direct REST Responses

You can now send binary-formatted data (for example, an image or audio file) as payloads in Direct REST responses.

To send binary-formatted payloads, set the return type of the REST method to be the **Binary** primitive type and set the value of the **EnableBinaryPayloads** parameter to **true** in the [WebOptions] section of the Jade initialization file.

Use the **JadeRestService** class **addResponseHeaderField** method to set the "**Content-Type**" header field of the response to identify the format of the binary data; for example, "**image/png**".

The **replyBinary** method of the **JadeRestService** class is called when a response with a binary payload has been generated. Although you can reimplement the method to allow the application to post-process the response, you must call the **JadeRestService** implementation to complete the processing.

Ingesting Binary Payloads from REST Requests

You can ingest requests with Binary-formatted payloads in Direct REST methods.

To enable this functionality, you must have a REST method that has a **Binary** primitive as one of the parameters, the value of the **EnableBinaryPayloads** parameter in the [WebOptions] section of the Jade initialization file set to **true**, and the REST request must contain a Binary payload.

Note The format of the request payload is determined by examining the value of the "**Content-Type**" header field. If any of the values **"image"**, **"audio"**, **"video"**, **"pdf"**, or **"octet-stream"** is present in the header field value, the payload will be considered to be in a binary format.

Setting the Web Service Connection Type Name

In earlier releases, all web applications that make use of the **MultiWorkerTcp** subsystem were hard-coded to use the "**JadeHttpToClient**" connection name. This was appropriate when all web applications sat in behind Internet Information Server (IIS), but this is not as useful for Direct REST and Direct SOAP web service applications that connect directly to incoming HTTPS traffic.

The [WebOptions] section of the Jade initialization file now provides the **ConnectionTypeName** parameter and the **JadeMultiWorkerTcpTransport** class now provides the **setConnectionTypeName** method. These set the connection type name specified in the initialization file **ConnectionTypeName** or method **connectionTypeName** parameter.

The connection type name is used to control the Secure Sockets Layer (SSL) configurations used by any web-based application. Configuration of this value therefore enables application-specific configuration of secure communications.

Support of the OAuth 2.0 Client Credentials Flow

The Jade Platform REST client now provides built-in support for OAuth 2.0 authorization protocols so that you do not need to write boiler-plate code and can instead focus on writing code to solve your business problems. When a Jade application uses server-to-server authentication where the client acts on its own behalf, the REST client support of the Client Credentials flow of OAuth 2.0 enables applications to access resources on their own behalf; not for individual users.

Copyright 2025, Jade Software Corporation Limited All rights reserved

This feature implements the OAuth2 Client Credentials flow for the proxy classes generated when importing an OpenAPI specification, using the security information included in the specification. For service-to-service interactions, the Client Credentials flow of OAuth 2.0 allows an application to use its own credentials (client ID and secret) to obtain an access token from the authorization server, which can then be used for authorized API calls.

Saving Call Stacks when Objects are Created and Deleted (PAR 69178)

In earlier releases, you could save call stacks when objects are locked, but not when they are created or deleted.

You can now save call stacks when objects are created or deleted. These call stacks can be accessed programmatically and are provided in various places in the Jade Platform development environment; for example, to determine where objects have been created in Jade, where an object that is the subject of a 1072 exception was deleted, or locating the source of leaking transient objects.

Call stack saving is enabled or disabled with a Jade initialization file parameter, on a per-process basis using the Jade Monitor, or from **Process** class methods.

In this implementation, the object in the call stack can be accessed only by the process that created or deleted the object. In addition, the call stack information is not persisted; that is, call stack information for persistent objects is available only during the life time of the process in which they were created or deleted. This implementation records create and delete operations for shared object references; not subobjects, exclusive collections, blobs and slobs, or dynamic property clusters.

For details, see the following subsections.

Accessing Call Stack Information in the Development Environment

The Jade Platform development environment provides a number of ways in which you can access call stack information.

Note The new **DefaultProcessSaveCreateCallStack** and **DefaultProcessSaveDeleteCallStack** Jade initialization file parameters enable the automatic saving of object call stack information for the respective operation. For details, see "Enabling or Disabling the Saving of Call Stacks", in the following section.

In the:

- Call Stack Browser, if call stack information exists for the object referenced by the selected item, the **Call Stack** command is now displayed in the Inspect menu. The **Call Stack** command enables you to view the creation or deletion call stack in another Call Stack Browser.
- Local Variables window of the Debugger, if call stack information exists for a selected object, the **Call Stack** command is now displayed when you right-click on that object. The **Call Stack** command enables you to view the creation or deletion call stack details in the Call Stack Browser.
- Inspector form, if call stack information exists for a selected object, the Call Stack menu is displayed in the menu bar. The **Call Stack** command enables you to view the creation or deletion call stack details in the Call Stack Browser.
- **Users** view of the Jade Monitor, you can specify whether the create and delete call stacks of an object are captured by using the **Capture Object Call Stacks** command that is displayed in the popup menu when you right-click on a process.

  Note As part of this work, the popup menu command for enabling or disabling the saving of lock call stacks has been renamed **Capture Lock Call Stacks**.

- Default exception dialog for error 1072 (*Object has been deleted in the current transaction*), if deletion call stack information exists for the object in error, the call stack is displayed in the extended text of the exception dialog and in log files.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Enabling or Disabling the Saving of Call Stacks

The [JadeClient] and [JadeServer] sections of the Jade initialization file can now contain the following parameters.

- **DefaultProcessSaveCreateCallStack**, which is set to **false** by default.
- **DefaultProcessSaveDeleteCallStack**, which is set to **false** by default.

In the:

- [JadeClient] section, the parameters enable automatic saving of object call stack information for all processes on all client nodes for the create or delete operation. The parameters also apply to single-user nodes.
- [JadeServer] section, the parameters enable automatic saving of object call stack information for all applications on the database server for the create or delete operation.

Note Create and delete call stacks for system classes are not captured by default. To capture call stacks for system classes, add the class to the filter by using the **addCallStackFilter** method.

Jade creates and deletes objects of system classes internally. Enabling call stack capture for system classes results in call stacks for these internal operations being captured, and can have unexpected consequences such as a significant impact on memory use and performance.

Process Class Methods to Control and Retrieve Call Stacks

The **Process** class now provides the methods summarized in the following table to enable you to control and retrieve call stacks. For details, see Volume 2 of the *Encyclopaedia of Classes*.

Method	Description
addCallStackFilter	Restricts the saving of the call stack only to instances of the specified class or classes
allInstancesWithSavedCallStack	Populates the specified array with all objects that have a saved call stack for the specified operation
clearCallStackFilter	Clears the list of classes to which saving call stack information has been restricted for this process
clearCallStackInformation	Clears stored call stack information for the current process
getCallStack	Returns the call stack information for the object.
getCallStackAsString	Returns the call stack for the specified object as a human-readable string
getCallStackFilter	Returns the list of classes to which saving call stack information has been restricted for this process
getCallStackFilterActive	Returns **true** if any class filter exists for call stack saving on the receiving process for the specified operation
getSaveCallStack	Returns **true** if call stack capturing is currently enabled on the receiving process for the specified operation
hasCallStack	Returns **true** if call stack information exists for the specified object
setSaveCallStack	Enables or disables call stack saving on the receiving process of the specified operation

The existing lock call stack methods are now exposed through a new generic call stack interface, and the old methods are deprecated. For a list of lock call stack **Process** class methods that are deprecated in this release, see "Process Class Lock Call Stack Method Deprecations", earlier in this document.

Copyright 2025, Jade Software Corporation Limited All rights reserved

The new call stack methods take an **operation** parameter that specifies the call stack collection on which the method should operate. All of the methods will affect only the specified operation. The options are:

- process.CallStack_Create
- process.CallStack_Delete
- process.CallStack_Lock

The following **Process** class constants are now available to the **operation** parameter of the methods listed in the previous table.

Class Constant	Integer Value	Specifies the method is to work on the...
CallStack_Create	1	Create call stack
CallStack_Delete	2	Delete call stack
CallStack_Lock	4	Lock call stack

Security

This section describes the security changes in this release.

Securing a Web Session (PAR 69858)

The **WebSession** class now provides the **secureSession** method, which replaces the value of the **currentSession** system variable with a new **WebSession** object that it creates. After calling this method, the receiver of this method (which should be the existing **currentSession** object) is invalidated and is flagged to expire. (This method is intended to be used with the **JADE Forms** web application type.)

You must call this method *before* storing any sensitive information in the session referenced by **currentSession**, or before any action that might privilege or authenticate the **currentSession**. This is necessary to prevent "Session Fixation" vulnerabilities (https://owasp.org/www-community/attacks/Session_fixation). For example, you could have a routine to handle users logging in similar to the following code fragment.

```
if app.checkCredentials(username, password) then
    currentSession.secureSession();
    currentSession.username := username;
endif;
```

In this example, the **secureSession** method is called before storing the user name of the successfully logged in user in the **currentSession** object.

Important Notes about Secure Sessions

After calling the **secureSession** method, the old session is immediately invalidated.

The **WebSession** object associated with the session is deleted when the timer that cleans up expired sessions is next run, which happens within the number of minutes configured as the session timeout value in the **Session Timeout** text box on the **Web Options** sheet of the Define Application dialog. If your application has no session timeout configured (that is, it is set to zero (**0**)), the session, as with all sessions, remains active indefinitely. It is strongly recommended that applications that call the **secureSession** method have a session timeout configured.

Caution Until the object is deleted, the process license associated with the session remains in use.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Your application can handle deleting these objects, if required. If your application has no configured timeout, your application *must* remove sessions itself. A suggested way of doing this, shown in the following example, is to reimplement the **WebSession** class **processRequest** method to check if the active session was invalidated, and remove it if so. Invalidated sessions have their **lastAccessTime** and **startTime** properties set to null.

```
processRequest(httpString: String; queryString: String) updating;
begin
    inheritMethod(httpString, queryString);
epilog
    if app.isValidObject(self) and self.lastAccessTime = null and
    self.startTime = null then
        self.removeSessionWithMessage("Invalidated session");
    endif;
end;
```

Your application may need different handling, depending on how it makes use of its **WebSession** subclass.

Securing WebSockets to Jade Application Servers

You can now configure Jade to encrypt messages between the **JadeWebSockets_IIS.dll** and its application server to provide a simultaneous two-way communication channel over a single Transmission Control Protocol (TCP) connection.

The [Connections] section of the Jade SSL configuration file can now contain the **WebSocketsToClient** parameter, which controls the SSL settings for messages sent between IIS and Jade when using a **JadeWebSocketServer.IIS** set up.

Specifying the New Line Character for Regular Expression Searches

The **JadeRegexPattern** class now provides the **setNewlineConvention** method, which has the following signature.

```
setNewlineConvention(newlineConvention: Integer): JadeRegexPattern;
```

This method enables you to run regular expression (Regex) searches across data with different line-ending character sequences. The default is for the carriage return and line feed character pair to denote a line ending.

This method acts as an alternative to hard-coding **(*CR)**, **(*CRLF)**, and so on within your pattern strings.

Note In a Regex pattern, **\n** always matches the new line (line-feed) character and **\r** always matches the carriage return character. This cannot be changed by any options in the Perl Compatible Regular Expressions 2 (PCRE2) syntax. (For details about PCRE2, see https://www.pcre.org/current/doc/html/pcre2syntax.html.)

The constants listed in the following table are now available to the **newLineConvention** parameter in the **setNewlineConvention** method.

Class Constant	Value	Specifies...
Newline_Any	128	Any non-ASCII Unicode character as the line-ending character (in addition to those included in the **Newline_AnyCrLf** constant).
Newline_AnyCrLf	160	Any of the following as a line-ending character. ■ Carriage return ■ Carriage return followed by line feed ■ Line feed This means that the data can contain a mixture of line-ending character sequences.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Class Constant	Value	Specifies...
Newline_Cr	32	A single carriage return as the line-ending character.
Newline_CrLf	96	Carriage return followed by line feed as the line-ending character set.
Newline_Lf	64	A single line feed as the line-ending character.

Note The constants listed in the previous table are available to the **JadeRegexPattern** class.

SQL Server 2022 Support for RPS Nodes (PAR 69259)

From Jade 2018 and later, Microsoft SQL Server 2022 is supported for RPS nodes.

Microsoft SQL Server 2022 requires ODBC Driver 18 for SQL Server.

See also "Azure SQL Support for RPS Nodes (JAD-I-525, PAR 69475)", elsewhere in this document.

switch Instruction (JAD-I-96)

The Jade Platform now provides the **switch** instruction, which executes one block of statements from multiple alternative statements in a structured way, based on the value of an expression. It is typically used in place of multiple **if**/**elseif**/**else** statements in which the same expression is compared against several constant values.

For details, see "switch Instruction", in Chapter 1 of the *Developer's Reference*.

System Class Methods

The **System** class has new methods to manage a system sequence number and to return the number of event notifications for an object.

Managing a System Sequence Number

The **System** class now provides the following methods to manage a system sequence number.

- **deleteSystemSequenceNumber**, which has the following signature.

  ```
  deleteSystemSequenceNumber(name: String): Boolean;
  ```

  This method deletes the system sequence number with the name specified by the value of the **name** parameter.

- **getSystemSequenceNumberCurrent**, which has the following signature.

  ```
  getSystemSequenceNumberCurrent(name: String): Integer64;
  ```

  This method returns the current value of the system sequence number specified by the **name** parameter.

- **setSystemSequenceNumber**, which has the following signature.

  ```
  setSystemSequenceNumber(name: String; newValue: Integer64): Boolean;
  ```

  This method updates the current value of the system sequence number specified by the **name** parameter to the value specified by the **newValue** parameter.

Copyright 2025, Jade Software Corporation Limited All rights reserved

Returning the Number of Event Notifications

The **System** class now provides the following method to return the number of notification subscriptions for the object specified in the **target** parameter.

- **getNotesCountForObject**, which has the following signature.

```
getNotesCountForObject(target:      Object;
                       eventType:   Integer): Integer;
```

Unit Test Runner Form

The Unit Test Runner form now displays the **Cancel** button while tests are running. After the tests have completed, the **Run** button is redisplayed.

Click the **Cancel** button to stop further tests from running after the currently running test has completed.

Any relevant methods with the **unitTestAfter**, **unitTestAfterAll**, or **unitTestAfterAllClass** method option are still run.

You can now open the Unit Test Runner form by selecting the **Unit Test Runner** command from the File menu.

Visual C++ Redistributables Updated

For enhanced security and stability, we have updated the compiler toolset with which Jade is built. This requires that installations of Jade must be run with updated Visual C++ Redistributables. The minimum required version is distributed with Jade, in the **\bin** folder, or you can download the latest available from Microsoft.

The Jade installer will automatically install the redistributables when run, and a thin client download will also download and install the redistributables. You will need to install the updated redistributables on computers that host application or database servers yourself.

White Papers

The Jade 2025 product information library now contains the *Collection Concurrency* white paper in addition to a new Part 5 ("Transaction Agent Framework") in the *Erewhon Demonstration Reference*. For an overview, see the:

- *Collection Concurrency White Paper*, which covers the use of deferred operations to help minimize lock contention on persistent collections in multiuser systems.
- *Erewhon Demonstration System Reference*, "Part 5 - Transaction Agent Framework", which discusses persisting objects using the Jade Platform and it provides examples of a Transaction Agent Framework (TAF) that you can use in your own applications.

  Software requirements are constantly changing, so new methods and properties may need to be introduced at any stage of development. What the client initially requested is often different from their actual needs. Due to the high possibility of change, software should be written in a way that is open for extension and closed for modification. Simply put, we don't want to delete existing working code in order to develop additional requirements; instead we want to extend the code base with the new logic. This is referred to as the *Open/Closed Principle*.

  The Transaction Agent Framework does not use parameterized functions or constructors to create objects, because adding additional parameters when requirements change is likely to cause changes that break existing code. Instead, the TAF creates objects based on the properties stored in the transaction agent class. This method of creating objects not only eliminates parameterized functions and constructors but allows additional properties and logic to be added into the code base without affecting the original code.

Copyright 2025, Jade Software Corporation Limited All rights reserved