SPANNING JAVA & .NET

RELEASE NOTES

JNBridgePro™ V12.0

April 15, 2025

SYSTEM REQUIREMENTS

.NET-side

- When using .NET Framework: Windows 11, Windows 10 (version 1507 and later), Windows 8.1, Windows 7 SP1, Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, Windows Server 2008 R2 SP1. Both 32-bit and x64 are supported.
- When using the .NET (formerly .NET Core): Any version of Windows or 64-bit Linux on which .NET runtime is supported.
- At least one of the following .NET runtimes:
  - .NET Framework runtime 4.8 (which may be obtained by installing Visual Studio, the .NET Framework SDK, or the .NET Framework redistributable).
  - .NET 8 runtime (formerly called NET Core). We currently support both x86 and x64 Linux, as well as Windows.

Java-side

- Any operating system that will support the underlying JVM or application server
- Java Software Development Kit (JDK) or Java Runtime Environment (JRE) versions 8 through 24.0. If using a J2EE application server, any such application server supporting Java EE 8 through Jakarta EE 11.

Plug-ins:

- For the Visual Studio plug-in, Visual Studio 2017, 2019 or 2022.
- For the Eclipse plug-in, Eclipse 3.2 through 4.13 (32-bit or 64-bit) and .NET Framework 4.8

FEEDBACK AND SUPPORT

- If you encounter any bugs or other support issues, please contact support@jnbridge.com, or call 303-545-9371, for assistance.
- We would appreciate any and all feedback about this product, including any issues connected with the software, installation, usability, performance, missing features, or bugs. Please send feedback by e-mail to support@jnbridge.com.

NEW IN VERSION 12.0

- JNBridgePro has been updated to a single .NET Framework variant supporting .NET Framework 4.8. Earlier versions contained two .NET Framework directories, *2.0-targeted* and *4.0-targeted*. JNBridgePro v12.0 now contains the single .NET Framework directory *4.8-targeted.*
- JNBridgePro has been migrated to Visual Studio 2022 (Platform Toolset v143) and Windows SDK 10.0, using currently supported Microsoft runtime libraries. This is a major update from several previous versions of JNBridgePro and may require the installation of new VC++ runtime libraries. See the installer in the *Visual C++ runtime libraries* folder within the installation or look for Microsoft Visual C++ Redistributable downloads online.
- Support for .NET 8, formerly called .NET Core. Note that static methods in interfaces were introduced in .NET 7 and appear in some system classes such as System.Numerics.IBinaryInteger. These static interface methods are not supported in JNBridge. To use classes with static methods in interfaces first create a .NET wrapper using a non-static interface method.
- Support for Apache Commons BCEL 6.10, which eliminates usage of BCEL 5.1 and a JNBridge customization within it. BCEL 5.1 had a security vulnerability (CVE-2022-42920) which is no longer an issue with BCEL 6.6 and above. See the Users' Guide for details on updating usage which formerly referred to bcel-5.1-jnbridge.jar.

NEW IN VERSION 11.0

- Support for .NET 5, the Microsoft release which removed the use of the "Core" terminology.
- Fixed an issue in the Visual Studio plugin that caused certain items in the plugin's menus to not be properly enabled when running under later releases of Visual Studio 2019.
- Fixed problem in .NET Core projects where .NET-side CPU usage was unacceptably high when .NET side is configured programmatically.
- Updated documentation to put specific upper bound on versions of Java, .NET, and operating systems supported (i.e., removed "or later" phraseology).
- Fixed a problem with the installer where shortcuts were incorrect for non-default installation directories.

NEW IN VERSION 10.1.1

- .NET Core assemblies now are version 10.11.
- Fixed the problem in 10.1 with the use of shared memory in 32-bit Java-to-.NET Core projects. Previously, this would cause an application crash. It now works.
- Fixed an issue with NewJavaControl where it did not function properly when embedded in a tabbed Windows Forms panel.
- Removed the convenience versions of .NET Core DLLs, and now require that they be downloaded as NuGet packages.

NEW IN VERSION 10.1

- 10.1 is a security release. Most new features and changes are security-oriented.
- Removed HTTP/SOAP communications. Now, only TCP/binary and shared memory are offered.

- SSL communications now includes client authentication as well as server authentication. SSL communications is *not* turned on by default; it must be explicitly turned on.
- Whitelisting of classes is now available when using TCP/binary communications, and is used by default. See “Class whitelisting” in the *Users' Guide*.
- More informative exception in .NET-to-Java projects when a callback method is an explicit interface implementation.
- Fixed a problem with dynamic proxy generation inside callback methods, in .NET-to-Java projects.
- ***Known issue:*** Use of shared memory in 32-bit Java-to-.NET Core projects leads to an application crash. We believe this is a problem with the latest preview of .NET Core 3.0, and will be revisiting the issue when a new preview or the GA of .NET Core 3.0 is released.

NEW IN VERSION 10.0.1

- Implemented whitelisting and host IP binding on the .NET side for Java-to-.NET projects.
- Added /ll (license location) option used to specify license file location when using command-line proxy tool for .NET Core.
- When using .NET Core, created more informative error messages in Java-to-.NET shared memory projects when dotNetSide.coreClrPath, dotNetSide.appBase, dotNetSide.javaEntry, and dotNetSide.assemblyList properties are incorrect.
- Updated documentation, tools, and installed shortcuts to indicate that .NET Framework 4.8 is also supported.
- Updated documentation and tools to remove all mention of JAXP. Since we only support Java 5 and later, separate JAXP packages are no longer necessary for parsing SOAP requests.
- Fixed an issue in the whitelisting feature in .NET-to-Java projects when IPv6 is being used.

NEW IN VERSION 10.0

- Support for .NET Core 3.0 (on Windows and Linux).
- Support for Visual Studio 2019.
- Added whitelist feature in .NET-to-Java projects to allow specification of hosts from which .NET-side requests to the Java side will be accepted.

NEW IN VERSION 9.0.1

- Support for Java 11.
- Fixed a problem in .NET-to-Java projects where addressing a dynamically-generated proxy through an interface could lead to an erroneous exception.
- Fixed a problem in projects where cross-platform overrides are used and there are multiple Java sides or multiple .NET sides. Previously, if cross-platform overrides were used, and the application had multiple Java or .NET sides, an exception might be thrown.
- Fixed a problem in Java-to-.NET projects where an incorrect unsigned short value could be returned.
- Fixed a problem in Java-to-.NET projects where passing a null value to a proxied constructor or indexer could cause an AmbiguousMatchException to be thrown.

- Fixed a problem in Java-to-.NET projects when proxying an underlying .NET method with a parameter with no name. This can lead to a Java proxy that might fail verification on some JREs.
- Fixed a problem in Java-to-.NET projects where use of an anonymous Java method in place of a proxied delegate could cause an exception to be thrown.
- Fixed a problem where an embedded Java GUI component could pop out of the containing .NET Windows Form under certain conditions.

NEW IN VERSION 9.0

- Support for Java 10.
- Added support for multiple proxy DLLs in .NET-to-Java projects. See the *Users' Guide* for more information.
- Optimizations to improve the CPU usage percentage when generating proxies or using JNBridgePro in .NET-to-Java projects.
- Added a more accurate and informative error message in the proxy generation tool when attempting to load a Java class whose binary is targeted to a JVM whose version is later than the current JVM.
- Fixed problem where the GUI version of the proxy tool would proxy protected members in Java-to-.NET projects, but the command-line version of the proxy tool would not.
- Fixed a problem in Java-to-.NET classes where ValueTypes (structs) could not be passed as value proxies because they don't have explicit default constructors.
- Fixed a problem in Java-to-.NET projects where proxying methods that were explicit interface implementations could lead to a proxy class that failed Java bytecode verification. See the section "Explicit interface implementations" in the *Users' Guide* for more information.
- Fixed a problem in Java-to-.NET classes where passing an ArrayList object proxied as a mapped collection as a parameter could cause an exception to be incorrectly thrown.
- Fixed a problem sometimes encountered when building proxies using the Visual Studio plugin, where an "illegal character" error might be encountered.
- Fixed a problem in .NET-to-Java projects using HTTP/SOAP communications where returning a Java enum could cause an exception if the enum's toString() method was overridden.
- Fixed a problem in .NET-to-Java projects where multiple .NET sides are working with a common Java side, and the .NET sides shared a common class that overrode a method in the Java side. Previously, that could cause an exception to be thrown.

NEW IN VERSION 8.2.1

- Fixed a problem in proxy generation for Java-to-.NET projects where loading an assembly could lead to an erroneous error message saying that the assembly being loaded was not in the assembly list when in fact it was.

NEW IN VERSION 8.2

- Support for Java 9.
- Fixed a type inferencing issue in .NET-to-Java projects where a complex nested array passed from the .NET side might contain a different type when it arrives on the Java side, and similarly when a complex nested array is returned from the Java side back to the .NET side.

- Fixed problem in Visual Studio plugin where a proxy build would fail if the JNBridgePro editor window was not open.
- Fixed a problem in .NET-to-Java projects where a string returned through a method returning a java.lang.Object would not be wrapped in a JavaString (and consequently would throw an exception) if the method was in a dynamically-generated class.

NEW IN VERSION 8.1.1

- Fixed a problem with method overrides where an exception could be thrown because an object overriding a method on the other platform could be prematurely garbage collected.
- Updated the Eclipse plugin documentation to discuss how to install the plugin when plugins are pooled between multiple Eclipse installations.

NEW IN VERSION 8.1

- Visual Studio plugin support for Visual Studio 2017.
- Fixed a problem in Java-to-.NET classes where proxying a class that implements an interface with a method with a covariant return type might lead to a proxy that fails verification in the JVM.

NEW IN VERSION 8.0

- Cross-platform overrides are now possible in both .NET-to-Java and Java-to-.NET classes. Now, when a proxy class is subclassed and one of its methods is overridden, any calls to the method on the underlying object will be redirected to the overriding method in the subclass. In addition, in Java-to-.NET projects, properties and event accessors, in addition to methods, can be overridden. See the *Users' Guide* for more information.
- .NET classes can now inherit from proxies of Java abstract classes. Previously, when such .NET classes were instantiated, a Java-side InstantiationException would be thrown. This no longer occurs.
- Java classes can now inherit from proxies of .NET abstract classes. Previously, when such Java classes were instantiated, a .NET-side MemberAccessException would be thrown. This no longer occurs.
- There is an "experimental" version of the Java-in-.NET embedding feature, named *NewJavaControl*, that has been reimplemented to work with Java 7 and later. There are a few caveats:
    - Only embedding of Java controls inside Windows Forms is currently supported. Embedding Java inside Windows Presentation Foundation (WPF) will be supported in the next release.
    - There may be some unresolved issues related to z-order (that is, in certain situations, the Java control might appear in front of an element that it should not appear in front of, or behind an element that it should not be behind). Please report such issues if you encounter them.
    - The original version of the Java-in-.NET embedding feature, still named *JavaControl*, is still provided. Please see the *Users' Guide* for more information.
- Modified icons and controls in the GUI-based proxy generation tool and plug-ins, to provide a flatter, more "modern" look.
- When calling multiple .NET sides from Java, added the ability to dispose of all proxies and underlying objects on a single .NET side.

- Exe and Dll files now signed with an SHA-256 certificate, since SHA-1 certificates have been shown to be insecure and have been deprecated.
- Progress bar in proxy tool is now in circular "pie chart" style. Progress bar now works during Build operation. Previously it only worked when loading classes or jar files, or during the Add+ operation.
- In the GUI-based proxy generation tool and plug-ins, placing a check mark next to a package name in the Environment or Exposed Proxies pane through keyboard action (for example, hitting the space bar when a package name has been selected), will now cause a check mark to be placed next to all classes in the package. Previously, this only happened when the check mark was placed by clicking on the check box with the mouse.
- Improved error reporting in Java-to-.NET shared memory projects where an exception is thrown on the .NET side during the request serialization/deserialization process. Now, a more informative exception is thrown, and further information is recorded in the event log.
- Fixed an issue where embedded Windows Forms and WPF components would not properly resize when the surrounding Java control is resized. As part of the fix to this issue, we have added methods DotNetControl.setEmbeddedSize() and DotNetSWTControl.setEmbeddedSide() to be called from the surrounding control's resize listener. See the *Users' Guide* for more information.
- A more informative error message is output in the command-line proxy generation tool, when the /cp, /jp, or /bp options have been omitted and there's no CLASSPATH environment variable.
- Updated Users' Guide section on embedding multi-threaded .NET controls in Java to add code to wait until the control can quit before the surrounding Java application can terminate. In some cases, if that code is not there, the Java application can hang upon termination.
- In registration tool, added the ability to display the installation's registration key in a command-line console. This is particularly useful for getting the registration key when running on Mono, but it will also work on regular .NET installations. See the *Users' Guide* for more information.
- In registration tool, added the ability to display usage instructions when running the application from a command line, by entering "RegistrationTool.exe /help". Previously, this worked on Mono installations, now it works on regular .NET installations, too.

NEW IN VERSION 7.3.1

- Improved appearance of progress bar in proxy tool. Now, it isn't visible until it's actually used, and it uses the 'continuous' style, rather than the block style. Its size has also been adjusted.
- In VS 2013 and 2015 plug-ins, fixed rendering of Add/Add+/Delete buttons.
- Now, when a licensing or configuration issue causes a TypeInitializationException in an application that uses JNBridgePro, the application attempts to record the root cause of the error in the event log. Launch the Event Viewer and look in the Application log for an entry with the source "Application."
- Fixed problem in some .NET-to-Java projects where Java classes that implement a nested private interface could not be proxied.

NEW IN VERSION 7.3

- JNBridgePro Visual Studio plug-in now supports Visual Studio 2015.
- JNBridgePro now supports Windows 10.

- Licensing dlls are now named jnbauth_x86.dll, jnbauth_x64.dll, jnbauth_x86.so, and jnbauth_x64.dll. Previously, they were named rlm932 instead of jnbauth. The reason for the change was to more clearly express the purpose of the dlls, and to make them version-independent.
- In Java-to-.NET projects, proxy method, property getter/setter, and constructor parameters now have metadata giving them the same names as the underlying .NET methods, property getter/setters, and constructors, so that the parameter names appear when the proxies are used in Java IDEs, including Eclipse.
- In .NET-to-Java projects, proxy method and constructor parameters now have metadata giving them the same names as the underlying Java methods and constructors, so that the parameter names appear in IntelliSense. (Note: this only occurs when the Java classes were compiled with Java 8 or later, using the –parameters option, and the Java side used by the proxy generation tool is also Java 8 or later.)
- Licensing mechanism will now look for the license file in the application base folder after looking in the startup folder. This is particularly significant for Java-to-.NET applications using shared memory.
- Fixed a problem where a property passed through a jvmOptions element might not be seen.
- Double-clicking on a .jnb file now launches the 4.0-targeted proxy generation tool. Previously, the 2.0-targeted proxy generation tool was launched.
- Fixed a problem that could cause class name completion not to work in proxy generation in .NET-to-Java projects when certain recent versions of Java are being used.
- Fixed a problem in Java-to-.NET projects where use of a method with a covariant return type (that is, a method that returns a subclass of the return value returned by a superclass or an interface that the original class implements) could lead to a ClassCastException.

NEW IN VERSION 7.2.1

- Fixed issues with focus-based events (mouse wheel and keyboard events) when embedding Java controls inside .NET applications and Java 7 or 8 is being used. Now, mouse wheel events will be passed to simple (non-compound) Java controls. There are still issues with embedding Java in .NET and Java 7 or 8 are being used: see the jnbridge.com knowledge base for more information. When embedding Java controls in .NET, we currently recommend using Java 6. We will continue to work to eliminate this issue.
- Fixed a problem where, when passing or returning a value object whose class hierarchy includes multiple value proxies, not all fields are passed.
- Fixed a problem in Java-to-.NET projects where use of a non-public or anonymous event handler on the Java side could cause an IllegalAccessException to be thrown.
- Fixed problem when embedding .NET controls in Java applications, where a control might fail to redraw when its position in the Java UI hierarchy has been changed programmatically.
- Fixed a problem in .NET-to-Java projects where an interface hierarchy contains methods with covariant return types. Previously, resulting proxy dll might not verify, since not all such methods were proxied and a class might be flagged as not implementing all required interface methods. Now, all such methods are proxied.

NEW IN VERSION 7.2

- Support for Java 8, particularly for static and default methods in interfaces. See the *Users' Guide* for details.
- Substantial performance improvement in .NET-to-Java projects when many proxies have been instantiated.
- 32-bit proxy generation tool in 7.1 actually ran as "Any CPU." Now runs as 32-bit.
- Fixed a problem where a ThreadAbortedException could be thrown when exiting a Java application containing embedded .NET UI components.
- Fixed a problem in .NET-to-Java applications where public/protected fields in superclasses of by-value proxies were not properly assigned when the fields were returned from the Java side.

NEW IN VERSION 7.1

- JNBridgePro Visual Studio plug-in now supports Visual Studio 2013.
- Added ability to specify 32-bit and 64-bit jvm.dll in .NET-to-Java application configuration files, in order to support shared memory communications in Any CPU applications.
- Performance improvement in Java-to-.NET projects when creating boxed primitive values.
- Fixed a problem where a method containing an out or ref parameter that was of a generic type would not be proxied.
- Fixed a problem where an exception could be erroneously thrown in Java-to-.NET projects when calling a property setter whose type is generic.
- Fixed a problem where an exception could be erroneously thrown in Java-to-.NET projects when calling a proxied method or constructor with vararg arguments that are generics.
- Fixed a problem where an exception could be erroneously thrown in .NET-to-Java projects when using JavaBean-style value proxies that inherit from other JavaBean-style value proxies.
- Fixed a problem where there could be an error in a Java-to-.NET proxy where the underlying class implements a bound generic interface.
- Fixed a situation where an exception could be erroneously thrown when using shared memory and DNS is incorrectly configured on the machine.
- Fixed a situation where leaving bcel-5.1-jnbridge.jar out of the classpath in a .NET-to-Java project could cause an exception to be thrown. bcel-5.1-jnbridge.jar should not be required in .NET-to-Java projects. Now, the exception is no longer thrown.
- In Java-to-.NET projects, if a proxied method has a name and signature that are the same as one of the six final methods in java.lang.Object, the proxied methods are renamed so that "__JNBridgeProxy" is appended to the end of the name. That way, they will not override the final method of Object, and the proxy will not fail Java verification. See the *Users' Guide* for more details.
- Adding a more informative exception message when using a JavaBean-style value proxy that is missing a getter method for one of its public properties.
- Fixed a problem with very large messages compressed and sent over the tcp/binary channel. In some cases, the entire message might not be sent.

NEW IN VERSION 7.0.1

- Fixed a problem that could cause an exception to be thrown when running a project that embeds a Java UI element in a .NET Windows Forms or WPF application, or when running a bidirectional project using shared memory.

NEW IN VERSION 7.0

- JNBridgePro deployments will now work when the .NET side runs on Mono. Currently, we support Mono on Windows and on Linux on x86 and x64 platforms. See the *Users' Guide* for instructions and limitations.
- Added ability to use shared memory in .NET-to-Java projects that were built as "Any CPU." Previously, one had to choose to build projects as x86 or x64, then choose the appropriate 32-bit or 64-bit jnbsharedmem.dll.
- Added ability to create Java-to-.NET projects that use shared memory and can be used with either 32-bit or 64-bit JREs without any change.
- The 32-bit and 64-bit versions of JNBridgePro have now been combined into a single installer that installs both 32-bit and 64-bit components.
- Fixed a problem in .NET-to-Java projects where an exception could be thrown when a proxy is dynamically generated and it inherits from a class that contains a protected abstract member.

NEW IN VERSION 6.1

- Plug-in support for Visual Studio 2012.
- In Java-to-.NET projects, calls to proxied methods with ref or out parameters is now supported. Please see the *User's Guide* for more information.
- Added a license query API. Please see the *User's Guide* for more information.
- Access to protected members is now supported. Please see the *User's Guide* for more information.
- Fixed a problem where an exception could be thrown in .NET-to-Java projects where one attempts to pass a value whose type is an array of a Java nested type.

NEW IN VERSION 6.0.2

- Fixed a memory leak in Java-to-.NET projects.
- Fixed a problem in Java-to-.NET projects where an exception would be thrown if a proxied System.IntPtr or System.UIntPtr object were passed or returned.
- Improved error reporting and recovery in Java-to-.NET projects when an assembly cannot be loaded. Previously, this could cause the .NET side to fail with no explanation. Now, the problematic assembly will be skipped, and the problem will be logged in the event log.

NEW IN VERSION 6.0.1

- Fixed a problem in .NET-to-Java projects where the proxy generator could terminate when a very large jar file was added to the proxy generator's classpath.
- Fixed an error where if a user has x64 copies of two different versions of JNBridgePro installed, and the older version has an invalid license, then invoking the older version's proxy tool will cause the

registration tool for the newer version to launch. The appropriate version of the registration tool will now launch.

- Fixed a problem where class name recommendation/completion would not work when the proxy generator was configured to use shared memory.
- Fixed an error where a registration key would sometimes not be displayed in the registration tool when using off-line activation.
- Fixed issues with on-line activation when a proxy server is being used.

NEW IN VERSION 6.0

- Added a new licensing mechanism, including support for licensing in the cloud, in virtual machine images, and in terminal servers. See the *Licensing and Installation Guide* for more details.
- As part of the licensing mechanism update, we have eliminated the distinctions between the SE and EE editions of JNBridgePro.
- Added ability to drag-and-drop files and folders into the Edit Classpath/Edit Assembly List form in the GUI-based proxy generation tool.
- Added class name suggestion to the Add classes from classpath/assembly list form in the GUI-based proxy generation tool. (Not available in the .NET 1.x-targeted version.)
- Added the ability to use a wildcard '*' to indicate that all classes in a namespace/package should be loaded (e.g., java.util.* or System.Collections.*).
- Added support for the Eclipse plug-in in both 32-bit and 64-bit Eclipse 3.6 ("Helios").
- Changes made to installer to properly install plug-in into Visual Studio 2010 service pack 1.
- In .NET-to-Java projects, when the name of an automatically generated interface helper class conflicts with that of another class being proxied (for example, when you're proxying an interface X and also proxying a class XConstants), the proxy generator will now note the issue and report on all such issues at the end, while still generating the proxies that it could. Previously, the proxy generator would fail with an exception at the first point where it encountered such a name conflict.
- When generating proxies, and name conflicts or missing classes need to be reported, the displayed dialog box now also contains a link to relevant documentation informing the user as to how to correct the problem.
- Removed support for generating proxies for .NET-to-Java projects that are compatible with Microsoft Visual J#.
- Fixed an issue involving the ability of the 4.0-targeted proxy generator to read .jnb files written by the 2.0-targeted version, and vice versa.

NEW IN VERSION 5.1

- Visual Studio plug-in that is compatible with Visual Studio 2010, as well as with Visual Studio 2005 and 2008.
- JNBridgePro tools and components targeted toward .NET Framework 4.0, in addition to tools and components targeted toward .NET Framework 2.0/3.0/3.5.
- The Visual Studio 2010 plug-in supports multi-targeting, so that the user can choose whether generate proxies targeted toward .NET Framework 2.0/3.0/3.5, or toward .NET Framework 4.0.

- The standard JNBridgePro installer no longer contains tools and components targeted toward .NET Framework 1.0 and 1.1; these are now “by request” items. For more information, please contact sales@jnbridge.com.
- In Java-to-.NET projects, the generated proxies are now by default targeted to Java SE 5.0. They were previously targeted by default to J2SE 1.3, and the user had to explicitly turn on the option to target them to 5.0. Now, the user has to explicitly turn the option off.
- In Java-to-.NET projects, the proxy generator will no longer proxy members that return a nullable type (e.g., bool?) or that take a nullable type as a parameter. Previously, members that used nullable types were proxied, but if the members were accessed, an exception would be thrown.
- Modified bcel-5.1-jnbridge.jar so that it will not conflict with any existing bcel.jar that is being used in a Java-to-.NET project (for example, by a hosting Java EE application server). The new file is also named bcel-5.1-jnbridge.jar, and must be used in Java-to-.NET projects from JNBridgePro v5.1 onward.
- Fixes issues in .NET-to-Java projects when proxying and accessing Java classes that include “bridge methods” automatically generated by the Java compiler. For more information, please see the *Users' Guide* (“Bridge methods and ambiguous calls”).
- In Java-to-.NET projects, can now proxy .NET classes that contain double underscores. Previously, an exception might be thrown, or a missing class reported.
- Fixed an erroneous exception that would sometimes be thrown in both .NET-to-Java and Java-to-.NET projects when proxying or accessing classes whose names contain a double underscore (“__”).
- Performance improvements in .NET-to-Java projects.
- Fixed an issue in bidirectional projects using shared memory (where execution starts on the Java side), where proxy types were sometimes not loaded and an exception was thrown.
- Fixed an issue in mapping of dates between Java and .NET where the mapping could be off by one hour for a brief window around the time of the Daylight Savings Time changeover.

NEW IN VERSION 5.0

- Added cross-platform transaction interoperability. See the *Users' Guide* for more information.
- Proxy classes that were previously called “thread-true” are now called “transaction-enabled.”
- Added support for thread-true proxies in Java-to-.NET projects. See the *Users' Guide* for more information.
- In .NET-to-Java projects, added the ability to turn off the mapping of Java enums to .NET enums. When this mapping is turned off, the proxied Java enums are just ordinary proxies, and one can access methods associated with them, which was not possible when they were mapped to .NET enums.
- JNBridgePro will now look for a license file in the folder \Program Files (x86)\JNBridge if it doesn't find it in \Program Files\JNBridge.
- Selected code generation modifications in preparation for .NET 4.0 support.
- Fixed a type inferencing error that could occur in .NET-to-Java projects when arrays are passed as parts of mapped collections.

NEW IN VERSION 4.1.1

- Now certified “Compatible with Windows 7.”
- All registration is now done through the registration tool. When a “register” action is selected in the standalone proxy generator or plug-ins, the registration tool is now launched in its own process. This change was made to increase compatibility with UAC (User Access Control) in Windows Vista and Windows 7.
- In Java-to-.NET projects, wrapper classes (DotNetArrays, DotNetStrings, DotNetEnums) now implement now implement the proxies of the interfaces that the underlying .NET classes implement. For example, DotNetArray implements IList`1, since the underlying .NET arrays implement IList`1.
- In Java-to-.NET projects, JNBridgePro now allows Java code to interoperate with .NET code generated by the C# yield statement. Previously, interoperating with such code could cause an exception to be thrown.
- In Java-to-.NET projects, the method DotNetString.stringValue() is now declared to return an object of type java.lang.String. Previously, it was declared to return java.lang.Object.
- Fixed an issue that could cause an exception to be thrown during proxy generation when a Java-to-.NET project is being developed, shared memory is being used, and the proxy jar file is re-generated with changes.
- Fixed a memory leak when using asynchronous callbacks.
- Removed an erroneous exception that could sometimes be thrown by the licensing mechanism when using in an ASP.NET 1.1 Web application.
- Fixed a type inferencing error in Java-to-.NET projects when an object array is returned containing proxies and strings.

NEW IN VERSION 4.1

- In GUI-based proxy generation tool and Visual Studio plug-in, added the ability to generate a command-line script to generate the proxies. This can be useful in automated build processes. For more information, please see the *JNBridgePro Users’ Guide*.
- Added the ability to embed a Windows Presentation Foundation (WPF) control inside a Java UI (AWT, SWT, or Swing) application. For more information, please see the *JNBridgePro Users’ Guide* (“Embedding .NET GUI elements inside Java GUI applications”).
- Added the ability to embed a Java UI component in a Windows Presentation Foundation (WPF) program. For more information, please see the *JNBridgePro Users’ Guide* (“Embedding Java GUI elements inside .NET GUI applications”).
- Added “Purchase licenses” and “Request license key” buttons to the registration tool and to the GUI-based proxy generation tool and the Visual Studio plug-in.
- Updated examples for several Java EE application servers (JBoss, WebLogic, WebSphere). Added an example for GlassFish. Removed examples for Borland Enterprise Server, JRun, Oracle AS, Sun ONE Application Server, Sybase EA Server. The removed examples can still be found on the JNBridge Web site.
- Fixed an issue when embedding .NET controls in Java programs, where an embedded control could sometimes throw an exception due to the fact that it required the STA threading model, even when the STA model was specified in the properties.

- Fixed a problem that could cause an erroneous exception to be thrown when embedding a multi-threaded .NET control in an SWT application and calling Application.DoEvents() from the Java side.
- Created a much more informative exception in Java-to-.NET projects using tcp/binary or HTTP/SOAP, when the .NET side hasn't been started.
- Created a much more informative exception in Java-to-.NET projects when the .NET side is configured to use HTTP/SOAP and the Java side is configured to use tcp/binary.
- Created a much more informative exception in Java-to-.NET projects when the DotNetSide.init() isn't called before the first proxy call, or when the dotNetSide.serverType property isn't set.
- Created a much more informative exception in Java-to-.NET projects when shared memory is being used, and the .NET-side fails due to a licensing or configuration problem.

NEW IN VERSION 4.0.2

- Generated .NET-side proxy DLL files now contain an file version resource that matches the version value specified through the proxy generator's strong naming mechanism (if any). This means that if a version is specified through strong naming, the version will be also be displayed next to the DLL's icon in Windows Explorer, and the file's properties window in Windows Explorer will have a Version tab. If there is no strong naming version, the displayed file version will be 0.0.0.0.
- Added the ability, in Java-to-.NET applications using shared memory, to specify the apartment threading model of the .NET side. Please see "Specifying the .NET-side apartment threading model when using shared-memory communications" in the *JNBridgePro Users' Guide*.
- More user-friendly error handling in the proxy generation tool when a build is done using a .jnb file that references a class that can no longer be found.
- More user-friendly error handling in the proxy generator when loading a Java class causes a Java-side IncompatibleClassChangeError to be thrown.
- More informative error reporting in Java-to-.NET projects when the proxy generation tool cannot load an assembly.
- Now throws a JNBSharedMem.SharedMemoryException when the Java side fails when using shared memory. This is more informative than previous versions, where an AccessViolationException (when using the 2.0-targeted version) or a NullReferenceException(when using the 1.x-targeted version) was thrown. The SharedMemoryException contains a more informative message. To support compatibility with applications using previous versions of JNBridgePro, SharedMemoryException is derived from AccessViolationException (in the 2.0-targeted version) or NullReferenceException (when using the 1.x-targeted version).
- Updated Users' Guide to state that file paths can be directly entered into the New Classpath/Assembly List Element dialog, including UNC paths to shared network folders.
- In Java-to-.NET projects, can now specify the .NET-side server to use an IPv6 listener. Similarly, in .NET-to-Java projects, can now specify the Java-side server is an IPv6 listener.
- Fixed a race condition that could cause an erroneous exception to be thrown in certain multithreaded situations in .NET-to-Java projects.
- Fixed a possible exception when a dynamic proxy is created and returned, and the proxy implements a nested interface.

- Fixed an erroneous exception when a JavaString/JavaEnum/JavaArray coercion is performed on a null value.
- Fixed a problem introduced in 4.0, where .jnb files generated by the 2.0-targeted proxy generation tool could not be read by the 1.x-targeted proxy generation tool.
- Fixed an issue when Java UI elements are embedded in Windows Forms, where resize operations on the embedded Java element lagged one event behind the surrounding Windows Form. While this was most noticeable in maximize/minimize operations, it also occurred when elements were resized through drag operations. The resizing of the embedded elements now keeps up with the surrounding form.
- Fixed a type inferencing error in .NET-to-Java projects that could sometimes occur when an empty array is returned.
- Fixed an erroneous exception that could be thrown on Vista and Windows 2008 Server when tcp/binary is used in a .NET-to-Java project and IPv6 is being used.

NEW IN VERSION 4.0.1

- Fixed an issue where Eclipse plug-in could fail to load when Eclipse is running on JDK/JRE 1.5.x.
- Fixed an issue during proxy generation, where a class failing the Java verification process could cause the proxy generation tool to prematurely terminate.
- Fixed an issue where a deployment license file might not be found in certain circumstances.
- When using the Eclipse plug-in, modifying a proxy's reference/value designation will now cause the proxies to be re-generated. Previously, this did not cause the proxy jar file to be re-generated.
- Fixed a race condition that could appear in Java-to-.NET projects when a proxy class could be dynamically generated at run time on multiple threads.
- Fixed an issue that could prevent the Visual Studio plug-in from being properly installed if the Windows system folder was something other than C:\Windows\System32.
- Fixed an issue in Java-to-.NET projects where a proxy class dynamically generated at runtime, whose underlying class inherits from a generic class, might not inherit from the generic class's proxy, possibly leading to an erroneous class cast exception.

NEW IN VERSION 4.0

- Added proxy generator plug-ins for Visual Studio (supporting VS 2005 and VS 2008) and Eclipse (supporting Eclipse 3.2 and 3.3). See the *Users' Guide* for more information.
- JNBridgePro now has 64-bit support. See the *Users' Guide* for more information.
- The tcp/binary communications mechanism now compresses messages over a certain size before sending them between the .NET and Java sides. This should improve performance when large messages (particularly, calls and returns containing large strings or arrays) are passed. For more information, see the *Users' Guide*.
- Added the ability to specify the Java-side memory allocation used during proxy generation. See the *Users' Guide* for details.
- Added the ability to specify the .NET side's application base folder as part of a Java-to-.NET shared-memory configuration. See the *Users' Guide* for details.

- Added Callbacks.releaseCallback(), which can be used in Java-to-.NET projects to release the resources used by callbacks when the callbacks are no longer used. See the section "Callbacks" in the Java-to-.NET section of the *Users' Guide* for more information.
- In tcp/binary projects, larger messages are now compressed to improve performance. See the *Users' Guide* for more details.
- Improved cross-platform type inferencing, so that complex arrays of objects are more faithfully re-created when copied cross-platform (that is, their type hierarchy is reproduced).
- In Java-to-.NET projects, when .NET Hashtables are proxied as mapped collections, values that are null are now translated as NullValue objects in the mapped Java Hashtable. This is because Java Hashtables do not allow null values, while .NET Hashtables do. Similarly, when Java Hashtables are mapped back to .NET Hashtables, NullValue objects will be translated to null values.
- Modified behavior when proxying Java enum classes that have constant-specific class bodies that attach behaviors to the enum constants. Previously, the presence of such class bodies could cause exceptions to be thrown at proxy generation time. Now, these class bodies are ignored and not proxied. See the *Users' Guide* for more details.
- In Java-to-.NET projects, an AppDomain containing the JNBridgePro .NET side can now be unloaded after calling DotNetSide.stopDotNetSide(). Previously, a CannotUnloadAppDomainException was thrown when an AppDomain containing the .NET side was unloaded.
- Fixed a proxy generation error that could sometimes occur when attempting to proxy a Java enum that has been obfuscated.
- All saved project files are now guaranteed to have the extension .jnb. Previously, if a project was saved away with a name *x.y*, where *y* was an alphanumeric string, sometimes the saved file would erroneously have the name *x* and the file extension *y*. Now, the file will be saved as *x.y*.jnb.
- Fixed an issue where an erroneous exception could be thrown when proxy objects are used as keys in .NET Dictionaries.
- Fixed an error when returning a Java BigDecimal value as a mapped proxy, when using JDK 1.5 or later. Returning mapped BigDecimals now works with all versions of Java.
- Fixed a problem in .NET-to-Java projects, where final methods in non-public abstract classes might not be visible to proxies of classes further down in the inheritance chain.
- Fixed a problem in .NET-to-Java projects, where proxied enums might be passed as null values in certain multi-threaded situations.
- Fixed an issue that would sometimes cause a callback in .NET-to-Java projects to fail in multi-threading situations.
- Fixed a problem that could cause an exception to be thrown in .NET-to-Java projects with callbacks and multiple Java sides.

NEW IN VERSION 3.2.2

- Corrected an issue in Java-to-.NET projects where a proxy could have duplicate interfaces of the same name, leading to an exception being thrown.
- Corrected an issue in Java-to-.NET projects where the Java side could fail during proxy generation when proxying an interface that itself had nested interfaces or events.

- Corrected an issue in management of callbacks in Java-to-.NET projects which could cause a callback to hang, or to fail with an incorrect exception.
- Corrected an issue in Java-to-.NET projects where an exception could be thrown when returning an array of .NET objects represented by by-value proxies.
- Fixed a potential memory leak that could occur in .NET-to-Java projects when a large number of callback objects are created.
- Corrected an issue that occurred when an object that had been disposed was revived (due to the fact that the underlying Java object was again returned from a proxy call). Previously, if such a revived proxy was again disposed, an exception was erroneously thrown. Now, the proxy can be disposed again.
- The proxy generator now properly handles bridge methods (methods sometimes generated by compilers from JDK 1.5 and later to support generic classes and inheritance). Previously, the proxy generator would sometimes generate two methods in a class with the same signature but different return types. This would lead to proxy assemblies that would not verify, and sometimes to exceptions sometimes being thrown when one of these methods was called.
- In Java-to-.NET projects, JNBridgePro now associates a dedicated timer with each Java-side socket that is created. Previously, a new timer was created each time a socket was put back into the available socket pool, so that if it was unused for a specified period of time, the timer would fire and cause the socket would be closed. Since every timer has an associated thread, this caused a large number of threads to be created. Although each thread was properly terminated and disposed, and most of the memory recovered, a small amount of unrecoverable memory was allocated when each timer thread was created and this led to a small memory leak. Now, since each socket has a single dedicated timer, the number of timer threads remains stable and this memory leak no longer occurs.
- When the <dotNetToJavaConfig> tag is missing the “scheme” element, an informative exception is now thrown. Previously, an exception was thrown, but the cause of the exception was not clear to the user.

NEW IN VERSION 3.2.1

- Implemented a substantial performance improvement in Java-to-.NET projects using tcp/binary communications between machines.
- In Java-to-.NET projects, it is now possible to pass arguments that represent boxed copies of unsigned integer, unsigned short, unsigned long, and (unsigned) byte. For more information, please see the section on “Boxed primitive types” in the Users’ Guide.
- In Java-to-.NET projects, when value types without constructors are proxied, the proxies now have a default constructor. Previously, such proxies had no constructors and couldn’t be instantiated.
- In .NET-to-Java projects, proxies of Java inner exceptions now support the Message, JavaExceptionMessage, StackTrace, and JavaStackTrace properties.
- In the GUI-based proxy generation tool, the contents of the up/down dropdown box in the Find dialog can no longer be edited.
- Eliminated a situation in debugging .NET-to-Java projects with callbacks, where an erroneous InterruptedException may be thrown during the debugging session.
- Corrected an issue where a generic classes were sometimes not detected as supporting classes of other classes.

NEW IN VERSION 3.2

- JNBridgePro 3.2 has been *Certified for Windows Vista*, and exhibits the *Certified for Windows Vista* logo.
- The JNBridgePro Java side now trims white space from names and values of Java-side properties, in order to avoid potential user errors where white space is accidentally added to the beginning or end of the names or values. Since there should be no white space at the beginning or end of the names or values in Java-side properties, this is a safe change.
- Launch JNBProxy dialog box will now default to "Open recent project" option if there are recent projects. If there are no recent projects, it will default to "New .NET-to-Java project."
- If a recent project has been removed, it will no longer appear in the recent project list of the Launch JNBProxy dialog box.
- In the GUI-based proxy generator, dialog boxes are now centered on their parent boxes' locations. Previously, Windows assigned the locations of the dialog boxes.
- Modified Java-to-.NET proxy generation so that proxies for nested delegates do not cause compilation errors when some Java compilers are used.
- In certain circumstances when using shared memory in previous versions, proxy calls by more than one thread could cause an access violation exception. This exception no longer occurs.

NEW IN VERSION 3.1.3

- In Java-to-.NET projects, parameters to callbacks (delegates/events) can now be value proxies. Previously, they could only be reference proxies.
- A number of changes have been made to make JNBridgePro, and particularly its installer, more compatible with Windows Vista. In particular, the installer now works with Windows Vista's User Access Control (UAC) mechanism. Also, all the executables are now signed with an Authenticode certificate, in addition to having .NET strong names.
- Fixed a problem that could cause an exception to be thrown when initializing multithreaded Java-to-.NET projects.
- Fixed a problem in Java-to-.NET projects where a dynamically generated proxy object might not implement all the interfaces that the underlying .NET object implements.
- Fixed a problem that could cause a .NET-to-Java application to lock up when certain timing conditions occur relating to callbacks.

NEW IN VERSION 3.1.2

- Deployment license files can now be deployed in the application's current working directory (generally, the folder in which the application's .exe file resides).
- It is no longer necessary to have run the registration tool under administrative or power user privileges before running a JNBridgePro-enabled application for the first time (or run the application itself under such privileges the first time) when using a deployment license file.
- Added support in Java-to-.NET projects for accessing .NET-side methods and properties that are explicit interface implementations (that is, whose declaration includes the name of the interface whose method or property they are implementing). Previously, accessing such methods and properties could cause an exception to be thrown. Note that if the class has more than one member with the same name of the same name that are explicit interface implementations of different

interfaces, only one will be proxied. The proxy generator will indicate which member has been proxied, and which members have not.

- Added Callbacks.releaseCallback(), which can be used in .NET-to-Java projects to release the resources used by callbacks when the callbacks are no longer used. See the section "Callbacks" in the .NET-to-Java section of the *Users' Guide* for more information.
- JNBridgePro executables now contain an embedded manifest resource containing User Account Control information, which is used by Windows Vista to enforce security policy.
- JNBridgePro executables and dll files are now signed with the JNBridge Authenticode certificate.
- Fixed a Windows Vista-related problem that caused the .NET DNS resolution API to break, and prevented the tcp/binary communications mechanism from working properly.
- Fixed a problem that could cause the proxy generator to fail when loading a jar file that contains no classes.
- Fixed a problem in Java-to-.NET proxy generation that could cause the proxy of .NET class that inherits from a generic class to have an incorrect superclass.

NEW IN VERSION 3.1.1

- Fixed problem in Java-to-.NET projects using shared memory, where JNBridgePro always attempted to load jawt.dll, even when GUI embedding was not being used. Now, jawt.dll is only loaded when GUI embedding is being used.
- Fixed problem in .NET-to-Java projects, where calling a callback method which has a parameter that is declared to be an array of interfaces could cause an erroneous exception to be thrown.
- Fixed problem where an exception might be thrown when the application using JNBridgePro has already registered the http Remoting channel.
- Fixed problem when passing or returning JavaBean-style value objects that contain static constant fields.

NEW IN VERSION 3.1

- Added the ability to embed Java GUI elements inside .NET GUI applications, and vice versa. See the *Users' Guide* for more information.
- Implemented the ability, in Java-to-.NET projects, for a .NET side to contact multiple Java sides. See the *Users' Guide* for complete information.
- Implemented ability, in Java-to-.NET projects, to query a .NET side to determine whether the .NET side is up and running, using the DotNetSide.isAlive() API. See the *Users' Guide* for complete information.
- In .NET-to-Java projects, methods of value proxies are now annotated with the ObsoleteAttribute, which will cause the compiler to generate warnings that calls to these methods will cause NotImplementedExceptions to be thrown.
- The New Classpath Element dialog box in the GUI-based proxy generation tool now displays the Desktop and My Documents folders.
- Proxy generator dialog boxes that had been resizable are now fixed-size.
- In Java-to-.NET projects, the command-line proxy generator will now delete the proxy jar file before starting up the Java side. This will avoid problems when the proxy jar file is in the Java side's

default classpath, in which case the JVM may fail when the proxies (which have already been loaded by the JVM) are re-generated.

- Fixed problem in Java-to-.NET projects that could cause a class in a dynamically loaded assembly not to be visible from the Java side at run time.
- Fixed problem in Java-to-.NET projects where an exception is thrown when a class is loaded that depends on an assembly that is not in the assembly list or is otherwise unavailable.
- Fixed problem when generating proxies of classes whose names begin with "__" (two underscores). Previously, generating a proxy for such a class could cause an exception to be thrown.
- Fixed a problem that could cause the element class of an array parameter to not be counted as a supporting class during proxy generation. This could sometimes lead to an exception during proxy generation.
- Fixed a problem in Java-to-.NET projects where a <dotNetToJavaConfig> element was still required in an application configuration file, and an erroneous exception was thrown if it was missing. The element is no longer required in Java-to-.NET projects, and the exception is no longer thrown.
- Eliminated an erroneous exception that occurred in Java-to-.NET projects when the .NET code performed a remoting operation using the ipc: protocol.
- Fixed a problem that could lead to an exception when generating a proxy for a generic .NET class that inherits from another generic .NET class.
- Fixed a problem in the command-line proxy generator that could lead to an exception being thrown when generating a proxy for a Java class whose superclass chain contains a nested class.
- Fixed a problem in .NET-to-Java projects where a call to a method could result in an erroneous InvalidAccessException when the method is part of a non-public superclass of a public class that implements a public interface.
- Fixed a problem where calling ChannelServices.UnregisterChannel() (and some other ChannelServices methods) on the JNBridgePro tcp/binary and shared-memory channel objects could cause an exception to be erroneously thrown.
- Fixed a problem in Java-to-.NET projects where an exception could be thrown when dynamically generating a proxy for a .NET class whose members use reference classes for parameters or return values.
- Fixed a problem that could cause an exception to be thrown in Java-to-.NET projects when a proxy class has been loaded by two different classloaders.
- Fixed problem where mapping between .NET decimal and Java BigDecimal would not work in cultures/locales where the decimal separator was not '.'.
- Fixed a problem in proxy generation that, in some situations, might cause the proxy generation tool to erroneously report that a class could not be proxied because it depended on a class that could not be found.
- Fixed a problem that, when instantiating a by-value proxy with a circular dependency, could cause a StackOverflowException.
- Fixed a problem where proxying a .NET class with overloaded indexers could yield a Java-side proxy that, when used, would throw a ClassFormatError.

NEW IN VERSION 3.0.2

- Change to the command-line options for registrationTool.exe. Now, when used in silent mode (that is, with the /n option), one should now also specify the /EE or /SE options depending on whether an EE or SE license will be installed. See the sections "Deploying JNBridgePro as part of another application" in the *Users' Guide* for more information.
- Fixed problem in Java-to-.NET projects where an incorrect proxy exception might be thrown on the Java side when an exception which is a nested class is originally thrown from the .NET side.
- Fixed a problem when dynamically generating proxies for anonymous, non-nested classes such as Enumerations. Previously, an exception could be thrown.

NEW IN VERSION 3.0.1

- Fixed a problem that could cause a Java-to-.NET proxy to be generated incorrectly if a member had a generic parameter or return value.

NEW IN VERSION 3.0

- .NET generic classes and methods are now mapped to Java proxies and can be accessed and instantiated at runtime. Java generics can also be accessed from .NET (but only in their raw form, since this is all that is available after they are compiled). See the *Users' Guide* for more information.
- Secure communications between .NET and Java, using the Secure Socket Library (SSL) is now supported. See the *Users' Guide* for more information.
- Java 5.0 enums are mapped to .NET enums, and .NET enums are mapped to Java enums (only when the proxy generator has been set to generate Java 5.0-targeted proxies).
- Static .NET classes are now mapped to final Java proxy classes without a public constructor.
- Now, when a vararg method is encountered in an underlying class, the method in the corresponding proxy is also vararg.
- The JNBridgePro v3.0 installer installs two versions of JNBridgePro v3.0: a .NET Framework 2.0-targeted version, and a .NET Framework 1.0/1.1-targeted version. See the section "1.x- and 2.0-targeted versions," in the *Users' Guide*.
- Changed the order of parameters in the proxied setter method of an indexer property. For example, if the underlying .NET indexer is of the form

  ```
  object this[object index1, object index2]{ get ; set }
  ```

  then in the Java-side proxy, the corresponding setter method will be of the form

  ```
  void Set_Item(System.Object value, System.Object index1, System.Object index2).
  ```

  In previous versions, the setter method had the order of the parameters reversed:

  ```
  void Set_Item(System.Object index1, System.Object index2, System.Object value).
  ```

  This change was made to accommodate indexers with a variable number of arguments.
- New option has been added to generate J2SDK 5.0-targeted proxies in Java-to-.NET projects.

- The Add Classes from Classpath and Add Classes from Assembly List dialogs now provide two methods of editing the entered class information – via a new “Edit …” button and by double-clicking on a given class.
- The Add Classes from Classpath and Add Classes from Assembly List dialog now allow multiple selection of classes for the purpose of deletion.
- The Add Classes from JAR File and Add Classes from Assembly File now support multiple selection of files.
- The directory navigation pane in the New Classpath Element and New Assembly List Element dialogs now includes all available drives on the computer running JNBProxy. Previously, only fixed drives were listed.
- The directory navigation pane in the New Classpath Element and New Assembly List Element dialogs now supports multiple selection of files and/or folders.
- Added the ability to toggle between SE and EE versions during evaluation period.
- New licensing mechanism won’t allow proxy generation tool to generate proxies for classes in J2EE packages when license is for SE.
- If a proxy assembly was generated with an SE license, an exception will be thrown if it is run with an application that has an EE license.
- Improved tcp/binary communication performance in Java-to-.NET projects where the .NET and Java sides are on different machines.
- Fixed problem where Java-to-.NET project would terminate if there were a delegate or event handler with a non-primitive value type as a parameter or return value.
- Fixed problem in Java-to-.NET projects where an exception was erroneously thrown when a boxed primitive is passed from .NET to a Java callback.
- Fixed problem in .NET-to-Java projects where an exception was erroneously thrown when the callback object was written in Visual Basic.NET and the case of the object’s callback method did not match the case of the callback method in the proxy interface.
- Fixed problem where a Java-to-.NET project could be created without proxies for System.Object and System.Type.
- Fixed problem in .NET-to-Java projects, where a method declared to return an array of interface element type could erroneously throw an InvalidCastException when returning an array of objects of different types that implement the interface.
- Fixed a problem that could sometimes occur during proxy generation in Java-to-.NET projects, where the system localization used non-Latin character sets (for example, Chinese), and a proxy with a field of float or double and a literal value of positive or negative infinity or NaN (not a number) was generated. In such cases, an erroneous exception would be thrown.

NEW IN VERSION 2.2.1

- By-value JavaBean proxies now allow access to static fields, which are passed through to the Java side. Previously, by-value JavaBean proxies did not include the static fields of the underlying Java object.
- Now provides a more informative error message when a Java side is accessed from two .NET sides, one of which has a by-value proxy for a given underlying Java class, and the other of which has a

by-reference proxy for the same underlying Java class. Also, a more informative error message for similar situations in Java-to-.NET projects.

- Command-line version of jnbproxy no longer reports error if there is no /xp option and CLASSPATH environment variable is missing.
- More informative error message in Java-to-.NET shared memory projects, if jnbshare.dll, jnbsharedmem.dll, or jnbjavaentry.dll are not in the GAC.
- More informative exception thrown when a callback method attempts to throw an exception not "understood" by the other side.
- An exception is no longer thrown when attempting to return a multi-dimensional array of primitives in a Java-to-.NET project.
- Fixed problem where an incorrect value might be returned if a release message is processed before a regular access to an object member occurs.
- An exception is no longer erroneously thrown in some situations where accessing a nested class proxy in a Java-to-.NET project.
- Fixed problem where an erroneous exception might be thrown in some situations when dynamically generating a proxy for a nested class.
- An erroneous exception is no longer thrown when generating a proxy for a class that inherits from a mapped collection class.
- By-value-mapped date classes now work correctly in all time zones. Previously they did not.
- Fixed an exception that was erroneously thrown when a public-fields value proxy with a static final field was returned from a call to a proxy method in a .NET-to-Java project. The exception is no longer thrown.
- Previously during proxy generation, the proxy generator would sometimes erroneously report that a proxy could not be generated because it depended on a class not in the classpath. This no longer occurs. Also fixed a related problem that could cause an exception to be thrown during proxy generation.

NEW IN VERSION 2.2

- Callbacks (including events and delegates) are now supported in Java-to-.NET projects. See the *Users' Guide* for more details.
- Pass-by-value objects and direct-mapped collections are now supported in Java-to-.NET projects. See the *Users' Guide* for more details.
- Now supporting conversions between .NET's System.DateTime and Java's java.util.Date, and between .NET's System.Decimal and Java's java.math.BigInteger and java.math.BigDecimal. See sections on direct-mapped classes in the *Users' Guide* for more information.
- DotNetString, DotNetArray, and Java-side boxed values (System.BoxedInt, etc.) now implement toString(). Previously, when toString() was called on objects of either of these classes, an exception would be thrown.
- Added /java option to command-line proxy generator to specify java.exe to use.
- Improved error reporting in Java-to-.NET applications when the user forgets to call DotNetSide.init() at the beginning of the Java client code.

- There are no longer separate installers for the EE and SE versions. Starting with v2.2, there is a single installer, and the usable functionality is determined by the installed license key.
- Removed verbose mode from GUI-based proxy generation tool.
- A more informative error message is included in the thrown exception when .NET-side configuration information is missing.
- An explanatory message has been included in the Java Options dialog box explaining that the communication settings in that box only apply to proxy generation and not to the Java/.NET communication mechanism that will be used by the application that will run JNBridgePro.
- GUI-based proxy generation tool now displays hourglass cursor and informative message when saving or restoring a project file.
- Fixed a problem that sometimes caused proxy generation to fail when the /nj option was used with the command-line proxy generator.
- Fixed a problem in Java-to-.NET projects that could cause an exception to be thrown when an array or collection was returned from the .NET side that contained arrays of non-primitive objects.
- Fixed a problem in Java-to-.NET projects that only allowed the .NET side to be contacted as "localhost," thereby making it impossible to contact the .NET side from another machine, or via an IP address other than 127.0.0.1.

NEW IN VERSION 2.1.3

- Fixed a problem with callbacks in .NET-to-Java projects, where an exception would be thrown when a string or an array was passed to a callback method that expected a java.lang.Object. This situation is now handled correctly, without an exception being thrown.
- Fixed a proxy generation problem that could cause an exception when setting properties in Java-to-.NET projects.
- Fixed a problem that could cause an exception to be thrown when generating .NET-side proxies from JDK 5.0 classes.
- In the JNBProxy GUI-based proxy generation tool, the check/uncheck behavior of JDK 5.0 generic classes (classes whose names contain a '+') is the same as that of nested classes (classes whose names contain a '$') in the environment and exposed proxies panes. Depending on the context, the generic classes or their parents will be automatically checked or unchecked in the same way that nested classes and their parents previously were.
- Fixed a memory leak in .NET-to-Java projects that used HTTP/SOAP communications.

NEW IN VERSION 2.1.2

- Changed behavior of shared-memory communications channel so that ASP.NET Web apps using shared memory will still work after touching configuration files without having to restart IIS. Change will also allow applications being tested under NUnit to be retested with NUnit having the default settings. The change will not affect other applications.
- Fixed a problem when using shared memory where multiple calls from .NET to Java from the same .NET thread appear to be in different Java threads.
- Fixed a problem where the keywords *public*, *abstract*, and *static* may appear more than once in the JNBProxy GUI-based proxy generation tool signature pane for a single property member of a .NET class.

- Fixed a problem where the Java Options dialog box would be repeatedly displayed upon launching the JNBProxy GUI-based proxy generation tool when there is no jnbcore.properties file in the same folder as jnbcore.jar.
- Fixed a problem that could cause an exception to be thrown in Java-to-.NET projects.

NEW IN VERSION 2.1.1

- Fixed a problem that could sometimes cause an exception to be erroneously thrown when generating proxies using the command-line proxy generator.
- Fixed problem causing an erroneous exception to be thrown when JavaSide.disposeAll() was called.
- TCP/binary transport mechanism modified to stop the accumulation of sockets in the CLOSE_WAIT state, which occurred in some rare cases.
- Fixed an incorrect version number in the supplied jnbproxy.config file.
- Fixed an exception erroneously thrown when an application using JNBridgePro 2.1 is used with .NET Framework 1.0.

NEW IN VERSION 2.1

- Added a shared-memory communications channel between .NET and Java. Unlike binary or HTTP/SOAP communications, shared-memory communications does not use sockets, and is substantially faster. See the *Users' Guide* for more information, and for information on when shared-memory communications can and cannot be used.
- Added the ability to configure the JNBridgePro .NET side programmatically, or through the application's configuration file. It's no longer necessary to use jnbproxy.config, although one can still use jnbproxy.config.
- Added the ability to specify the communication method used by the JNBProxy proxy generation tool from within the tool itself; using a jnbproxy.config file is no longer necessary. See the *Users' Guide* for more information.
- When JNBridgePro is used with an ASP.NET Web app, or with NUnit, or in the GAC, jnbproxy.config, if used, can now be placed in the build folder with jnbshare.dll; it need not be placed in <system-drive>:\inetpub\wwwroot or in <system-drive>:\ as previously. Placing it in \inetpub\wwwroot or in \ (the root folder) will continue to work, as previously.
- Added the ability to specify the Java side properties programmatically, or individually on the command line.
- Added the ability to specify the assemblies to be accessed by the .NET side in Java-to-.NET projects through the application configuration file.
- Added a standalone .NET-side server, JNBDotNetSide.exe, for Java-to-.NET projects.
- The <jnbridge> section of the application configuration file of applications using JNBridgePro has been reorganized.
- Added the method JavaSides.isAlive(), to determine whether a given Java-side is currently running.
- More informative error reporting when the JNBProxy proxy generation tool attempts to load a corrupt jar file.
- Fixed a problem where an erroneous exception may be thrown when a proxy is generated that depends on a class not in the proxy generator's classpath. This exception is no longer thrown.

- Fixed problem that caused an exception to be erroneously thrown when the command-line version of jnbproxy to be terminated.

NEW IN VERSION 2.0

- Added ability to call .NET code from Java code, thereby supporting full two-way interoperability. See the *User's Guide* for further information.
- Added ability for a .NET program to discover information about the Java side with which it is communicating, through the JavaSides.getJavaSideInfo() and JavaSides.CurrentJavaSideName APIs. See the *User's Guide* for more information.
- Added ability to export a list of exposed proxy classes from the JNBProxy GUI tool as a text file, for later use as a class file to input to the command-line version of JNBProxy.
- All .NET-side proxies now implement the IDisposable interface. This allows the .NET client program to release large Java objects earlier than they might be released if .NET garbage collection is relied upon. Java-side proxies also implement a similar disposal method. See the *User's Guide* for more information.
- .NET-side proxies now map calls to Equals() and GetHashCode() to the Java-side equals() and hashcode() methods. Similarly, Java-side proxies map calls to equals() and hashcode() to the .NET-side Equals() and GetHashCode(). See the *User's Guide* for more information.
- Automatic implicit conversions between native .NET strings and JavaString objects, and between native .NET arrays and JavaArray objects have been added. See the *User's Guide* for more information.
- .NET-side proxies of Java exceptions now map the Message property to return the same message as the JavaExceptionMessage property. This means that accessing Message yields the Java-originated message rather than a less-useful .NET-originated message. Similarly, the StackTrace property of a .NET-side proxy of a Java exception now returns a combined stack trace containing .NET-side and Java-side stack trace information.
- JNBMain now contains start() and stop() methods that allow the JNBridgePro Java side to be started and stopped programmatically from a Java-side program.
- Implemented a new, more general, class-ordering mechanism for the generation of proxies. Previously, there were rare situations where class dependencies were not detected and a TypeLoadException could be thrown when proxies were generated.

LIMITATIONS AND KNOWN ISSUES

- Java-in-.NET embedding feature has been reimplemented to work with Java 7 and later. There are a few caveats:
    - Only embedding of Java controls inside Windows Forms is currently supported. Embedding Java inside Windows Presentation Foundation (WPF) will be supported in the next release.
    - There may be some unresolved issues related to z-order (that is, in certain situations, the Java control might appear in front of an element that it should not appear in front of, or behind an element that it should not be behind). Please report such issues if you encounter them.
- JNBridgePro proxies cannot be called directly from SQL Server stored procedures. SQL Server 2005 and later require that all called assemblies and their dependencies be pure managed code, and jnbsharedmem.dll contains unmanaged code. As a workaround, create a COM-Callable Wrapper (CCW) for the proxy DLL, and call the CCW from the stored procedure using COM Interop.

- Borland/CodeGear Delphi for .NET generates classes and class members whose names may begin with '@', contrary to .NET guidelines. When accessing such classes and members from Java in a Java-to-.NET project, a Java exception may be thrown, since identifiers beginning with '@' are illegal in Java.
- In certain situations when embedding Java UI elements in Windows Forms, if a callback is executed while the UI element's proxy is created, or the UI element's JavaControl wrapper is created, the Java side will fail and an exception will be thrown. In such cases, it is necessary to surround the creation of these two objects with a synchronization lock to prevent callbacks from being executed during that time.
- In Java-to-.NET projects, returning __TransparentProxy objects is not supported. For example, if you proxy System.Activator and call Activator.GetObject() where the underlying object is in a remote application domain, an exception will be thrown because __TransparentProxies are not supported. As a workaround, all calls to GetObject() should be performed on the .NET side and accessed from Java through a proxied façade method. The values returned by GetObject() should be wrapped in a pass-through wrapper class before being returned to the Java side.
- When using SSL, client authentication is not yet supported. Only server authentication is supported.
- SSL is not available for Java-to-.NET projects using the HTTP/SOAP communications mechanism. All other combinations of directions and communications mechanisms support SSL.
- When mapping Java enums to .NET enums, special programmed behaviors of the Java enum types will not be accessible from .NET, since the proxies are .NET enum types.
- When using shared-memory, it may be necessary to move classes from the regular classpath to the boot classpath to allow certain classes to be found, and to avoid ClassNotFoundExceptions and NoClassDefFoundErrors. See the *Users' Guide* for more information.
- In Java-to-.NET projects, members of .NET classes that use or return ref or out parameters, or that return or take parameters of a nullable type (e.g., bool?), will not be proxied.
- In Java-to-.NET projects, if a Java-side proxy is generated for a .NET-side interface, and the interface contains constant field declarations, if a field is an object that is neither a primitive nor a string, it is possible that the object's constructor will be executed. If the constructor has side-effects, or throws an exception, problems could arise. It is unlikely that this will pose a problem in real usage, since neither C#, C++, nor VB.NET support interfaces with fields (only J# does), and in Java/J#, while it is legal to have a non-primitive/string field constant in an interface, it is not a recommended practice.
- When using the JRockit JVM, the class literal feature (.JavaClass) and calls to the single-parameter version of java.lang.Class.forName() may not behave as expected. The workaround is to use java.lang.Class.forName("class name", false, java.lang.Thread.currentThread().getContextClassLoader()). You will need to generate proxies for java.lang.ClassLoader and java.lang.Thread. This workaround will also work with other JVMs.
- Thread-true classes only work with the TCP/binary communications channel and with shared memory. They do not currently work with the HTTP/SOAP channel.
- Array parameters are always passed by value. Therefore, it is not possible to return results through an array parameter. Similarly, passing arrays by value leads to special considerations when assigning to an element of an array which is a field of a proxy object. These issues are discussed in the section "Arrays" in the *User's Guide*.

- With (non-asynchronous) callbacks, the developer must be careful to avoid deadlock in certain cases where the main thread of a Windows Form performs a Java call that leads to a callback. See the *User's Guide* for details.
- .NET callback objects that have been passed to the Java side cannot be returned from the Java side back to the .NET side as return values in Java methods.
- A reference to a Java object residing on one Java-side may not be sent to a Java method residing on a different Java-side.
- .NET classes may only access public and protected members of Java classes, and Java classes may only access public and protected members of .NET classes, with the exception of explicit interface implementations, which are supported.
- When Java-originated exceptions are thrown back to .NET, the messages included in the Java exception do not appear in the message field of the .NET proxy exception object. The messages can be found by accessing the JavaExceptionMessage property of the .NET exception proxy object, and the Java-side stack trace can be found by accessing the JavaStackTrace property.