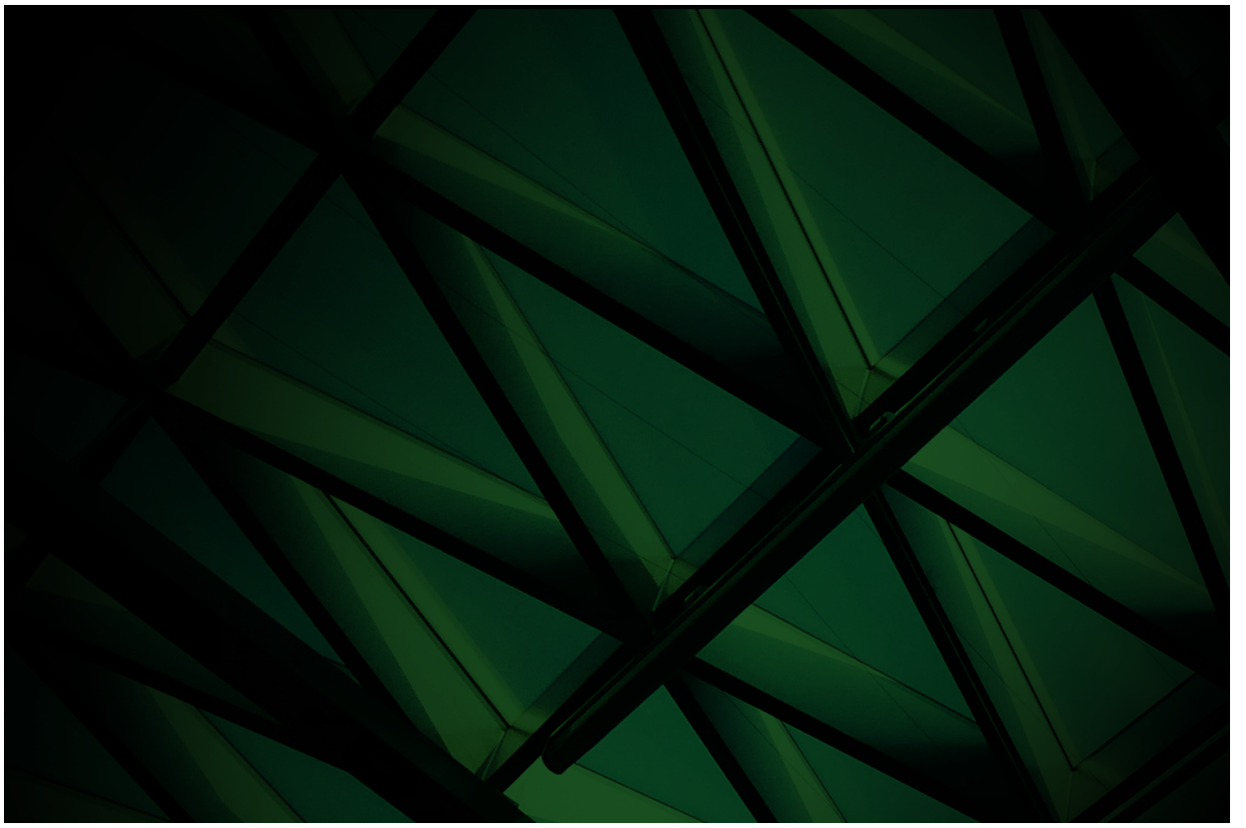


Upgrading to the Jade 2022 Release

VERSION 2022.0.04

jade platform

Contents

Upgrading to Jade Release 2022

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*.

Jade Release Support

For details about the:

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

Jade 2022 is built using Microsoft Visual Studio 2022, which requires the installation of the Microsoft Visual C++ 2022 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.

From release 2022.0.04, Jade supplies .NET 8 exposure binaries.

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.

RPS Working Set Data Store Mode Deprecated

The RPS **Working Set** data store mode has been deprecated, as have the related:

- **JadeDatabaseAdmin** class **RpsStorageMode_WorkingSet** constant
- **AutoExtractOnPrimary<*n*>** parameter in the [JadeRps] section of the Jade initialization file
- Exception 3420 (*RPS reorg extract on primary failed*)

Collection Class tryCopy__ Method Deprecated

The **Collection** class **tryCopy__** method, implemented in Jade 2020.0.02 (Service Pack 1), has been replaced by the **tryCopy** method.

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.

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 release 2022.0.03, and the new methods to use instead.

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

Highlights in this Release

The highlights in Jade release 2022, which help you to deliver high-performance, interoperable applications on Windows for both 32-bit and 64-bit platforms, are as follows. You can now use:

- Partitioning of database files that contain multiple classes, which enables more-efficient bulk delete operations with a reduction in infrastructure and human resource costs.

  For details, see "Database File Partitioning" under "Database Changes", later in this document.

- Collection enhancements, including:

  - Array membership of type **Any**
  - **Collection** class **tryCopy** and **tryCopyFrom** methods and the **tryAddIfNotNull** and **tryRemoveIfNotNull** conditional methods
  - **Collection** and **Dictionary** class methods to provide visibility to the effects of deferred collection operations

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

- REST enhancements, including:

  - Paginating responses when using the **JadeRestService** classes to provide web services
  - Use of more-standard REST request URLs
  - Support for determining the response format from the **Accept** header
  - Server-side support for **multipart/form-data** encoding
  - Support for ISO 8601-compliant **Date**, **Time**, **TimeStamp**, and **TimeStampOffset** formatting
  - Extended control of HTTP headers when using the Jade REST client

  For details, see "REST Services Enhancements", later in this document.

- DevOps test automation using the open-source **Mocking** framework.

  For details, see "Mocking Framework for Test Automation", later in this document.

- Support for Opportunistic Transport Layer Security (OTLS) from the **TcpIpConnection** class.

  For details, see "End-to-End SSL Encryption", in Chapter 2 of the *Installation and Configuration Guide*.

- Many new features and changes in the Jade Platform development environment, including:

  - Compiler Output Viewer, which displays the results of a compile including error, warning, and information messages
  - Customization of toolbar buttons
  - Debugger enhancements
  - Multiple caret and rectangular selection virtual space support in the **JadeTextEdit** and **JadeEditor** control classes
  - Unicode surrogate pair character support
  - Workspace refactoring

  For details, see "Jade Platform Development Environment", later in this document.

- Single Sign-On (SSO) authentication for presentation clients by importing the .NET assembly **JadeSoftware.Identity.Client.Desktop** component.

  For details, see "Presentation Client Single Sign-On (JAD-I-711)", later in this document.

Accessing Details about Faults Fixed in Releases

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

If you have any queries about **Parsys**, please direct them to Jade Parsys Support in the first instance, at parsyssupport@jadeworld.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 **2022** 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.

Upgrading to Jade 2022

This section covers the following topics.

For details about the Jade 2022 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 2022, refer to "Jade 2022 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 2022 from Jade 2020

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 2022 from Jade 2020, by performing the following actions.

1. On the Jade *2020* 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 2020 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.

3. If you defined your own **String** primitive type **endsWith__**, **replace__**, **replaceFrom__**, or **startsWith__** methods in Jade 2016.0.02 or later, you must rename them *before* you upgrade to Jade 2018 or higher, as Jade's new **endsWith__**, **replace__**, **replaceFrom__**, and **startsWith__**methods will conflict with any existing **String** method named **endsWith__**, **replace__**, **replaceFrom__**, or **startsWith__**.

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

5. Run the Jade 2022 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.

6. 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.

7. 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.

8. Use the Jade Database utility to take a full backup of your Jade 2022 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 2022, 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 2022. 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.

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.

Jade 2022 Changes that May Affect Your Existing Systems

This section describes only the changes in the Jade 2022 release that may affect your existing systems. Some changes may result in compile errors during the load process, or cause your Jade release 2022 systems to behave differently.

Converting a String to a Decimal (PAR 68630)

When a **String** is assigned to a **Decimal** variable, property, or parameter, the value being assigned is checked against the precision of the destination. An exception is raised if the value is too large for the destination. The value is rounded to the number of decimal places of the destination (the scale factor). If a **String** value was cast to a **Decimal** value and assigned to a variable of type **Any**, the precision check and rounding of decimal places was not done. This has been changed so that if a **String** value is cast to a **Decimal** and assigned to a value of type **Any**, exception 4042 (*Decimal conversion error*) or 4043 (*Result of expression overflows Decimal precision*), respectively, is now raised if the decimal string is invalid or if the **String** value is greater than the maximum **Decimal** value ($10^{\wedge 23}$). The decimal places are also rounded, to ensure that the **Decimal** value has fewer than 23 digits. (Note that this change could break existing applications that rely on the behavior of earlier releases.)

Exceptions Accessing System-only Features

Exception 1192 (*Feature is restricted to system processes*) is now raised when you attempt to execute **Object** class **sendMsg**, **sendMsgWithIOParams**, **sendTypeMsg**, **sendTypeMsgWithIOParams**, **sendTypeMsgWithParams**, **invokeIOMethod**, **invokeMethod**, and **setPropertyValue** methods to access system-only features, as these are restricted to internal system processes only.

Incorrect Evaluation Order of Method Calls in One Instruction (PAR 68983)

The order in which methods are called if a single instruction contains multiple Jade method calls and method calls as parameters has been changed.

Prior to Jade 2022, method calls used as parameters to a Jade method were evaluated *before* any method calls used for the receiver.

Note This change applies only if the method receiving the parameters is a Jade method. If the method is an external method, method calls used for the receiver are evaluated before any method calls used as parameters.

The new behavior removes the inconsistency between Jade methods and external methods.

Consider the following example.

```
method1().jadeMethod2(method3());
```

Prior to Jade 2022, the order of execution was **method3()**, **method1()**, **jadeMethod2()**.

In Jade 2022, the order of execution is **method1()**, **method3()**, **jadeMethod2()**.

If a variable was passed as a usage **io** or **output** parameter to one method and the variable was also dereferenced and passed as a parameter to another method, the wrong value was used. Consider the following example.

```
method1(variable).jadeMethod2(variable.property);
```

The value passed to **jadeMethod2()** used the value of **variable** before **method1()** is called. If the value was null, a 1090 exception was raised before any methods were called.

The [JadeInterpreter] section of the Jade initialization file can now contain the following parameter to revert the new evaluation order.

```
[JadeInterpreter]
LegacyParameterEvaluationOrder=true|false
```

Set this parameter to **true** if the parameters should be evaluated *before* the receiver; that is, the pre-Jade 2022 behavior is required. The default value is **false**. This parameter is read from the Jade initialization file when the node initializes.

The **_findNestedMethodCalls** method defined in the **Schema** class in the **RootSchema** has been implemented to search for methods that contain multiple method calls in a single instruction. Use this method to determine if changing the evaluation order will change application behavior.

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

```
_findNestedMethodCalls(fileName: String);
```

This method processes the code stream for all Jade methods, and candidate method calls are logged to the specified file. Review the file to determine if any changes are required to the application code.

The new order of evaluation could affect the application if the same variable is passed to multiple method calls, or if the method calls have side-effects and change global data. Rewrite the code using separate expressions to explicitly define the evaluation order of each method call, if required.

Jade Platform Development Environment Changed Values

The following changed default values could affect your existing systems.

- Preferences

  When you upgrade to Jade 2022, there are several preferences whose default value will change to show off useful functionality in the Jade Platform development environment. For details, see "Changes to Default Preference Values", under "Changes and New Features in Jade Release 2022.0.01", later in this document.

- Extract-related preferences and values.

  - The user preference for the default extract format of forms definition files is now the XML DDX format; that is, .**ddx**. The legacy DDB format is still supported.

    Notes If you exported your user preferences before upgrading to Jade 2022.0.01, your default settings are restored after you import your user preferences.

    DDX files extracted in Jade 2022.0.01 will not load into Jade 2020 or 2018 because they contain additional information and therefore are not backwards-compatible.

  - The default extract format is now .**ddx** where it is not controlled by the Preferences dialog.

  - The Extract dialog is now displayed instead of the common Save As dialog when you select the **Extract** command to extract an item.

  - UTF-8 with a Byte Order Mark (BOM) is now the default encoding format during all extract processes that you perform. If you want to continue to use native ANSI or Unicode as the default encoding format, uncheck the new **Extract as UTF-8 Encoded** check box on the **Source Management** sheet of the Preferences dialog.

  For details, see "Extract Format Changes", under "Changes and New Features in Jade Release 2022.0.01", later in this document.

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.

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.

The encryption used during the extraction of encrypted schema files has changed as part of the OpenSSL upgrade in this release.

- Encrypted schemas extracted after the OpenSSL upgrade will no longer load into systems that have not been upgraded; that is, encrypted schema files extracted with Jade 22.0.01.022 and higher will not load into earlier versions of 22.0.01 (for example, 22.0.01.021).

  Encrypted schema files extracted with Jade 22.0.01.021 or earlier, are still able to be loaded into all versions of 22.0.01.

- Encrypted schemas extracted before the OpenSSL upgrade will load into systems that have been upgraded.

- This update does not affect other normal schema load version requirements; that is, unencrypted schema files are unaffected.

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.

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.

SslConfigurationTool Executable (PAR 68118)

Because **localhost** is no longer a practical host name for clients to connect to, the **SslConfigurationTool.exe** utility has been changed to reflect this.

The **TryForDefaultConfiguration** and **SelectCertificateFromUI** actions have been enhanced to take the **ServerHostname** and **ServerPortNumber** command line arguments. If **ServerHostName** is not specified, by default, the SSL configuration tool uses the fully qualified domain name of the computer.

Changes in Jade Release 2022.0.04 (Service Pack 3)

This section contains details about product and documentation changes in Jade release 2022.0.04 (Service Pack 3).

For details about releases 2022.0.01 (the first general release of Jade 2022), 2022.0.02 (Service Pack 1), and 2022.0.03 (Service Pack 2), see "Jade 2022 Changes that May Affect Your Existing Systems, "Changes in Jade Release 2022.0.03 (Service Pack 2)", "Changes in Jade Release 2022.0.02 (Service Pack 1)", and "Changes and New Features in Jade Release 2022.0.01", elsewhere 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.

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
- 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.

Jade JSON Web Token Time Format Setting (PAR 69837)

The [WebOptions] section of the Jade initialization file can now contain the **JwtExpectUtc** parameter, which controls whether timestamps in Jade JSON Web Tokens (JWT) are processed as local time or as Coordinated Universal Time (UTC). This parameter is read each time a JWT timestamp is processed.

The parameter is set to **false** by default, which means that JWT timestamps are processed in local time.

Set the parameter to **true** to process JWT timestamps in UTC time.

Note From a future release, this parameter is deprecated and all Jade JWT timestamps are processed as UTC, which is the JSON Web Token standard (https://datatracker.ietf.org/doc/html/rfc7519).

The **JwtExpectUtc** parameter is available to Jade releases 2020.0.02 and 2022.0.03 and higher.

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**.

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.

Method	Description
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.

- **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 that have been added to the **JadeJson** class.

Method	Generates...
generateJsonDynamic	JSON based on the hierarchy of **JadeJsonObjects** and **JadeAnyArrays** of which the passed in **source** parameter is the root.
parseDynamic	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.

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 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.

REST Services Dynamic Parsing

REST services now provide dynamic parsing, which allows you to accurately ingest and reply with JSON data without a matching Jade class. You can navigate and modify this data in a Jade-like manner using the **JadeAnyArray** class and methods defined on the **JadeJsonObject** class.

The **JadeJsonObject** class stores an accurate representation of JSON data within a Jade system. It is transient-only and is valid only for the node on which it is created. The **JadeAnyArray** class implements the **JadeIteratorIF** interface and allows you to iterate through the property names and values of its matching object.

Instead of mapping the JSON to an existing Jade class, a hierarchy of **JadeJsonObject** and **JadeAnyArray** transient objects are created that accurately represents the provided JSON data. When the method has completed execution, these transient objects are deleted.

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 [JadeClient] section of the Jade initialization file can now contain the **DefaultIteratorsIncludeDeferredOperations** parameter, 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
	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

***** 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.

Delete Exposure Command File Argument (JAD-I-457)

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

- Delete Exposure *schema-name*::*exposure-name*

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 potential errors. The default value of the **Replayable** combo box is now **True**.

Event Streaming

The Event Stream Producer (ESP) is a component of the Jade Platform, enabling real-time data capture and transmission for downstream processing. It uses a journal-based Change Data Capture (CDC) mechanism to record state changes from the database and publish events to an event streaming platform such as Kafka or Azure Event Hubs.

The key features are listed in the following table.

Feature	Description
Journal-based CDC	Captures state changes from database journals, starting from the latest log sequence number (LSN) by default.
Event serialization	Converts captured state changes into JSON-serialized events following a well-defined JSON schema for consistent data representation.
Event publishing	Events are published to one or more topics in an event stream, using Kafka or Azure Event Hubs.
Configurable operation	The ESP is enabled via configuration settings and supports customization, including the ability to set or reset the starting journal offset.
At-least-once delivery	The ESP ensures "at least once" message delivery semantics, retransmitting messages in case of failure.
Globally unique event identifiers	Each event has a unique identifier in the format <*database-unique-ID*>-<*object-OID-and-edition*>, enabling consumers to detect and manage duplicate events.
Flexible deployment	The ESP can run on primary or secondary (SDS) databases, supporting a variety of deployment environments.

Feature	Description
Large data handling	Support for handling large binary (blob) or string (slob) data via external storage. External data is referenced in events using URIs.
Schema registry support	The producer is designed to support the Confluent Schema Registry protocol for managing and validating schemas, ensuring compatibility with Azure Event Hubs.

As Event Streaming is a new feature, we are interested in learning about the different use cases the community plans to address with it. To request access, click the **Contact us** button on the Event Streaming tile on the **Extensions** page of the Jade World Developer Centre; that is, https://www.jadeplatform.com/developer-centre/extensions. Complete the contact form. Jade Support will then send a questionnaire to the specified email address. After you return the completed questionnaire, Jade Support will send a ZIP file that contains the ESP components.

Integration and Extensibility

Integration and extensibility is provided by the components listed in the following table.

Component	Description
Kafka producer	A tested Kafka producer integrates with Azure Event Hubs, using the Event Hubs Kafka Interface client API
Schema evolution	Events are described using JSON schemas, to enable data consistency across system versions and to enable easy validation

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.

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.

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.

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.

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.

These methods (which are useful for unit testing, for example) take a variable number of arguments of the same primitive type as the receiver.

Note The number of arguments is limited to the Jade Platform limit of 129 for a method, but at least two possible matches must be provided.

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.

Changes in Jade Release 2022.0.03 (Service Pack 2)

This section contains details about product and documentation changes in Jade release 2022.0.03 (Service Pack 2).

For details about releases 2022.0.01 (the first general release of Jade 2022) and 2022.0.02 (Service Pack 1), see "Jade 2022 Changes that May Affect Your Existing Systems, "Changes in Jade Release 2022.0.02 (Service Pack 1)", and "Changes and New Features in Jade Release 2022.0.01", elsewhere in this document.

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.

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

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.

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 ExportedType *schema-name*::*exported-package-name*::*exported-type-name*
- Delete ExportedFeature *schema-name*::*exported-package-name*::*exported-type-name*::*exported-feature-name*

JadeJson Class modifyReadOnlyProperties Property (PAR 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 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.

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

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 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.

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.

Overriding the Patch Number when Loading Multiple Schemas or Files

When loading multiple schemas or files, you can override the incoming patch number so that the specified patch number is used.

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.

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

Method	Description
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.

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

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** or **HTML Documents** web application type. Attempting to call the **secureSession** method in a web service or REST application raises a 1068 *(Feature not available in this release*) exception.

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.

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.

White Papers

The Jade 2022.0.03 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.

Changes in Jade Release 2022.0.02 (Service Pack 1)

This section contains details about product and documentation changes in Jade release 2022.0.02 (Service Pack 1).

For details about release 2022.0.01 (the first general release of Jade 2022), see "Jade 2022 Changes that May Affect Your Existing Systems" and "Changes and New Features in Jade Release 2022.0.01", elsewhere in this document.

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)", elsewhere in this document.

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*

Disabling Enhanced Accessibility Features (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**.

Jade Platform Development Environment

This section describes the Jade Platform development environment changes in this release.

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 will not load into Jade 2022.0.01 or any earlier Jade releases.

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.

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

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.

expectTerminate Method

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

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*.

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
```

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.

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.

Changes and New Features in Jade Release 2022.0.01

This section summarizes the product and documentation changes and new features in Jade release 2022.0.01. For details about the changes in release 2022 that may affect your existing systems, see "Jade 2022 Changes that May Affect Your Existing Systems", earlier in this document.

Application Class setParamListTypeEntry Method (NFS 68690)

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

```
setParamListTypeEntry(index :     Integer;
                      value:      Any;
                      paramList: ParamListType io) final;
```

This method sets the value of the parameter in the **ParamListType** pseudo type list specified in the **paramList** parameter at the position specified in the **index** parameter to the value specified in the **value** parameter.

The first entry in the parameter list is at index **1**. If the value of the **index** parameter is outside the bounds of the parameter list, an exception is raised.

Checking for Orphan Subobjects in User Data Files

In the absence of a full logical certify of all instances of all user classes (for example, before upgrading to a higher Jade release), you can check for orphan subobjects in user data files by specifying the **orphanSubobjects** operation in the Jade Logical Certifier non-GUI application of the **jadclient** program; for example:

```
jadclient path=c:\jade\system ini=c:\jade\system\jade.ini schema=RootSchema
app=JadeLogicalCertifierNonGui server=SingleUser startAppParameters
operation=orphanSubobjects logDir=c:\jade\logs\certify
```

Collections

This section describes the changes to **Collection** classes and subclasses in this release. (See also "Reorganizing Changed Array Definitions (PAR 68306, PAR 68349)" under "Jade Platform Development Environment", elsewhere in this document.)

Array Membership of Type Any

The **Any** primitive type can now be selected for the membership type of a user-defined **Array** class. Values added to the array can contain an object reference or any primitive value.

The maximum length of **Binary**, **String**, and **StringUtf8** values is as follows.

- **Binary** - 16,000 bytes
- **String** - 15,999 characters
- **StringUtf8** - 8,000 UTF-8 characters

The RootSchema does not have a subclasses of **Array** with membership type of **Any**. If you require such an array, subclass the **Array** class in your user schema, selecting **Any** as the membership.

Collection Class Methods

The **Collection** class now provides the **tryCopy** method and the **tryAddIfNotNull** and **tryRemoveIfNotNull** conditional methods.

- The **tryCopy** method copies the values from the receiver collection to the specified target **toColl** collection that are not present in the target collection, and returns a reference to the target collection. This method is reimplemented in the **Collection** class **Dictionary**, **DynaDictionary**, and **Array** subclasses.
- The **tryCopyFrom** method copies the values that are not present in the receiver from the collection specified in the **sourceCollection** parameter to the receiver.
- The **tryAddIfNotNull** method attempts to add the value specified in the **value** parameter to the collection if the value is not null and it is not already contained in the collection. It returns **true** if the value was successfully added; otherwise it returns **false**.
- The **tryRemoveIfNotNull** method attempts to remove the value specified by the **value** parameter from the collection if it is not null and it is contained in the collection. It returns **true** if the value was successfully removed; otherwise it returns **false**.

Deferred Collection Methods

The **Collection** and **Dictionary** classes now provide the following methods to provide visibility to the effects of deferred collection operations. The:

- **includesWithDeferred** method of the **Collection** class returns **true** if the collection contains the object specified in the **value** parameter, taking account of deferred operations visible to the process.

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

  ```
  includesWithDeferred(value: MemberType): Boolean;
  ```

- **getAtKeyWithDeferred** method of the **Dictionary** class returns a reference to an object in the receiver collection at the specified **keys** parameter value, taking account of deferred operations visible to the calling process. If an entry with the specified **keys** parameter value is not found, the method returns a **null** value.

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

  ```
  getAtKeyWithDeferred(keys: KeyType): MemberType;
  ```

- **includesKeyWithDeferred** method of the **Dictionary** class returns **true** if the receiver contains an entry at the specified **keys** parameter value, taking account of deferred operations visible to the calling process. If an entry with the specified **keys** parameter value is not found, the method returns **false**.

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

  ```
  includesKeyWithDeferred(keys: KeyType): Boolean;
  ```

Iterating through Virtual Collections (PAR 67606)

The ability to iterate through virtual collections using the **Iterator** class **startAtObject** method has now been reinstated.

MergeIterator Class startKey Methods (PAR 68511)

If the dictionaries attached to a **MergeIterator** object do not have the same keys, the values specified in the **keys** parameter of the **MergeIterator** class **startKeyGeq**, **startKeyGtr**, **startKeyLeq**, and **startKeyLss** methods must be the same as the subset of common keys of the attached dictionaries.

Command Line Handling (PAR 68537)

Some issues related to command line handling have been addressed (for example, the **startAppParameters** argument) so that the **jade.exe** and **jadclient.exe** command line handling is now consistent.

Compound Assignments

The Jade language now supports compound assignment operators, which are an extension to the existing **:=**assignment operation. They provide a shorthand syntax for assigning the result of an arithmetic or concatenation operation.

The compound assignment operators listed in the following table are now supported.

Compound Assignment Operator	Assignment
+=	Addition
-=	Subtraction
*=	Multiplication
/=	Division
^=	Power
%=	Modulo
&=	Concatenation

Compound assignment operators perform the operation specified by the additional operator on the left and right operands, then assign the result to the left operand. For details and an example, see "Compound Assignments", in Chapter 1 of the *Developer's Reference*.

Context Menu showContextMenu__ Method

The **Window**, **ListBox**, **Table**, and **JadeEditor** classes now provide the **showContextMenu__** method, which has the following signature.

```
showContextMenu__(menu:   MenuItem;
                  conwin: Window input;
                  mouse:  Boolean;
                  x:      Integer;
                  y:      Integer);
```

The **showContextMenu__** method is a helper method that can be called from **contextMenu** events to display a menu using the event details. If the event was generated by the mouse, the x and y positions are converted to be relative to the form for the call to **Form** class **popupMenu** method. If the event was generated by the use of the keyboard, the required x and y positions will be generated by the method.

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

Parameter	Description
menu	The context menu to display. If this parameter is null, no menu is displayed.
conwin	The window being right-clicked or the control with the focus when the keyboard menu key is pressed. **Note** When the mouse is used, the **x** and **y** positions are relative to the start position of this window.

Parameter	Description
mouse	Contains **true** if the mouse generated the event or **false** if the keyboard generated it. If the **mouse** parameter is **true**, the context menu is positioned at the specified **x** and **y** positions relative to the window specified in the **conwin** parameter. If the **mouse** parameter is **false**, for: ■ A **ListBox** class, the context menu is positioned below the current **listIndex** entry, or at the top of the list if the **listIndex** property value is **-1** or the item is not visible. ■ A **Table** class, the context menu is positioned at the current table column and below the current row entry, or at the left if the column is not visible and at the top if the row is not visible. ■ A **JadeEditor** class, the context menu is positioned on the line below the cursor position if visible, or at **<conwin width/3, 5>** if the cursor position is not visible. ■ All other windows, the context menu is positioned at **<conwin width/3, 5>** relative to the **conwin** parameter.
x	The left position in pixels to display the context menu relative to the window specified in the **conwin** parameter when the **mouse** parameter is **true**, or ignored if the **mouse** parameter is **false**.
y	The top position in pixels to display the context menu relative to the window specified in the **conwin** parameter when the **mouse** parameter is **true**, or ignored if the **mouse** parameter is **false**.

The following example shows the use of the **showContextMenu__** method.

```
lstCustomers_contextMenu(listbox:ListBox input; conwin: Window input;
                         mouse: Boolean; x: Integer; y: Integer):Boolean
updating;
begin
    listbox.showContextMenu__(mnuEditMenu, conwin, mouse, x, y);
    return false;
end;
```

Containerization

This section describes the Jade containerization changes in this release.

■ Docker Images

2022 Docker images have been published with a **22.0.01** build version in the image tag; for example:

```
registry.jadeworld.io/jade/database-server:22.0.01-x64-U
```

We will also continue to publish **20.0.01** and **20.0.02** images containing the latest hotfixes as they are released.

■ Container Logging

Logging to **STDOUT** is now done using the JOM Logging console target instead of using the Microsoft LogMonitor tool. Console logging is achieved by specifying **JadeLog.UseLogServer=true** and **JadeLog.LogServer=Console** as **Entrypoint** command line arguments.

Note As advised in Jade 2020, the LogMonitor is no longer included in container images.

For details about containerization, see Chapter 3 of the *Installation and Configuration Guide*.

Converting a String Type to a Time Type (PAR 68119)

When converting a **String** primitive type to a **Time** primitive type, the valid time string is now as follows.

```
hh:mm[:ss.fff] [pre-or-post-noon-indicator]
```

This string is converted to **Time**. The rules that apply when casting a string to a time are as follows.

- The delimiter between time elements is a single non-alphanumeric characters.
- The hour element must be numeric in the range 0 through 24 (inclusive).
- The minute and second elements must be numeric in the range 0 through 59 (inclusive).
- The millisecond element must be numeric in the range 0 through 999 (inclusive).
- A term specifying pre- or post-noon (that is, case-insensitive **"am"**, **"a.m."**, **"pm"**, or **"p.m."**) is valid only if the hour element is less than or equal to 12.
- **"12:00 a.m.".Time** converts to **00:00:00**.
- **"12:00 p.m.".Time** converts to **12:00:00**.

Custom MenuItem Events (NFS 68101)

You can now add user-defined event methods for menu items, to handle populating or refreshing the state of each in a recursive manner in a similar way that you can controls. (See "Adding Methods to Your Subclassed Control", in Chapter 5 of the *Developer's Reference*".) In earlier releases, only the **click** and **select** events were available.

To add a custom menu item event method, add an external method to the **MenuItem** class as follows:

```
event-name(parameter-list) is CallMenuEvent in jadpmap updating;
```

After you have defined the custom menu item, clicking on a menu item property of a form will include that menu item name in the menu item event list that is displayed.

You can then define the event logic. To take effect, this event must be called manually in logic; for example:

```
mnuAddCustomer.event-name();
```

If the menu item is deleted, the associated event methods (including any custom events) are also deleted.

Database Changes

This section describes the Jade database changes in this release.

Database File Partitioning

The Jade database partitioning capability has been enhanced to allow files that contain multiple classes to be partitioned, with a goal to enable more-efficient bulk delete operations with a reduction in infrastructure and human resource costs.

The enhancements that allow for the partitioning of database files with multiple class maps includes the following changes.

- The database engine now supports a variant database partitioned file that can contain instances of more than one class.
- The compiler now supports the definition of a *partition ID* method, which returns an *absolute partition ID*. The return type of a partition ID method is **Integer64**.

- The Jade Platform development environment and compiler allow marking database files with multiple class maps as partitionable, as follows.
  - The ability to map more than one class to a database file that has **partitionable** enabled.
  - Allows setting the **partitionable** property on an existing database file that has more than one class mapped to it.
  - The **Partitionable** check box on the File Dialog is now enabled for an existing database file that has more than one class mapped to it. All other partitioned file restrictions are retained, including:
    - You cannot mark the default map as partitionable
    - A partitionable database file can be defined in one schema only in the hierarchy
  - The Define Class dialog now enables you to map more than one class to a database file that has the **DbFile** class **partitionable** property set to **true**.

    The top option button (the **common** option) in the Map Exclusive Instances To group box is disabled for **Collection** classes mapped to a partitionable database file, and the selection of the **Same file as collection owner** option button is enforced.
- The Jade Database Administration utility (**jdbadmin**) **MakePartitioned** action now supports:
  - Partitioning of a file containing instances of several classes.
  - Partitioning an encrypted database file.
  - Two new parameters have been added to the **action=MakePartitioned** command line argument: **partitionIdMethod** and **maxPartitionId**.
    - **partitionMethod** identifies the partition ID method to be invoked for the object processed.

      Note Only one of the **method** or **partitionIdMethod** parameters should be specified.

      A partition ID method returns the absolute partition ID of the partition in which the receiver should be stored. Subobjects (for example, collections, blob or slob properties, and Jade bytes) are automatically stored in the same partition as the parent object.
    - **maxPartitionId** specifies the maximum partition ID value that will be returned by calls to **partitionIdMethod** during execution of the **MakePartitioned** action. This limit serves two purposes: range checking and resource allocation.

      If a partition ID method call returns a value that exceeds the specified maximum partition ID value, an exception is raised causing the **MakePartitioned** action to fail.
- The **DbFile** class **setPartitioned** method now supports partitioning a file with multiple class maps (and no instances).
- The new **DbFile** class **setMaxPartitionId** method sets the maximum partition ID for the database file.
- The new **DbFile** class **getMaxPartitionId** method returns the maximum partition ID of the database file.
- The new **JadeDbFilePartition** class **purgeInstances** method.

  This server execution method executes a bulk-removal operation that performs the required side-effects of deleting each object in the partition without deleting them from the database, and then deletes the partition. Deleting a partition removes the physical file from the file system and changes the partition state to *deleted*. Because objects and subobjects stored in the partition are not deleted, file maintenance and auditing overheads are avoided.

The object deletion side-effects that are performed include:

- Inverse maintenance
- Destructor execution
- Deletion of child objects that can be reached through parent-child relationships

When a purge instances operation is executing, attempts to create, update, or delete objects in a partition that is being purged are rejected with exception 3186 (*Operation not valid: partition being purged*).

- The new **JadeDbFilePartition** class **remove** method.

  This server execution method removes the database partition associated with the receiver. Before a partition can be removed, it must be empty.

- Extends the .NET API to provide the following new methods.

  - The static **SetObjectPartitionID** method of the **DatabaseFile** class specifies the absolute partition in which to locate the specified object. It must be called within the creating transaction. Exception 3187 is raised if the specified object is not being created in the current transaction.

    ```
    JadeSoftware.Joob.MetaSchema.DatabaseFile.SetObjectPartitionID
    public static void SetObjectPartitionID(JoobObject obj, Int32
    partitionID)
    ```

    The value of the **partitionID** parameter must be a value in the range **1** through the lower of the maximum partition ID or (**Int32.MaxValue – 15**). See the **GetMaxPatitionID** and **SetMaxPartitionID** methods. Exception 3146 is raised if the specified partition ID or partition index is out of range.

    The target partition must be present and not frozen. The value that is set is observed only when the transaction commits.

  - The static **GetObjectPartitionID** method of the **DatabaseFile** class returns the identifier of the database partition in which the specified object is located.

    ```
    JadeSoftware.Joob.MetaSchema.DatabaseFile.GetObjectPartitionID
    public static Int32 GetObjectPartitionID(JoobObject obj)
    ```

  - The **SetMaxPartitionID** method of the **DatabaseFile** class sets the maximum partition ID for the database file.

    ```
    JadeSoftware.Joob.MetaSchema.DatabaseFile.SetMaxPartitionID
    public static Int32 SetMaxPartitionID(JoobObject obj, Int32 partitionID)
    ```

  - The **GetMaxPartitionID** method of the **DatabaseFile** class returns the maximum partition ID of the database file.

    ```
    JadeSoftware.Joob.MetaSchema.DatabaseFile.GetMaxPartitionID
    public Int64 GetMaxPartitionID(JoobObject obj)
    ```

  - The **PurgeInstances** method of the **DatabaseFilePartition** class executes a bulk-removal operation that performs the required side-effects of deleting each object in the partition, and then deletes the partition. The deletion step removes the file from the file system and changes its state to *deleted*.

    ```
    JadeSoftware.Joob.Management.DatabaseFilePartition.PurgeInstances
    public PurgeInstances()
    ```

    The object deletion side-effects that are affected include:

    — Inverse maintenance

  - Destructor execution
  - Deletion of component aggregates that can be reached through parent-child relationships

- Jade provides a process to convert a legacy (that is, pre-2022 Jade version) single-class partition file to a multi-class format, using the new **UpgradePartitionedFile** action in the Jade Database Administration utility (**jdbadmin**).

Bulk removal of objects can now be achieved using the existing Jade Database Administration utility **PurgePartition** operation or the **JadeDbFilePartition** class **purgeInstances** method.

Note The underlying purge instances database operation is already optimized in that individual object deletions are not journalled, removing one of the main operational pain points.

Database Initial and Extent File Sizes (PAR 68240)

From Jade 7.0 and higher, the database file initial and extent size minimum and default values increased. The documentation has now been updated to state that the minimum value is **64K** and the default value is **128K** for both the extent size and initial size file attributes.

JadeDbFilePartition Class drop Method (PAR 62426)

The following caveat has been documented for the **JadeDbFilePartition** class **drop** method, which removes objects in the global partition index, removes the partition, and marks it as deleted.

Caution The **drop** method does not execute destructors or trigger inverse maintenance, which means that inversed collections that referenced objects in the partition will contain invalid object references after the **drop** method has completed.

Deleting Subobject Dynamic Properties (PAR 61609)

The **JadeDynamicPropertyCluster** class now provides the **deleteSubobjectDynamicProperty** method, which deletes a subobject dynamic property definition and any subobject instances that may exist for the class on which the cluster is declared and any subclasses. This allows for deletion of an exclusive dynamic property when there are instances of the parent class.

Subobject instances include:

- Exclusive collection instances
- **Binary** and **StringUtf8** values for dynamic properties with a maximum length or a length greater than **540**
- **String** values for dynamic properties with a maximum length or a length greater than **539**

Note As this method can result in a large number of objects being deleted, consideration should be given to calling this method in its own transaction.

Subobject dynamic property definitions and instances can be deleted only if the class in which it is defined is not being used by any other process. If production mode is set, a subobject dynamic property can be deleted only in single user mode.

Displaying Time and TimeStamp Primitive Type Millisecond Values (JAD-I-648)

The **Time** and **TimeStamp Primitive** types now provide the **displayWithMilliSeconds__** method, which has the following signature.

```
displayWithMilliSeconds__(): String;
```


This method returns a string of the value that includes the milliseconds part of the time or timestamp.

You can use this method so that the Jade inspector and debugger include the milliseconds in the display of non-null time and timestamp values.

Note This method affects the **Object** class **display** method. The **Time** and **Timestamp** primitive type values on an object now include milliseconds, which could have an impact on your system when this method is called. For example, if you have a test that compares the output with a previous output, the match will fail. In this case, you should update your test.

Exception Handler Ex_Resume_Method_Epilog Return Value

The Jade exception handler can now have an **Ex_Resume_Method_Epilog** return value, which passes control back to the method that armed the exception handler. This extension to the existing Jade exception-handling model provides a way to implement try-catch-finally behavior.

Execution resumes at the start of the method epilog or at the end of the method if there is no epilog section. Execution resumes at the next statement in the epilog if the exception was raised while executing the epilog. Epilog sections are executed for any methods on the call stack between the method that armed the handler and the method where the exception is raised.

To use **Ex_Resume_Method_Epilog** as the return value, the exception must be resumable; otherwise, another exception 1238 *(Exception handler invalid return code)* is raised. For **SystemExceptions**, **resumable** is **true** by default, and **false** for **FatalErrors**. For exceptions that you raise yourself, you should set the value of **resumable** to meet your application requirements.

Ex_Resume_Method_Epilog is generally useful only for local exception handlers. If you were to use this return value with a global exception handler, the method that armed the exception handler may no longer be executing. If **Ex_Resume_Method_Epilog** is returned by a global exception handler, the behavior is as if the return value is **Ex_Abort_Action**; that is, the execution stack is cut back, and the application reverts to an idle state.

Extended create Instruction Expansion

In this release, the Jade language syntax has been expanded to make the extended **create** instruction compatible with more usages.

In addition to allowing the extended **create** instruction to be used as the right-hand side expression of an assignment statement, it can now be used in the following situations.

- As a parameter of a method call, provided that the formal parameter is **constant** or **input**.
- As the expression of a **return** instruction.
- As a parameter to a superclass parameterized constructor within a **create** method.

Exception 6801 (*Cannot assign to create expression*) is raised if you attempt to assign an extended **create** expression as the value of an **io** or **output** parameter.

See "**create** Instruction" in Chapter 1 of the *Developer's Reference* for more details and examples.

Extracting a Class using the jadclient Non-GUI Client Application

The **JadeBatchExtract** application in the **jadclient** non-GUI client application now enables you to automate the extraction of a specified class from the current or latest schema context.

Extracting Binaries for RPS (PAR 68724)

The performance of extracting **Binary** primitive types for an RPS mapping has been significantly improved.

File Open Dialog Prompt (PAR 65662, PAR 66871)

When using the common File Open dialog (the **CMDFileOpen** class), selecting a file, then navigating to another directory, you are no longer prompted to create the file selected, which prevents the display of unnecessary Create File dialogs. You are now advised that file was not found, advising you to check the file name and try again.

The default value of the **CMDFileOpen** class **createPrompt** property has been changed to **false**. When a user now attempts to open a file that does not exist, the create prompt is not displayed and the user is no longer prompted to create the file each time a folder is navigated to from within the File Open dialog.

Note To achieve the behavior in earlier releases, you must now specify **cmdFileOpen.createPrompt := true;** before you call the **CMDFileOpen** class **open** method.

GroupBox Class backColor (PAR 68610)

When a **GroupBox** control is added to a form in the Jade Painter, the Painter changes the **backColor** property to **Parent's Colour** instead of retaining the **3D Face** setting set by the constructor.

This conflicts with the way skin drawing is handled, because the skin logic considers **3D Face** as the default when deciding whether to draw the background of the control using the skin or the **backColor** property value of the control.

Creating a dynamic group box control at run time defaults its value from **backColor** to **3D Face**.

To enable all group boxes added in future in the Jade Painter to default to **3D Face**, implement the method **setDefaultPainterControlProperties** on the **GroupBox** class to set the **backColor** property to **3D Face** as follows.

```
setDefaultPainterControlProperties() updating;
begin
    inheritMethod();
self.backColor := Window.SystemColor + Window.Color_3D Face;
end;
```

The above logic will be called by the Jade Painter when a new **GroupBox** control is added to a form.

Future releases of Jade will retain the **3D Face** property default value for new **GroupBox** controls in Jade Painter.

Note Existing controls are not affected. You could have to manually change **GroupBox** controls back to **3D Face** if skin drawing is affected by PAR 67350.

Ignoring the Application Skin on a Form and its Child Controls

The new **Form** class **ignoreSkinAllControls** property specifies whether a form and all of its child controls use a skin. By default, this property is set to **false** and the form and all of its child controls (including dynamic controls) use any appropriate skin.

When you set this property to **true**, the application skin is ignored on the form; that is, you do not have to set the value of the **Window** class **ignoreSkin** property to **true** for each control on the form.

Note If the control has been explicitly set using the **Form** or **Control** class **setSkin** method, the skin is not ignored.

Jade Platform Development Environment

This section describes the Jade Platform development environment changes in this release.

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.

Changes to Default Preference Values

When you upgrade to Jade 2022, there are several preferences whose default value will change to show off useful functionality in the Jade Platform development environment.

The changed default preference values, listed in the following table, are automatically applied to all new users. For existing users, a subset of the new default preference values is automatically applied. In the following table:

- **Y** indicates that the new value is used.
- **N** indicates that the old value is used.

Note If you load (import) user preference values from an existing user preferences extract initialization file, the values in that file are applied.

Preference	Old Value	New Value	New Users	Existing Users
Editor Options sheet				
Folding Options	None	Normal	Y	Y
View Line Numbers	Unchecked	Checked	Y	Y
View Status Line Info	Unchecked	Checked	Y	Y
Window sheet				
Save Size and Position	Unchecked	Checked	Y	N
Show Backdrop	Unchecked	Checked	Y	Y
Show Alternating Row BackColor	Unchecked	Checked	Y	Y
Browser sheet				
Show Navigation Bar	Unchecked	Checked	Y	N
Mdi	Use Mdi	Use Mdi With Tabs	Y	N
Exit sheet				
Save Breakpoints	Unchecked	Checked	Y	Y
Save Windows	Unchecked	Checked	Y	N
Source Management sheet				
Unset Extract option 'Create JCF Command File'	Checked	Unchecked	Y	Y
Extract as DDX (xml format) instead of DDB	Unchecked	Checked	Y	Y
Painter sheet				
Properties On Top	Unchecked	Checked	Y	Y

For details, about the:

- **Editor Options**, **Window**, **Browser**, **Exit**, and **Source Management** sheet preferences, see "Setting User Preferences", in Chapter 2 of the *Development Environment User's Guide*.
- **Painter** sheet preferences, see "Maintaining Painter Options", in Chapter 2 of the *Installation and Configuration Guide*.

Custom Toolbar Buttons (NFS 68720, JAD-I-162)

You can now add one or more custom buttons to the Jade Platform development environment toolbar to run a specific application, **JadeScript** method, or Jade test cases in standard run or debug mode, regardless of the current schema. You can click these buttons to initiate the required action without having to switch the Jade Platform development environment view to the required environment.

Note Custom toolbar buttons apply to *your* view of the Jade Platform development environment; not that of other users.

A new button with the caption **+** is positioned to the right of the clipboard toolbar. This button initiates the dialog from which you define the functionality to be assigned to a new button that will be added to the toolbar area. You must select a schema and application from the relevant combo boxes on the dialog.

The dialog provides the following option buttons that determine the type of action.

- **Run Application**, which means that clicking the new button runs the selected application using the selected schema. This action is the same as selecting the required schema in the Jade Platform development environment, initiating the Run Application dialog, and then selecting the required application.
- **Run Script**, which adds all **JadeScript** classes for the selected schema to the **JadeScript Class** combo box. Check the **Show Super Schema Classes** check box to include all superschema **JadeScript** classes in the combo box. Only classes that have suitable **JadeScript** methods are included (that is, no type methods and only methods that have no parameters).
- **Run Test Case**, which adds all **JadeTestCase** classes for the selected schema to the **Test Case Class** combo box. Check the **Show Super Schema Classes** check box to include all superschema **JadeTestCase** classes in the combo box. Only classes that have suitable **JadeTestCase** methods are included (only test case methods).

When you click a custom button that you have added, a new copy of the selected application is started and it runs under the selected schema.

For details, see "Using Custom Toolbar Buttons" and "Adding a Toolbar Button to Run a Script or Application", in Chapter 2 of the *Development Environment User's Guide*.

Debugger

This section describes the Jade Debugger changes in this release.

Breakpoint Conditions (NFS 68684)

The **Condition** text box on the Breakpoint Options dialog has now changed to a **JadeEditor** control class so that the condition text is now colored as it is for a Jade method.

If AutoComplete functionality is enabled, you are provided with the usual prompts when the text is entered. The local parameters, local variables, and local constants for the method in which the breakpoint is defined are pre-loaded in AutoComplete for the entry of the condition. The condition logic expects a **Boolean** return value.

The breakpoint condition can now be expressed in two ways, as follows.

1. A Boolean expression; for example, **indx > 2** (this is unchanged from earlier releases).
2. Using one or more **if** instructions and specifying a return value of **true** or **false**; for example:

```
if indx = 13 then
    return true;
elseif indx = 1 and cust.number = 345 then
    return true;
else
    return false;
endif;
```

This condition must begin with the conditional **if** instruction. The final semicolon (**;**) of an expression is optional.

Jade constructs a transient method that is essentially as follows.

```
m1(): Boolean;
vars
    local-variable-and-parameters
begin
    load-locals-entities-with-values-from-the-actual-method-being-debugged
    return expression;  // for case 1 in the above list
```



```
    expression;          // for case 2 in the above list
end;
```

Tip The Breakpoint Options dialog can now be resized.

Debugger Breakpoints (NFS 68667, JAD-I-315)

Conditional breakpoints in earlier releases were set only in the Jade Debugger.

Jade now provides additional functionality for setting breakpoints in the Jade Platform development environment, as follows.

- If the line number display or folding is enabled in the editor pane for a Jade method, an extra column is added to the display in the left margin after the line number and before the folding margin.
- When an enabled breakpoint is set on a line, a red circle is displayed in the breakpoint margin.
- When a disabled breakpoint is set on a line, a white circle is displayed in the breakpoint margin.
- Clicking on an empty breakpoint margin for a line sets a breakpoint.
- Clicking on a disabled breakpoint (indicated by a white circle) enables the breakpoint.
- Clicking on an enabled breakpoint (indicated by a red circle) removes the breakpoint.
- Right-clicking in the breakpoint margin displays a breakpoint menu that has the following commands.
  - **Clear All Breakpoints From Method**.
  - **Breakpoint Options**, which initiates the Breakpoint Options dialog that allows you to disable or enable a breakpoint, set a pass count, and set a conditional expression for the breakpoint.
  - **Disable** or **Enable**, which toggles the disabled or enabled status of the current breakpoint.
  - **Remove Breakpoint**.

  The **Disable Breakpoint** or **Enable Breakpoint** and the **Breakpoint Options** commands are disabled if the method has been changed and not compiled or if there is no breakpoint on that line.
- Hovering the mouse over a breakpoint indicator displays a bubble help window with the breakpoint details; for example, **Disabled breakpoint**.
- The Jade Platform development environment editor pane right-click breakpoints menu now also includes the ability to invoke the Breakpoint Options dialog and to clear all breakpoints.
- The Breakpoint Options dialog provides a large **Condition** text box that allows for multiple-line input. The dialog can also now be resized.

  The **Condition** text box also now provides AutoComplete functionality.

Note All of the above functionality is also available in the Jade Debugger method source window.

In addition, when the debugger stops on a breakpoint and the Jade Debugger Breakpoints window is displayed, the breakpoint is selected in the Debugger Breakpoints window.

Any selection in the Jade Debugger Breakpoints window is cleared if the halt was for any reason; for example, a step into or step over action.

A breakpoint always shows an icon in the breakpoint column of the user method in the development environment editor pane and the Jade Debugger method source window.

The Preferences dialog **Editor Options** sheet now provides the **Highlight entire source line for breakpoints** check box that controls whether the display of a breakpoint highlights the whole source line for a breakpoint. (This check box is checked by default.)

Inspecting Code in the Debugger (PAR 68357)

A number of changes and enhancements have been made when the debugger is at a breakpoint and when inspecting code in the debugger.

- When the debugger is at a breakpoint, if nothing is selected and you hover the mouse over a variable, the value of the variable is displayed when possible (including path expressions). No methods are executed.

  If an expression is selected, hovering over the selection attempts to execute the expression. The expression must be compilable within the context of the method and must return a value.

  If the compile and execution succeed, the returned value is displayed.

  Note It is your responsibility for the effects that the expression may have on the database and execution of the method logic.

- The Inspector dialog accepts expressions that can be compiled and executed. There are no restrictions other than the expression able to be compiled and return a value.

  Expressions longer than 100 characters are accepted, and it is your responsibility for the effects that the expression may have on the database and execution of the method logic.

  Note You can now resize the debugger Inspect Variable dialog, which enables you to enter longer expressions.

- The inspector value dialogs for both the primitive value display and modification can now also be resized to better display longer and multiple line values.

- The Local Variables window now includes **self** unless the method being executed is a type method.

- The debugger now accepts expressions as watches. Watches, however, are limited to 100 characters (stored in an identifier array where the key is a maximum of 100 characters).

  The debugger accepts only the initial entry of a new watch expression if it successfully compiles, executes, and returns a value. To enter a new watch, therefore, the debugger must be at a break position where the expression is valid and the objects involved are not null.

  When the watch is active, if it is not valid in another context, the value is displayed as **[Out of scope]**. If the expression cannot be executed because a variable is null, ****not Initialised**** is displayed.

  Note It is your responsibility for the effects that the expression may have on the database and execution of the method logic.

- Watches in the debugger are again restored if the watch window was restored when the debugger starts up.

Delta Searches (NFS 68089)

When you enter text in the **Search for Delta Id Containing** text box on the Delta Browser, pressing Enter now performs the search so that you no longer have to tab to the **Search** button or use the mouse to click it.

If a matching delta is found, the delta is selected. If the delta is *not* found, a bell sound will result.

Display of the Class in which a Method is Defined (NFS 68122)

When the **Show Inherited** command is selected in the View menu, it can be difficult to determine the class in which a method is defined when a class has a large number of superclasses and methods. The Class List of the Class Browser or Interface Browser now provides additional information about a displayed method and for methods in the Methods List of a browser.

For the methods displayed in the:

- Editor pane, when the mouse is hovered over the method name at the start of the source, the Jade AutoComplete feature now displays bubble help for the method, which includes the schema and class to which the method belongs.
- Methods List in the Jade Hierarchy Browser and Methods Browser, when the mouse is hovered over an entry, bubble help now includes the schema and class of the method, as well as the method signature.

Displaying Multiple Sheets in the Editor Pane (JAD-I-106)

Multiple-sheet functionality is now available in the editor pane of the hierarchy browser forms in the Jade Platform development environment.

Multiple sheets in the editor pane enable you to navigate away from a method that you are currently editing (to view property or class details, for example) without losing your position in the source or being prompted to save your changes every time.

Note You are prompted to save your changes if clicking on a control or menu item property can result in the Methods List window reloading (to show control event methods, for example).

When you upgrade to Jade 2022.0.01, the multiple-sheet functionality is turned on, by default. If you do not want to display multiple sheets in the editor pane, use the Preferences dialog to turn off (or hide) the multiple sheets and restore the editor pane to a single pane. The Preferences dialog **Browser** sheet provides the new Class Browser Options group box that controls display options for multiple-sheet functionality.

The browser forms that have multiple-sheet functionality available include the following.

- Class Browser
- Interface Browser
- Primitive Types Browser
- Class References Browser
- Exposure Browser
- Export Packages Browser

For examples of the multiple-sheet functionality, see "Displaying Multiple Sheets in the Editor Pane", in Chapter 3 of the *Development Environment User's Guide*.

Depending on the browser form type, the following tabs are displayed in the editor pane when multiple-sheet functionality is enabled.

- Class Details, Interface Details, Type Details, or Collection Details

  Displays details about the class selected in the Class List window, interface selected in the Interface List window, primitive type selected in the Primitive Types List window, respectively.

- Property Details or Constant Details

  Displays details about the property selected in the Properties List window or the constant selected in the Constants List window, respectively.

- Source

  Displays details about the method selected in the Methods List window.

When you have multiple-sheet functionality enabled, right-clicking in the multiple-sheet tab area displays a menu that has the following commands.

- **Align Tabs Top**, which displays the tabs at the top of the editor pane. In addition, the Class Browser Options group box options are updated on the **Browser** sheet of the Preferences dialog.
- **Align Tabs Bottom**, which displays the tabs at the bottom of the editor pane. In addition, the Class Browser Options group box options are updated on the **Browser** sheet of the Preferences dialog.
- **Hide Tabs**, which turns off the multiple-sheet functionality; that is, the tabs are no longer displayed. In addition, the Class Browser Options group box options are updated on the **Browser** sheet of the Preferences dialog.
- **Combine Panes** or **Separate Panes**, which enable you to toggle between combined and separate sheets. When the sheets are combined, the **Source** sheet is separate from the two details sheet.

Note You can also double-click on the tabs to toggle between combined and separate sheets.

Displaying Subschema Copies of a Class (JAD-I-662)

A class defined in a superschema might no longer be used but cannot be removed because it has methods or constants declared in one or more subschemas.

Use the new Copy classes of *schema-name::class-name* browser to identify the subschema copies of a class.

To display the subschema copies of a class

- Perform one of the following actions.
  - Select the **Show Subschema Copies** command from the Classes menu in the Class Browser for your schema.
  - Right-click on the class and select the **Show Subschema Copies** command from the popup menu that is then displayed.

Note The **Show Subschema Classes** command is enabled only if the selected class has subschema copies.

The Copy classes of *schema-name::class-name* browser is then displayed.

The subschema copies of the current class are displayed in the upper area of the browser. Select a subschema copy to view more details in the lower area of the browser.

To open a Class Browser for a subschema selected in the upper area of the copy classes browser

- Right-click and select the **Open Browser For Class** command from the popup menu that is then displayed.

The Class Browser for the selected subschema and class is then displayed.

Editor Pane Enhancements (NFS 67028)

The **JadeTextEdit** and **JadeEditor** control classes have been changed to support the following behaviors.

- Multiple caret selection and editing
- Rectangular selection now takes into account virtual space in the selected region

This section uses the terms *stream format* and *rectangular format* to distinguish between content copied by using single or multiple carets, and content copied by rectangular selection, respectively.

Multiple Carets

You can now create multiple carets and select multiple (non-contiguous) ranges in the editor pane, as follows.

- Multiple carets enable you to enter text at multiple locations at the same time.
- Multiple selections enable you to perform edit actions on selected (non-contiguous) content at the same time.

» To create additional carets

- Press Ctrl+left-click in each location at which you want to add a caret.

The main caret is the color specified in your user preferences (that is, the **Caret** component on the **Editor** sheet of the Preferences dialog) and additional carets are gray.

Note The main caret is the last one that you add.

You can then enter text, paste content, and perform the **Undo** and **Redo** actions at each caret at the same time.

» To make multiple selections

- Press Ctrl and then click and drag your cursor over the content that you want to select.

You can then enter text and perform the **Copy**, **Cut**, and **Paste** actions on selected content at the same time.

Multiple selection copying concatenates the selections in the order in which you made each selection. One instance of the concatenated selections is then pasted in stream format at each caret or selection. Alternatively, if you paste into a rectangular selection, a single instance of the concatenated selections is pasted for each line of the rectangular selection.

Note There is no change to the way single caret selection copying behaves.

Rectangular Selection Virtual Space

The existing rectangular selection functionality has been extended to take into account virtual space in the selected region.

You can create a rectangular selection anywhere within the text in the editor pane. You cannot extend the selection beyond the end of the text.

» To create a rectangular selection

- Perform one of the following actions. Press:
  - Shift+Alt and then use the arrow keys
  - Alt and then click and drag your cursor over the content that you want to select

Note A rectangular selection overrides any single caret or multiple caret selections you have made in the editor pane.

For examples of pasting behavior when using single and multiple carets, see "Pasting Behavior" under "Performing Edit Actions", in Chapter 2 of the *Development Environment User's Guide*.

Editor Clipboard Changes

The Jade Platform development environment editor clipboard toolbar supports copying and pasting of selected rectangular content in the clipboard buffers C, **1** through **9**, or zero (**0**). The rectangular state of copied content is now retained in the clipboard buffer.

Additional clipboard toolbar items do not support selected rectangular content, which is pasted in stream format.

In a clipboard buffer:

- Modifying the content of a rectangular selection retains its rectangular state as long as the content retains at least one carriage return / line feed character (**CrLf**).
- Reducing the content of a rectangular selection to one line removes its rectangular state and the content is pasted in stream format.
- Adding lines to content in stream format in a clipboard buffer does not add the rectangular state.

Extract Format Changes

This topic covers changes to forms definition file extract format options and default values on the Preferences dialog and the Extract dialog. For details about UTF-8 format changes that affect the Jade Report Writer extracts, see "UTF-8 Encoding Support", later in this document.

The default extract format of forms definition files is now the XML DDX format. The legacy DDB format is still supported. If you exported your user preferences before upgrading to Jade 2022.0.01, your default settings are restored after you import your user preferences.

Note DDX files extracted in Jade 2022.0.01 will not load into the Jade 2020 or 2018 releases because they contain additional information and therefore are not backwards-compatible.

The following changes have been made to the Preferences dialog **Source Management** sheet.

- **Unset Schema Extract option 'Create Command File'** check box has been moved to this sheet from the **Schema** sheet and has also been renamed to **Unset Extract option 'Create JCF Command File'**.
- **Use DDX style (xml format) as Default instead of DDB** check box has been moved from the **Schema** sheet to this sheet and has also been renamed to **Extract as DDX (xml format) instead of DDB**.

  To extract definition files (for example, form, database, and ActiveX definitions) in the legacy .**ddb** format instead of the .**ddx** format, uncheck the **Extract as DDX (xml format) instead of DDB** check box.

  This check box is checked by default because .**ddx** is now the default format.
- **Extract as UTF-8 Encoded** check box has been added and is checked by default, which sets UTF-8 with a Byte Order Mark (BOM) as the default encoding format during all extract processes that you perform.

  Uncheck this check box to set the default encoding format as native ANSI or Unicode.

The Extract dialog is now displayed instead of the common Save As dialog when you select the **Extract** command to extract an item; for example, if you select a method in the Methods List and select the **Extract** command from the Methods menu, the Extract dialog is displayed.

As part of this change, the Web Service Consumer Extract dialog has been removed and is replaced by the Extract dialog.

Tip If you are a power user, press the SHIFT key when you select the **Extract** command, to display the common Save As dialog instead of the Extract dialog.

The following changes have been made to the Extract dialog. On the:

- **Extract Options** sheet, the **Extract Forms/Data as XML (ddx file)** check box has been renamed to **Forms/Mappings as XML/DDX** and is checked by default if the **Extract as DDX (xml format) instead of DDB** check box is set on the **Source Management** sheet of the Preferences dialog.

  If you check the **Forms/Mappings as XML/DDX** check box, extracted files are output in .**ddx** format.
- **Schema Options** sheet, the **UTF-8 Encoded** check box has been added. Check this check box to use UTF-8 BOM encoding in the extracted definition file. If you do not check this check box, the extracted definition file is output in native ANSI or Unicode format.

This check box is checked by default if the **Extract as UTF-8 Encoded** check box is set on the **Source Management** sheet of the Preferences dialog.

Finding a Class by Number (NFS 68102, JAD-I-592)

In Jade 2020.0.01, the Find Type dialog allowed partial matching on numbers (NFS 67605, JAD-I-574).

When you check the **Find Class Number** check box, it now no longer clears the **Find** text box when it contains numbers only.

Method Local Parameter, Variable, and Constant Color (JAD-I-632)

When the AutoComplete feature is enabled, the method source in the Jade editor pane now colors the uses of local method parameters, constants, and variables within the method. This enables you to better recognize the type of such elements within the text. Each of the elements is assigned a different color. If AutoComplete is not enabled, these local elements type uses are not colored.

The **Editor** sheet of the Preferences dialog also enables you to select your own color for local parameters, local variables, and local constants. These values are then used for each Jade Support development environment session when you log on with your user identifier. These values are also saved and restored when you extract and restore your user preferences to and from your preferences initialization file.

The sample method source displayed on the right of the **Editor** sheet now includes a parameter, local variables, and a local constant example of the color of these three elements.

Opening a Class Browser for a Form from the Jade Painter (PAR 67435)

A Class Browser is now displayed for the current Jade form when you select the **Browser** command from the File menu in the Jade Painter.

If a Class Browser already exists with that form class selected, it receives focus. If no Class Browser exists with that form class selected, a new Class Browser is opened.

Package Class Name Handling in the Find Type Dialog (PAR 68459)

If there are duplicate imported class names in a schema because multiple packages have the same class name, each class name copy is displayed in the Find Type dialog list box. Selecting any of the duplicate imported class names now displays the correct imported class rather than the same imported class.

In addition, the **mouseHover** event has been implemented for the Find Type dialog list box so that hovering over any entry with the mouse now displays the text returned by the **display** method for the associated object in bubble help.

Painter Hierarchy for Form Dialog (NFS 68385, JAD-I-573)

The Jade Painter Hierarchy for Form dialog now provides the following option buttons that control the way in which controls are ordered under their parent.

- **Order by Name**, which orders the control children by name (the default value)
- **Order by Top/Left**, which orders the control children by the top position and then the left position when two controls have the same top position, providing a visual association with the form layout

In addition, the description for each displayed form and control now includes the top and left positions and the height and width sizes; that is, (T:*top* L:*left* H:*height* W:*width*), as shown in the following example.

```
lblWebSite (T:18 L:12 H:21 W:60) (Label) 'URL'
```


Patch Versioning Dialog Changes

The Extract by Patch Number dialog and Patch Version Extract dialog have been rationalized into one dialog: the Extract Changes for Patch dialog.

The new Extract Changes for Patch dialog has the following new controls.

- **Forms/Mappings as XML/DDX** check box, which is checked by default if the **Extract as DDX (xml format) instead of DDB** check box is set on the **Source Management** sheet of the Preferences dialog.

  If you check the **Forms/Mappings as XML/DDX** check box, extracted files are output in .**ddx** format.

- **UTF-8 Encoding** check box , which is checked by default if the **Extract as UTF-8 Encoded** check box is set on the **Source Management** sheet of the Preferences dialog.

  Check this check box if you want the extract file encoded as UTF-8 with a Byte Order Mark (BOM). If you do not check this check box, the extracted file is output in native ANSI or Unicode format.

For details about the changes to the Preferences dialog **Source Management** sheet, see "Extract Format Changes", earlier in this document.

Refreshing the Process Usages for Class Browser and Classes in Use Browser (JAD-I-667)

The Process Usages for Class Browser and Classes in Use Browser now display the **Refresh** command at the bottom of the Edit menu. In addition, the F5 function key is associated with the **Refresh** command.

Pressing F5 or selecting the **Refresh** command has the same effect as clicking the **Refresh** button on the browser.

Reorganizing Changed Array Definitions (PAR 68306, PAR 68349)

Reorganization now supports some changes to the definition of an array.

Changing the definition of an array may require all instances of that dictionary, both exclusive and shared, to be reorganized.

Note You can change the definition for an array only if the membership type is one of **Binary**, **Decimal**, **String**, or **StringUtf8**.

The supported changes are:

- Decreasing the maximum length of an array definition with membership **Binary**, **String**, or **StringUtf8**. All values must be less than or equal to the new maximum length. Values that are longer than a decreased maximum length cause the reorganization process to fail and an exception to be raised.
- Decreasing the precision or scale factor of an array definition with membership **Decimal**. All values must be less than or equal to the new precision. Values that are longer than a decreased precision cause the reorganization process to fail and an exception to be raised.
- Enabling scaling of entries of an array definition with membership **Decimal**.

A reorganization is not required if the:

- Maximum length of an array with membership **Binary**, **String**, or **StringUtf8** is increased.
- Precision or scale factor of an array with membership **Decimal** is increased, or if scaling of entries is disabled.

RPS Wizard Database Type Column Mappings (PAR 66619)

On the **Define RPS** sheet of the Relational Population Service wizard, if you change the selected database type from **SQL Server 2000** or **SQL Server 2005** to **SQL Server 2008**, you are now prompted to confirm whether you want to retain existing legacy column mappings.

Click the **Yes** button to retain existing legacy column mappings; for example, **datetime**, **image**, **text**, **ntext**, and so on. Alternatively, click the **No** button, to use the new default column mappings; for example, **datetime2**, and so on.

RPS Wizard Selected Class Display (PAR 67654, JAD-I-528)

The classes selected for inclusion in the RPS mapping in the **Available Classes** list box at the left of the **Select Classes for RPS** sheet of the Relational Population Service wizard are now displayed as disabled (that is, gray).

Status Line Positioning (PAR 68131)

There was a conflict between the value of the **StatusLine** control **autoSize** property set to **true** and the use of the **parentAspect**, **relativeLeft**, and **relativeWidth** properties on the child controls. When the value of the **autoSize** property is **true**, the status line first positioned any children that are not fully visible, if possible. The parent aspect and **relativeLeft** and **relativeWidth** of the child control is then invoked but on the repositioned values. (Note that the relative properties are ignored when the **parentAspect** property is used.)

This behavior has now been changed so that **parentAspect**, **relativeLeft**, and **relativeWidth** properties used by child controls of a status line control work as expected regardless of the value of the **StatusLine** control **autoSize** property value.

When the value of the **StatusLine** control **autoSize** property is **true**:

- If a child control **left** property value position is less than zero, the control is moved to be zero (**0**) when the **parentAspect**, **relativeLeft**, and **relativeWidth** property values of the child do not affect the horizontal position (not stretch horizontal, anchor right, and centered horizontal, and the **relativeLeft** and **relativeWidth** property values are **false**).
- If a child control is not fully visible horizontally, the child is right-aligned in the status line control if it can be fully displayed or positioned at zero (**0**) if it cannot when the **parentAspect**, **relativeLeft**, and **relativeWidth** property values of the child do not affect the horizontal position (not stretch horizontal, anchor right, and centered horizontal, and the **relativeLeft** and **relativeWidth** property values are **false**).
- If a child control top position is less than zero (**0**), the control is moved to be zero when the **parentAspect** property value of the child does not affect the vertical position (not stretch vertical, anchor bottom, or centered vertical).
- If a child control is not fully visible vertically, the child is bottom-aligned in the **StatusLine** control when the **parentAspect** property value of the child does not affect the vertical position (not stretch vertical, anchor bottom and centered vertically).
- The values of the **relativeTop** and **relativeHeight** properties of child controls are always set to **false**, as their functionality is not compatible with auto-sizing the height of the **StatusLine** control (as has always been the case).
- All **parentAspect** flag values and **relativeLeft** and **relativeWidth** values are applied.

The height of the **StatusLine** control is then determined by analyzing the child control as follows, to determine the maximum height required. For a child control that does not have a fixed height and has the **parentAspect** property with the:

- **ParentAspect_StretchBottom** flag set, the height required is the **top** position, **height**, and **parentBottomOffset** property values of the child.
- **ParentAspect_AnchorBottom** flag set, the height required is the **height** and **parentBottomOffset** property values of the child; otherwise, the **height** of the child control.

These changes mean that the **parentAspect** property can be successfully used to position controls within a status line control.

Suspending Parent Alignment when Positioning Controls (NFS 68413)

In the Jade Painter, it was difficult to position docking controls to meet your requirements when the **alignContainer** and **alignChildren** properties are set.

To improve this situation, the Jade Painter Layout menu now provides the **Suspend Parent Alignments** command. When this command is unchecked (the default), the **alignContainer** and **alignChildren** properties of controls behave as normal.

When the command is checked, the **alignContainer** and **alignChildren** properties of controls are treated as though the property values are zero (**0**) so that no automatic alignment occurs, which enables you to position controls to meet your requirements. When you uncheck the command again, normal behavior is resumed and the required alignments are applied.

Notes If you attempt to save the form when the command is checked, a message box is displayed asking if you want to continue. If you indicate that the save process is to continue, the form is saved using the current control positions that could differ from those that apply when normal alignment operations are in effect. If you do not want to continue, the save process is cancelled.

Suspending alignment applies only to the **alignContainer** and **alignChildren** properties and does not affect the behavior of the **parentAspect** property or the **StatusLine** control.

Unicode Surrogate Pair Character Support (PAR 68066)

Jade now supports 16-bit Unicode code characters that are greater than **0xFFFF**, which includes emojis, mathematical symbols, and others.

The term *emoji* in the Jade Platform product information includes any type of surrogate pair characters. Such characters:

- Are stored as *surrogate pair* characters.
- Require the use of two 16-bit values to store the value. These Unicode values are stored as two encoded characters.

The actual value of the original Unicode character is split into two and stored in the lower part of each of the two characters, with **0XD800** added to the first 16-bit character and **0xDC00** added to the second. For example, the Unicode value **0x1F783** that represents an emoji is stored as **0xD83c** and **0XDF83**.

Jade now handles surrogate pair characters; in particular, the conversion from **String** to **StringUtf8** and **StringUtf8** to **String** primitive types. As a result of this, emoji characters can be included in text in a Jade system with some limitations.

The following code fragment is an example of Unicode surrogate pair character handling.

```
strUtf8 := #[f0 9f 8e 85].Binary.StringUtf8;// Father Christmas emoji
str := strUtf8.String;
write str;
write strUtf8;
```

A Unicode UTF-8 encoding table and emoji characters can be found at https://www.utf8-chartable.de/unicode-utf8-table.pl?start=127872.

The following is a list of notes about surrogate pair emoji usage in Jade.

- Emojis can be used only in a Unicode Jade system because they cannot be represented in ANSI.
- Emojis can be represented using **StringUtf8** in an ANSI Jade system, but they cannot be converted to a **String** value as there is no ANSI representation.
- Emojis can be included in any text displayed or printed, except for the rich text restriction later in this list.
- Emojis can be copied and pasted to and from **TextBox** controls. The emoji selection window (displayed using the Windows key + period (**.**), or dot, character key combination) can be used to paste a selection.
- Emoji characters can be converted to and from a **StringUtf8** primitive type, which means that can be included in web text data.
- An emoji character cannot be stored in a **Character** property because it requires two characters, so it will not fit.
- Because an emoji cannot be stored in a **Character**-type property, using an arrow key can leave the cursor positioned in the same place and may require multiple presses to step over each part of the emoji.
- Emojis can be made up of multiple surrogate pair values that are overlayed on each other to get the final displayed representation.
- The Jade length of a string that includes emojis is the number of 16-bit characters, *not* the number of displayed characters.
- Locate emoji characters in text by checking whether the first character is greater than **0xD800** and the second character is greater than **0xDC00**.
- The **JadeRichText** control, which is a Microsoft control, does not support emoji characters.
- Use of emoji characters in the Jade editor is not yet recommended. They can be inserted in methods and will display correctly. However, there are problems because the editor counts each surrogate pair as one displayed character, while Jade counts them as two. The positioning, therefore, of the cursor programmatically so that when the compiler shows the text that is in error or the debugger showing the current line may not be correct.

  In addition, the editor pane does not support the selection of an emoji (that is, by pressing the Windows and period keys).

Viewing Compiler Errors, Warnings, and Information Messages

The new Compiler Output Viewer enables you to view error, warning, and information messages that occur when you compile schemas, classes, and methods in the Jade Platform development environment. For details, see "Using the Compiler Output Viewer", in Chapter 4 of the *Development Environment User's Guide*.

To configure or opt-out of the Compiler Output Viewer, use the new **Compiler Output** sheet on the Preferences dialog. For details, see "Maintaining Compiler Output Options", in Chapter 2 of the *Development Environment User's Guide*.

Workspace Refactoring (PAR 68352)

The Refactor menu is disabled for a **Workspace** method because a **Workspace** has no **self** class object and therefore the menu is not relevant.

Jade Initialization File

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

Allowing Reorganizations to Use Disk Cache (PAR 68767)

By default, the reorganization process uses memory mapped files for output (.**reo**) files and input (.**dat**) files if the reorganization was read-only. For large reorganizations, this means that an 80G byte database file, for example, uses at least one, and possibly two, 80G byte chunks of virtual memory, which could exhaust the virtual memory of the server. The memory would also not show up as being consumed by Jade, as it belonged to the operating system. With multiple reorganization workers, this problem would be multiplied.

The [JadeReorg] section of the Jade initialization file can now contain the **UseDiskCacheForReorg** parameter, which controls how the database manages reorganization files. The default value for this parameter is **false**, which means that memory mapped files are used.

Set the parameter to **true** if you want reorganizations to use database disk cache. All reorganization input and output then uses the database disk cache, and obeys all its settings.

Note When this parameter is set to **true**, apply it to both the primary and secondary databases, if applicable.

Jade Report Writer

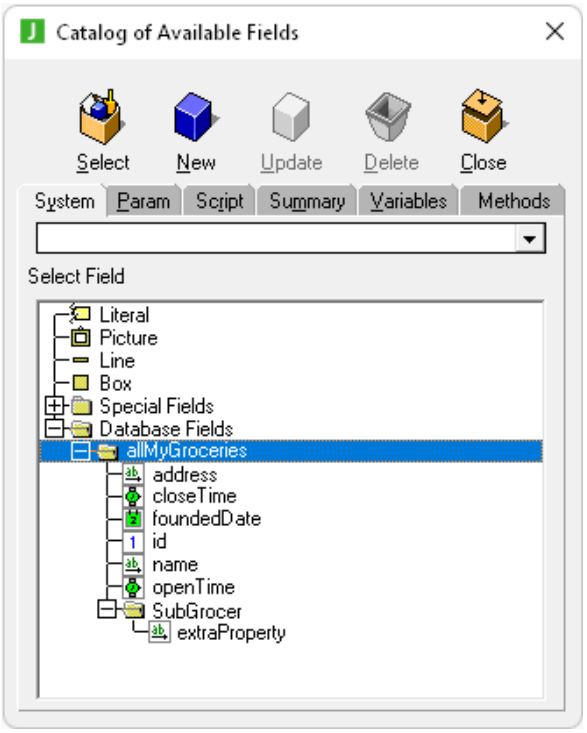
This section describes the Jade Report Writer changes in this release.

Report Writer (JAD-I-552)

The following changes have been made to the Jade Report Writer.

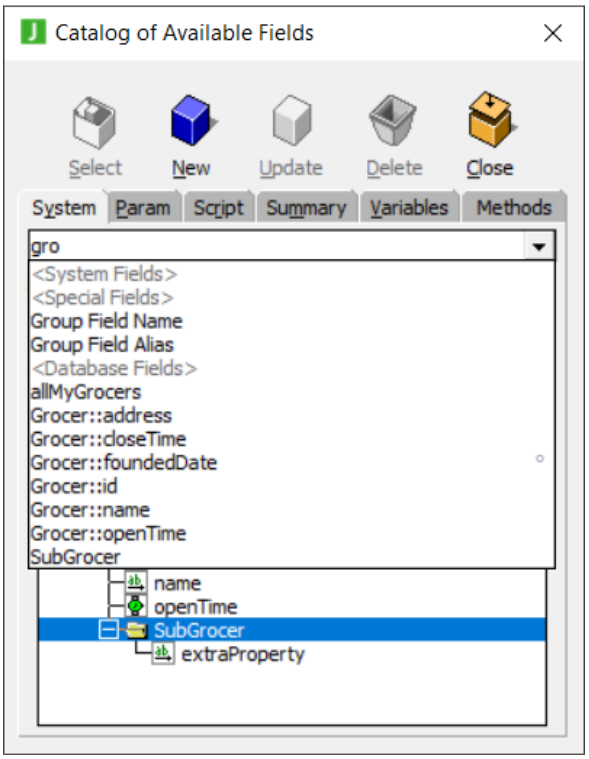
- Ability to copy and paste fields (for example, literals, database fields, and so on) between sections in the same report or different reports with the same root collections.

- The **System** sheet of the Catalog of Available Fields dialog now groups attributes and collections under their classes and allows the class to be collapsed, as shown in the following image.



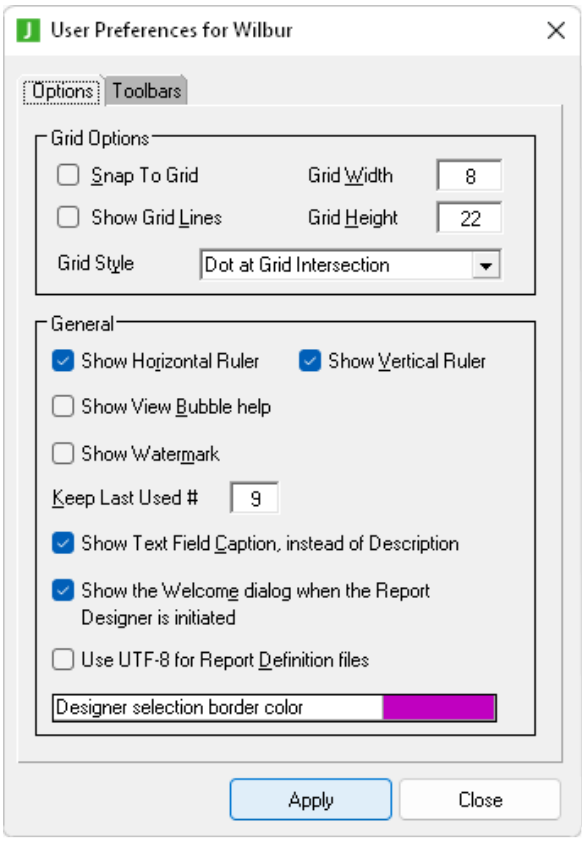
In addition, the Catalog of Available Fields dialog now provides a combo box that enables you to quickly search for fields. The F4 function key also enables you to start your search.

All fields in the list box are searched; not just exposed classes, as shown in the following image.



- You can now change the color of the dashed border around the currently selected fields on a report so that they stand out. The default border color is now purple (it was blue in earlier releases).

  The **Designer selection border color** control on the **Options** sheet of the User Preferences dialog, shown in the following image, enables you to configure the border color of selected fields on a report to a color of your choice.



UTF-8 Encoding Support

As part of the extract format changes in the Jade Platform development environment, Jade Report Writer now supports extracting and reloading the following files encoded as UTF-8 with a Byte Order Mark (BOM).

- Report Definition files (.**rwr**)
- Report View Definition files (.**rwv**)
- Report Folder files (.**rwf**)
- Report User Options files (.**rwu**)
- Report System Options files (.**rwo**)
- Report All Definitions and Options file (.**rwa**)

The **Use UTF-8 file encoding** check box is now displayed on the respective dialogs for extracting and unloading the files in the previous list. Check this check box to output files encoded as UTF-8 with a Byte Order Mark (BOM). If you do not check this check box, the file is output in native ANSI or Unicode format.

In addition, the dialogs that enable you to extract or unload data are now resizable.

The **Options** sheet of the User Preferences dialog enables you to specify the default setting of the **Use UTF-8 file encoding** check box, using the following new check box.

- **Use UTF-8 for Report Definition files** check box. If you check this check box, the **Use UTF-8 file encoding** check box is checked by default on dialogs for extracting and unloading Jade Report Writer files. By default, this check box is not checked; that is, native ANSI or Unicode is the default format.

The **JadeReportWriterReport** class now provides the **setOutputFileEncoding** method, which has the following signature.

```
setOutputFileEncoding(fileKind: Integer; unicodeBOM: Boolean);
```

The **setOutputFileEncoding** method sets the file kind and Unicode Byte Order Mark (BOM) of the file that is created when the report runs. The **setOutputFileEncoding** method parameters are listed in the following table.

Parameter	Description
fileKind	Contains the kind of file that is to be output, represented by one of the following constant values. ■ **0** (Native); that is, existing ANSI or Unicode behavior ■ **File.Kind_ANSI** (**2**), only for an ANSI system ■ **File.Kind_Unicode** (**3**), only for a Unicode system ■ **File.Kind_Unicode_UTF8** (**8**), Unicode Transformation Format (UTF) 8 bit
unicodeBOM	Specifies whether a Unicode BOM is present in the file, represented by the **File** class **unicodeBOM** property. **Note** Applies only if **File.Kind_Unicode_UTF8** is specified for the **fileKind** parameter.

For details, see "JadeReportWriterReport Class Constants", in Chapter 1 of the *Encyclopaedia of Classes* and for details about extract format changes in Jade, see "Extract Format Changes", earlier in this document.

JadeTextEdit Class setFileEncoding and getFileEncoding Methods

The **JadeTextEdit** class now provides **setFileEncoding** and **getFileEncoding** methods, which have the following signatures.

```
setFileEncoding(kind: Integer; unicodeBom: Boolean);

getFileEncoding(kind: Integer output; unicodeBom: Boolean output);
```

The **setFileEncoding** method sets the file kind and Unicode Byte Order Mark (BOM) to be used when the **saveTextToFile** method is called.

The **getFileEncoding** method returns the current file kind and Unicode BOM values associated with the control. It is updated to reflect the values for any file loaded using the **loadTextFromFile** method or manually set using the **setFileEncoding** method.

For details about:

- Extract format changes in Jade, see "Extract Format Changes", earlier in this document.
- UTF-8 format changes that affect the Jade Report Writer extracts, see "UTF-8 Encoding Support", earlier in this document.

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*.

Mocking Framework for Test Automation

Jade now implements a **Mocking** framework, which enables you to create mock objects that mimic the interface of real objects and simulate their behavior. In addition, the mock provides information about which methods of an object were called and in what order.

The **Mocking** framework is implemented as a package exported from a user schema; *not* RootSchema classes. As the framework is open source, you can modify and extend it.

Download the **Mocking** framework files from the **JADE-Mocking** link at https://github.com/jadesoftwarenz.

Real Type roundedTo and truncatedTo Methods Accuracy (PAR 68509)

Changes have been made to the implementation of the **Real** primitive type **roundedTo** and **truncatedTo** methods to improve their accuracy. The value returned may differ, due to the approximate nature of floating point arithmetic. For example, the closest floating point representation of **0.155** is **0.154999...**, which rounds down to **0.15** and not up to **0.16** at two decimal places.

The following code fragment outputs **"0.15"** in Jade version 2020 and **"0.16"** in Jade version 2022.

```
write 0.155.roundedTo(2);
```

REST Services

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

REST Request PDF Data Format (PAR 68114)

The **JadeRestRequest** class now provides the **DataFormat_PDF** class constant, which has an integer value of **12** and corresponds to the **application/PDF** HTTP content-type.

When using this format, set the **body** parameter as required by the REST API specification.

REST Services Enhancements

This section describes the Jade REST services enhancements in this release (which are documented in the respective topics in Chapter 11 of the "*Developer's Reference*).

ISO 8601 Date and Time Support

In previous releases, when a **Date**, **Time**, **TimeStamp**, or **TimeStampOffset** primitive type was returned from a web service, it was formatted in MicrosoftDateFormat, an old and long-deprecated format that many modern tools do not understand; for example, \\/**Date(1488389116170-0500)**\\/. To maintain existing behavior, you can still use the old MicrosoftDateFormat.

Jade 2022 now supports the ISO 8601 format, as specified in https://www.rfc-editor.org/rfc/rfc3339.

The use of ISO 8601 is enabled on the Define Application dialog for your REST web application by checking the new **Use ISO 8601 Time** check box on the **Web Options** sheet. This check box is unchecked by default. (This functionality is displayed on the **Web Options** sheet of the dialog only when the application selected in the **Application Type** list box on the **Application** sheet of this dialog is **Rest Services** or **Rest Services, Non-Gui**.)

When the **Use ISO 8601 Time** check box is checked, the application formats all **Date**, **Time**, **TimeStamp**, and **TimeStampOffset** primitive types in ISO 8601 format rather than MicrosoftDateFormat.

In addition, when a **TimeStamp** is returned from the web service in ISO 8601 format, it includes time zone offset information, similar to a **TimeStampOffset**. This is required because the consumer of the web service would not otherwise know the time zone to which the **TimeStamp** is local. As such, all **TimeStamp** primitive types are converted to UTC time and an offset included that represents by how many hours and minutes to offset the **TimeStamp** to get to local time.

You can specify the time zone that should be assumed for **TimeStamp** primitive types by typing or selecting the appropriate time zone IANA name from the **Timestamp TimeZone** combo box on the **Web Options** sheet of the Define Application dialog. The combo box is populated with **Etc/UTC (Z)** as well as the time zone in which the database server is located. You can select any time zone you require, by typing it in full.

ISO 8601 with the JadeJson class

These changes are also applicable to the **JadeJson** class, which provides the following new properties and method to enable the new functionality.

- **useISO8601** property, which has a **Boolean** value. Set the property to **true** to use the ISO 8601 format instead of the MicrosoftDateFormat.
- **offsetForTimeStamps** property, which has an **Integer** value. Set this property to the number of hours by which you want to offset **TimeStamp** values. Alternatively, you can set this from a time zone name by using the new **setOffsetForTimeStamps** method.
- **setOffsetForTimeStamps** method, which enables you to specify the name of a time zone that can be the IANA or Windows name, and sets the **offsetForTimeStamps** property to the value of the offset of the specified time zone.

Multipart Form Data Encoding Server-Side Support

The **multipart/form-data** is a content type, defined by https://www.ietf.org/rfc/rfc2388.txt, used for **POST** requests. It is typically used when uploading the contents of files that may be very large and have special characters or be in binary format.

In a **multipart/form-data** request, the HTTP body is separated into multiple parts, with each part containing some information about the file or data, the data itself, then a delimiter prefixed with two dash characters.

```
--905191404154484336597275426 6
Content-Disposition: form-data; name="text"

text default
--905191404154484336597275426 6
Content-Disposition: form-data; name="file1"; filename="a.txt"
Content-Type: text/plain

Content of a.txt.

--905191404154484336597275426 6
Content-Disposition: form-data; name="file2"; filename="a.html"
Content-Type: text/html

<!DOCTYPE html><title>Content of a.html.</title>

--905191404154484336597275426 6--
```

The delimiter is specified in the **Content-Type** HTTP header after **multipart/form-data**, and prefixed with **boundary=**; for example:

```
Content-Type: multipart/form-data; boundary=905191404154484336597275426 6
```


New Server-Side Support

In earlier releases, Jade automatically converted the **multipart/form-data** body into **url/form-encoded**, where each bit of form data was separated by an ampersand character (**&**) and the boundaries, **Content-Disposition**, and **Content-Type** information were removed. In addition, this data was not sent through to the Jade REST method as parameters, so it relied on user code to extract it from the body (for example, by reimplementing the **JadeRestService** class **processRequest** method).

In Jade 2022, Jade provides the option to leave the HTTP body as it is and send the data through to the Jade REST method as parameters. By default, the existing behavior is maintained, so this feature must be explicitly enabled by setting the value of the new **UseNewStyleMultipart** parameter to **true** in the **jadehttp.ini** file under the application for your web service; for example:

```
[JadeDefaultApp]
TcpConnection=localhost
TcpPort=6124
ApplicationType=RestServices
MessageTimeout=5
MinInUse=1
MaxInUse=1000
MaxMessageSize=1000000
MinMessageSize=10
UseNewStyleMultipart=true
PurgeDirectoryRule=default
PurgeFileAge=default
VirtualDirectory=
```

Note When the **UseNewStyleMultipart** parameter is *not* set to **true**, not only will the data not be extracted to parameters but the behavior in earlier releases of converting the body is also preserved.

OpenAPI Import Wizard Enhancements

The OpenAPI Import Wizard now provides:

- Warnings on the second sheet of the wizard; for example, if the specification contains a class that does not have any properties.

  These do not prevent the loading of the specification but may indicate that something is wrong or missing.

- Ability to see and rename any collection proxy classes that are to be generated for the specification on the third sheet of the wizard.

  In earlier releases, these were just generated at the end of the import process, depending on need, and were neither visible nor able to be renamed in the import wizard.

REST Client Enhanced Header Support

The following new **JadeRestRequest** class constants and methods enable you to add, modify, remove, and view the headers that are to be sent in a REST request using the Jade REST client.

- **JadeRestRequest** class **AddHeader_Forbidden** and **AddHeader_Success** constants.
- **JadeRestRequest** methods:
  - **addHeader**, which adds or modifies the header specified in the **key** parameter with the value specified in the **value** parameter
  - **getAllHeaders**, which returns all headers that will be included in the REST request that can be modified
  - **getHeaderValue**, which returns the value of the header specified in the **key** parameter if it has been set

- **isHeaderForbidden**, which returns **true** if the header specified in the **header** parameter is one that is forbidden and cannot be modified
- **setAcceptHeader**, which sets the **Accept** header to the value specified in the **value** parameter
- **setContentTypeHeader**, which sets the **Content-Type** header to the value specified in the **value** parameter
- **setPragmaHeader**, which sets the **Pragma** header to the value specified in the **value** parameter
- **setUserAgentHeader**, which sets the **User-Agent** header to the value specified in the **value** parameter

The special forbidden headers that cannot be modified are listed at:

https://developer.mozilla.org/en-US/docs/Glossary/Forbidden_header_name

For details, see Volume 1 (that is, **EncycloSys1.pdf**) of the *Encyclopaedia of Classes*. See also "REST Client Header Support", in Chapter 11 of the *Developer's Reference* for a usage example.

REST Services Pagination

When returning large collections from a REST web service, it is important to ensure that an individual response is not excessive. The standard way of limiting the size of a response to a subset of the large collection is pagination, where only some of the collection is returned in addition to information about how to get the next part of the collection in a subsequent REST request.

In Jade 2022, Jade supports the paginating of responses when using the **JadeRestService** class to provide web services, as well as providing a mechanism to easily get the next page when using the **JadeRestClient** class to consume web services that employ common pagination strategies.

Jade supports the following main styles of pagination from both the server side and client side of a REST service.

- With envelope-based pagination, the server returns an envelope object that contains both the paginated results and the pagination information.
- With link header pagination, the server returns the paginated results directly in the HTTP response body, with the pagination information (for example, a link to the next page of results) in the HTTP response headers, as defined by:

  https://www.rfc-editor.org/rfc/rfc5988.html#section-5

Jade now provides the following new interface and class, as well as changes to entities in existing REST service classes.

- New **JadePaginationEnvelopeIF** interface, which is used by the **JadeRestResponse** class **parsePaginationEnvelope** method to ensure that an envelope is able to be used by Jade to iterate through the pages of results. When consuming third-party REST services, you will need to define a class matching its envelope definition, ensuring it implements the **JadePaginationEnvelopeIF** interface.
- New **JadePaginationEnvelope** class, which is used as the return type for a Jade REST method that would otherwise return a collection in order to provide pagination information such as a link to the next page of results in the HTTP response body. Use the **loadCollection** method to load the collection you would otherwise return directly into the envelope.

  This class provides the following properties and methods.

  - **JadePaginationEnvelope** properties:
    - **data**, which is a subset of the paginated collection; that is, one "page" of the collection
    - **next**, which is a link to the next page of results

- **JadePaginationEnvelope** methods:
  - **getNextPage**, which returns the link to the next page of results and fulfills the **JadePaginationEnvelopeIF** interface
  - **loadCollection**, which contains a collection and a **JadeRestService** object, and sets the values of the **data** and **next** properties based on the requested pagination

- The existing **JadeRestRequest** class provides the following new class constants and methods.
  - **JadeRestRequest** class **QueryString_SortAscending** and **QueryString_SortDescending** constants.
  - **JadeRestRequest** methods:
    - **setPaginationKeyAndLimit**, which sets the limit query string for the request to the value specified in the **limit** parameter and adds a query string for the key and value pair specified in the **keyName** and **keyValue** parameters
    - **setPaginationLimitOnly**, which sets the limit query string for the request to the value specified in the **limit** parameter
    - **setPaginationOffsetAndLimit**, which sets the limit query string for the request to the value specified in the **limit** parameter and adds a query string for the offset specified in the **offset** parameter

- The existing **JadeRestResponse** class provides the following new properties and methods.
  - **JadeRestResponse** properties:
    - **lastPage**, which contains the URL of the last available page of results
    - **firstPage**, which contains the URL of the first available page of results
    - **nextPage**, which contains the URL of the next available page of results
    - **previousPage**, which contains the URL of the previous page of results
    - **hasMorePages**, which specifies whether there are more pages available on the server
    - **url**, which specifies the URL to which the request was made that generated the response
  - **JadeRestResponse** methods:
    - **getNextPage**, which makes another call to the same REST API using that URL if the value of the **nextPage** property is set to a valid URL
    - **parsePaginationEnvelope**, which attempts to deserialize the JSON string in the **data** property of the **JadePaginationEnvelope** class to an object of the class specified in the **envelope** parameter

For details, see Volume 1 (that is, **EncycloSys1.pdf**) of the *Encyclopaedia of Classes*. See also "REST Services Pagination", in Chapter 11 of the *Developer's Reference* for usage examples.

Standardized URLs for Web Services

In earlier releases, Jade REST URLs had to be of the following form.

```
server-name/IIS-app-name/jadehttp.dll/resource.JSON?app-name
```

In this format:

Entity	Description
server-name	Base part of the URL, which is where to find the IIS; for example, **localhost**
IIS-*app-name*	Name of the application in IIS that links up to the **jadehttp.dll**; for example, **JadeRestSite**

Entity	Description
jadehttp.ini	Makes **ISAPI-dll** in IIS map to **jadehttp.dll**, which is the handover point from IIS to Jade
.JSON	Optionally specifies whether the response is formatted in JSON or XML for a GET request (the default value is JSON)
?*app-name*	Used to look up the application in the **jadehttp.ini** file

In Jade 2022, the following enhancements and suggestions enable you to simplify the Jade REST URL.

Default Application

You can define default application details that are to be used if no application name is specified in the query string in the **jadehttp.ini** file. This application is defined the same way as any other application, except that the section header is [JadeDefaultApp].

Using HTTP Header instead of .JSON or .XML

Jade REST servers will respect the **Content-Type** and **Accept** headers, so you no longer need to include **.JSON**, **.JSONN**, or **.XML** in the URL. If they are included, they will have first priority; otherwise the **Accept** header will be used to determine the type in which to return serialized objects.

You can also use the **Content-Type** header for this purpose and it will be respected, but the **Accept** header is more correct for specifying the format in which to return the serialized object.

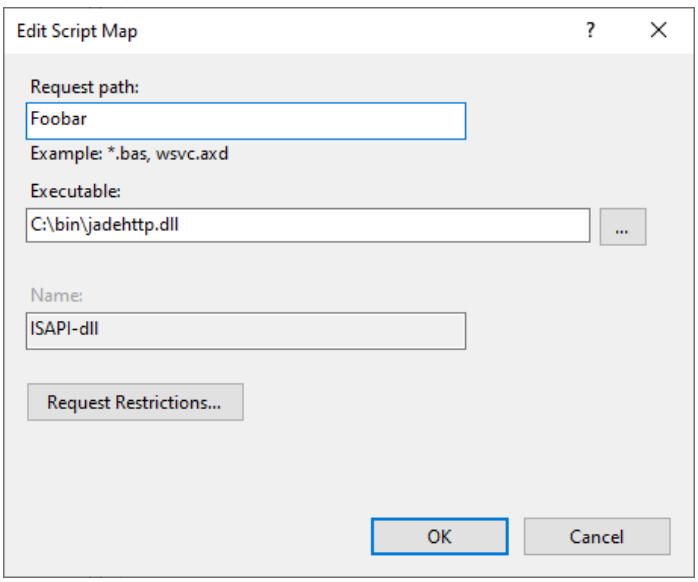
jadehttp.dll not Needed in the URL

Although this is not new to Jade 2022, with a bit of IIS configuration you can rename **jadehttp.dll** in the URL to whatever you want. IIS needs only to know how to break up the URL into its component parts. For example, in the **http://localhost/JadeRestSite/jadehttp.dll/foo?RestApp** URL. The first part gets to the IIS; that is, to **http//localhost**, the second part specifies the web site; that is, **/JadeRestSite**, and the third part lets IIS know that the **jadehttp.dll** ISAPI is to be used.

For information about configuring IIS, see "Configuring the WebSocket Protocol in IISMgr", in the *Installation and Configuration Guide*; "Web Server Setup", in the *Web Services Tips and Techniques* white paper; or "Configuring IIS", in Part 1 of the *Erewhon Demonstration System Reference*.

To rename jadehttp.dll in the IIS configuration

1. To change the **jadehttp.dll** part of the URL to the name of your choice (for example, to **Foobar**), you need to change the request path in the Edit Script Map dialog.

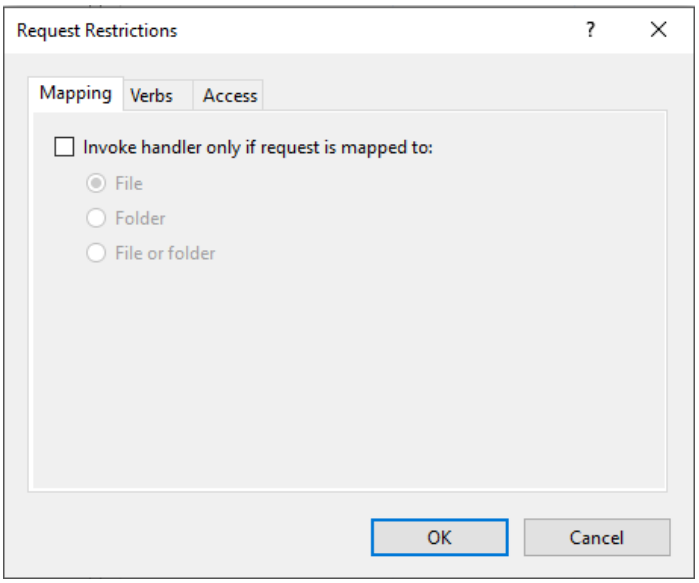


The new URL would be **http://localhost/JadeRestSite/Foobar/foo?RestApp**. You can change the **Foobar** request name to any name that is appropriate for your web service.

Notes For this to work, you need to turn off the request restrictions.

In addition, if you have a custom IIS mapping with a request path that uses a wildcard character (for example, ***.dll**), you must manually add the **allowPathInfo="true"** attribute to the **web.config** file created by IIS Manager to ensure the **pathIn** parameter of the **processRequest** method honors the specified partial path.

2. Click the **Request Restrictions** button, to determine if restrictions are set for the request path. The Request Restrictions dialog, shown in the following image, is then displayed.



3. If the **Invoke handler only if request is mapped to:** check box is checked (that is, set) on the **Mapping** sheet, uncheck it.

4. Click the **OK** button. Alternatively, click the **Cancel** button to abandon the action.

Running a Workspace in a Deployed System (PAR 68293)

If your Jade system has no developer licenses or has the Jade schema database files marked as offline, you can now run encrypted Workspaces supplied by Jade Support.

The **jadclient** executable program now provides the **executeEncryptedWorkspace** command, which has the following syntax.

```
jadclient ini=jade-initialization-file
          path=database-path
          executeEncryptedWorkspace=file-name
```

The encrypted Workspace is read from the specified file, which must have been provided by Jade Support. (Use of this command in **jadclient** is equivalent to opening an encrypted Workspace file and executing it in the Jade Platform development environment.)

The **schema** and **app** commands are not required, and are ignored if they are specified.

The encrypted Workspace can be run in single user or multiuser mode, depending on the requirements of the Workspace.

Schema Load Utility Enhancements

The Load Schema Source dialog (called the Load Schema dialog in earlier releases) accessed from the **Jade Loader** program icon in your Jade program folder, can now be resized. The **More Options** button has been replaced with an **Advanced Options** button and instead of the dialog expanding, a separate modal Advanced Load Options dialog is now displayed. This allows the Load Schema Source dialog to stretch horizontally unconstrained and provides greater visibility of the various path parameters. The font size has also increased slightly, as it now uses the standard Jade Platform development environment font.

The batch mode **jadloadb** program now provides the following loader arguments that make the file types more obvious.

- [ddxFile=*file-name*], which is the optional fully qualified name of the XML (DDX format) file
- [mulFile=*file-name*], which is the optional fully qualified name of the multiple schema extract file

These loader arguments are the same as the existing **ddbFile** (DDB format) file and **commandFile** loader arguments (which have not changed) but have more-appropriate names.

Security

This section describes the Jade Security changes in this release.

Security Supplementary Course Module

The https://www.jadeplatform.com/developer-centre/learn/additional-learning web page now enables you to read and optionally download the new **Security** supplementary course module, which is in Portable Document Format (PDF) only.

SSL-Enabled Connections (JAD-I-431)

SSL-enabled connections now include all client node to database connections, that is:

- Application server to database server
- Standard client to database server
- .NET client to database server
- ODBC standard client to database server

These connections are controlled with the **ClientToRap** parameter in the [Connections] section of the Secure Jade SSL configuration file.

In addition, the Jade ODBC Thin Client Setup dialog now enables you to specify the name and path of the SSL configuration file.

Setting the File Kind for the JadeXMLDocument Class writeToFile Method

The **JadeXMLDocument** class now provides the **setFileEncoding** method, which has the following signature.

```
setFileEncoding(kind: Integer; unicodeBom: Boolean) updating;
```

This method sets the file kind and Unicode Byte Order Mark (BOM) to be used when the **writeToFile** method is called. If the **setFileEncoding** method is not called, the encoding of the file is native ANSI or Unicode.

String Primitive Type startsWith__ and endsWith__ Methods

The **String** primitive type now provides the **startsWith__** and **endsWith__** methods, which have the following signatures.

```
startsWith__(prefix: String): Boolean;

endsWith__(suffix: String): Boolean;
```

The **startsWith__** method of the **String** primitive type returns **true** if the receiver starts with the string specified by the **prefix** parameter, or if the **prefix** parameter is an empty string (null). In all other cases, including if the receiver is an empty string, the method returns **false**.

The **endsWith__** method of the **String** primitive type returns **true** if the receiver ends with the string specified by the **suffix** parameter, or if the **suffix** parameter is an empty string (null). In all other cases, including if the receiver is an empty string, the method returns **false**.

Note If you defined your own **String** primitive type **startsWith__** or **endsWith__** methods in an earlier release, you must rename them *before* you upgrade to Jade 2022, as Jade's new **startsWith__** or **endsWith__** methods will conflict with any existing **String** method named **startsWith__** or **endsWith__**.

Type Class Hierarchy Tree Methods

The **Class** class **getConstantInHTree** and **getMethodInHTree** methods have now been moved to the **Type** superclass, and are reimplemented by the **Class** class. In addition, the **getConstantsInHTree** method is now defined in the **Type** class and reimplemented in the **Class** class. For details, see the **Class** and **Type** classes in the *Encyclopaedia of Classes.*

Unit Test Runner Form (NFSes 65290, 65287)

In this release, the Unit Test Runner form has been updated to include some new usability features. The following changes have been made to the **Select Tests** pane of the form.

- It now displays the test classes in a hierarchical list. A top-level item is displayed for the current schema and subitems are added for all test classes in the schema. Individual tests are still represented as leaf items.
- The number of tests for each class is now displayed beside each test class. The total number of tests for a schema is also displayed beside the top-level schema item.
- If you select a class that has no local tests but it has subclasses with unit tests, clicking the **Run** button runs all tests from all subclasses.
- The text color of individual tests now reflects their status; that is, they are displayed in blue if they are yet to be run, green if the test succeeded, and red if the test failed.
- Double-clicking a method or class in the **Select Tests** pane now moves the focus to the Jade Platform development environment and navigates to the selected entity in the relevant Hierarchy Browser or opens a new browser if one does not already exist for the current schema.

Other changes made to the Unit Test Runner form are as follows.

- The **Refresh** command in the View menu (or pressing F5) now runs or reruns all currently selected tests.
- New **Expand all** and **Collapse all** icon buttons have been added to the top right corner of the **Select Tests** pane.
  - The **Expand all** button expands the list box, making all tests visible.
  - The **Collapse all** button collapses the list box, showing only the schema item and its direct children. This represents all direct children of the **JadeTestCase** class.

