JNBridgePro™ Evaluation and Quick Start Guide

Version 10.0

jnbridge.com

JNBridge, LLC
jnbridge.com

April 23, 2019

Table of Contents

Preface

This document contains general guidelines for evaluating JNBridgePro, including steps to install and configure, pointers for building your first integrated Java/.NET application, feature summaries, sources for additional information, and a list of business benefits.

For more detailed information, please see the following documents:

- *JNBridgePro Users' Guide* contains information on running and using JNBridgePro.
- *JNBridgePro for .NET Core Users' Guide* contains information to running and using JNBridgePro in connection with .NET Core. All of this information is also in the general *Users' Guide.*
- *JNBridgePro Release Notes* contains the latest information on system requirements and known issues.
- The *demos directory* included in the JNBridgePro installation directory contains a variety of sample code and instructions for building the examples, including integrating with common Java EE application servers.
- *Getting Started with the JNBridgePro Plug-ins for Visual Studio and Eclipse* explains how to use the JNBridgePro plug-ins for Visual Studio and Eclipse as part of your development cycle.

What is JNBridgePro?

JNBridgePro is a Java/.NET interoperability tool that enables full two-way interoperability between Java and .NET. With JNBridgePro, Java can access .NET objects and classes, and .NET can access Java classes and objects. The Java and .NET sides can each be on the ground, or in the cloud. It will work with the .NET Framework, and with .NET Core (starting with .NET Core 3.0) on Windows and Linux. (MacOS support will come in a future release.)

Cross-platform inheritance and callbacks are supported, as is bi-directional operation, where .NET code calls Java *and* Java code calls .NET in the same program. JNBridgePro allows the Java and .NET code to run in the same process, on the same machine in different processes, on different machines on your LAN, or on different machines across the Internet. In each case, your code need not be changed, only the configuration files. JNBridgePro works off of binaries: if you're calling Java code from .NET, you only need to have the Java binaries; you don't need the Java source code. Similarly, if you're calling .NET from Java, you don't need the .NET source code. JNBridgePro also allows you to access all Java EE facilities from .NET, including EJBs and JNDI.

Getting started

Download the installation file

If you do not already have a copy, you may download JNBridgePro from the JNBridge web site, https://jnbridge.com, at any time.

Before you install: usage scenarios

Use of JNBridgePro falls into two broad scenarios: **proxy generation** and **proxy use**. JNBridgePro is installed, configured, and used in a slightly different manner for each scenario.

Proxy generation typically occurs during the development phase. Depending on the direction that you want the inter-platform calls to go (.NET-to-Java or Java-to-.NET), you specify the classes on one platform with which you want the classes on the other platform to communicate. You generate the proxy classes, each of which manages communications with the corresponding class on the other platform. For example, if you are generating proxies for .NET-to-Java calls, you will identify the Java classes with which you want the .NET classes to communicate, then generate the proxy classes for the .NET side. The proxy classes manage communication with the corresponding Java classes.

Proxy use generally occurs during deployment testing and production, where classes on one platform communicate with classes on the other platform by way of the generated proxies. For example, in the .NET-to-Java direction, the .NET classes communicate with Java classes via the generated .NET proxy classes.

Installation and system configuration

The following steps are required to install and configure JNBridgePro:

1. **Run the installer**, `jnbSetup10_0.msi` on the development machine where you will generate the proxies, and select the *Development Configuration* option. Depending on which versions of the .NET Framework are installed on your machine, the installer will install the .NET 2.0/3.0/3.5-targeted version of JNBridgePro, the .NET 4.0/4.5/4.6/4.7-targeted version of JNBridgePro, or both. It will install both 32-bit and 64-bit components, regardless of whether you are on a 32-bit or a 64-bit machine. The .NET Core-targeted components will always also be installed.
2. **Activate the license.** Launch the JNBridgePro registration tool, then click on the "Online License Activation" tab. Find the activation key that you received via email when you downloaded the product, paste it into the activation key slot, and click on the "Activate License" button. You now have a 30-day evaluation license. (If your installation does not have Internet connectivity, visit https://jnbridge.com/support/license-key from a different machine and follow the instructions there, or contact registration@jnbridge.com.)
3. **Launch the proxy generation tool and configure the communications** between the appropriate .NET and JVM machine or machines. (Please see the *Initial Invocation* section in the *JNBridgePro Users' Guide* for detailed instructions.) For demo purposes, it's probably best to use the defaults. You can also try out the Visual Studio and Eclipse plug-ins (see the document *Getting Started with the JNBridgePro Plug-ins for Visual Studio and Eclipse* for more information.)

Now you've got JNBridgePro set up for proxy generation, and for working through the various demos. If you're trying to access Java code that's on a different machine, you may need to configure and deploy the JNBridgePro Java side to the other machine and start the Java side manually. See the *JNBridgePro Users' Guide* ("Generating proxies with JNBridgePro: Starting Java manually") for more information.

License types and license keys

When you install JNBridgePro and activate the license as described above, you will automatically be granted a 30-day free trial period. At any time during or after the 30-day trial period, you may purchase a permanent license. You may purchase a license on our Web site, https://jnbridge.com/software/jnbridgepro/license-and-purchase, or by contacting sales@jnbridge.com.

If 30 days is an insufficient amount of time to either evaluate JNBridgePro or to obtain purchasing authorization, you may, at our discretion, obtain a trial-extension key – but you have to ask. You will also need to provide us with the registration key that you can obtain from either the GUI version of JNBProxy or the Registration Tool. Please see the licensing section of the *JNBridgePro Users Guide* for detailed instructions.

When JNBridgePro is used with an expired or invalid license, it will throw a `com.jnbridge.jnbcore.InvalidLicenseException`.

When you are ready to purchase a JNBridgePro license, you may purchase licenses, depending on the desired capabilities. A *development license* allows you to use the proxy generation tool as part of program development. A *deployment license* allows you to use those proxies and the rest of JNBridgePro as part of an application that interoperates between Java and .NET. The table below summarizes the various licenses.

Available Features	Development	Deployment
Generate proxies	✓	
Use proxies	✓	✓
Deploy to Java EE application servers	✓	✓
Cross-platform transactions	✓	✓
Access EJBs, JMS, and JNDI	✓	✓
.NET side can interact with multiple Java sides	✓	✓
Java side can interact with multiple .NET sides	✓	✓
Multiple proxy DLLs	✓	✓
Embed Java GUI elements in a .NET app, and vice versa	✓	✓
Work with .NET Core on Windows and Linux	✓	✓
Failover	✓	✓
All other JNBridgePro features, including: • Full 2-way interop • Callbacks • Cross-platform overrides • Pass by value or by reference • Shared memory	✓	✓

Please note that there are additional license types that allow you to run JNBridgePro in a virtual machine image, with a terminal server, or in the cloud.

Please see https://jnbridge.com/software/jnbridgepro/license-and-purchase for more information on JNBridgePro licensing and pricing.

How do I…

… call Java from .NET?

The best way to get started with JNBridgePro is to work through one of the demo examples we provide. It only takes a few minutes, and when you're done you'll know just about everything you need to tackle your own project. JNBridgePro includes two basic .NET-to-Java examples for you to try. The "Log demo" shows how to create a simple .NET console application that accesses a Java package (in this case, the Java log4j logging package). The "Books demo" shows how to connect .NET user interface code to a Java back end. It includes two projects: one to connect a Windows Forms-based GUI with the Java back and, and one to connect an ASP.NET-based Web application with the Java back end.

The demos walk through everything you need to do, and include all the necessary source code. They're installed as part of the JNBridgePro installation process, and can be found through the Start menu.

… call .NET from Java?

The best way to get started with JNBridgePro is to work through the provided demo. It only takes a few minutes, and when you're done you'll know just about everything you need to tackle your own project. JNBridgePro includes one basic Java-to-.NET example for you to try. The "Java-to-.NET demo" shows how to create a simple Java Swing-based application that causes Windows Forms to pop up when certain actions are taken. It also demonstrates how .NET methods can fire Java-based callback methods.

The demo walks through everything you need to do, and include all the necessary source code. They're installed as part of the JNBridgePro installation process, and can be found through the Start menu.

… use JNBridgePro in the cloud?

JNBridgePro allows you to interoperate between .NET and Java anywhere. One side can be in the cloud and the other can be on the ground, or both sides can be in the cloud. The two sides can be in the same cloud image or they can be in different cloud images, and the two images can be on the same cloud platform (including Windows Azure or Amazon EC2) or on different cloud platforms. For more information, contact info@jnbridge.com, and see the information and examples on our website.

… use JNBridgePro on Linux?

You can run the .NET side (as well as the Java side) of your application on Linux using .NET Core. Both TCP/binary and shared memory communications is supported. For more information, see the *JNBridgePro for .NET Core Users' Guide*, and the relevant .NET Core examples from our website.

You can also run the .NET side of your application on Linux using the Mono runtime. Currently, we support use of Mono on both Linux (x86 and x64) and Windows. For more information, please see the *JNBridgePro Users' Guide*.

… use JNBridgePro with .NET Core?

You can run the .NET side (as well as the Java side) of your application using .NET Core, on both Windows and Linux. (.NET Core on MacOS will be supported in a future release.) Both TCP/binary

and shared memory communications is supported. For more information, see the *JNBridgePro for .NET Core Users' Guide*, and the relevant .NET Core examples from our website.

… access an EJB from .NET?

The JNBridgePro installation includes lots of examples showing how an EJB can be accessed from .NET. The details are slightly different depending on which Java EE application server you're using, so we've included examples using all the leading Java EE app servers. If you plan to create a project that accesses EJBs from .NET, we recommend that you work through the supplied example that uses the app server that you intend to use in your project.

The demos walk through everything you need to do, and include all the necessary source code. They're installed as part of the JNBridgePro installation process, and can be found by drilling down into the Start menu.

If you don't see an example that employs your desired Java EE application server, tell us! We'll do our best to create a new example for you.

… use callbacks?

JNBridgePro not only allows you to call .NET code directly from Java (or Java code directly from .NET), it also allows you to register .NET objects as listeners that can be called back from Java code, or to register Java objects as listeners to be executed when .NET events are fired.

To use a .NET class as a callback in a .NET-to-Java project, it must implement a proxy of the appropriate Java listener interface, and be annotated with a [Callback] or [AsyncCallback] attribute. To use a Java class as a callback in a Java-to-.NET project, it needs to implement the appropriate proxy delegate or event handler attribute. Details of how this is done can be found in the "Callbacks" section in the part of the *Users' Guide* appropriate to .NET-to-Java or Java-to-.NET projects.

The Java-to-.NET demo supplied with JNBridgePro includes callbacks and illustrates how they are used in Java-to-.NET projects.

… use cross-platform overrides?

When you proxy a Java class, you can write a .NET class that inherits from the proxied Java class and overrides methods in the original Java class. Because the .NET class inherits from a Java class, you can pass the .NET class back to the Java side (where it looks like the original Java class), but when the overridden Java method is called, the .NET method that overrode it will be executed instead. This also works when you proxy a .NET class and write a Java class that inherits from it and overrides methods in the original .NET class.

Details of how this is done can be found in the "Cross-platform overrides" section in the part of the *Users' Guide* appropriate to .NET-to-Java or Java-to-.NET projects.

… extend software frameworks across platforms?

Software frameworks support extensibility and customizability through overriding default method behavior and through abstract classes that can be subclassed into concrete classes. Up until now, software frameworks could only be extended on the same platform on which they were implemented; that is, Java-based frameworks could only be extended in Java, and .NET-based frameworks could only be extended in a .NET language.

Now with JNBridgePro, you can take a class in a Java framework and extend its behavior in .NET by proxying the class, subclassing the proxy as a .NET class, and adding a method in that .NET class that overrides underlying Java method. When an object of that class is passed back to the Java side, and the overridden method is called from Java code, the override behavior is executed on the .NET side. You can similarly use JNBridgePro to override .NET methods in Java code.

You can also now proxy an abstract Java class, subclass that proxy on the .NET side, and instantiate the .NET subclass (and similarly proxy an abstract .NET class, subclass it on the Java side, and instantiate the Java subclass).

Details can be found in the sections "Abstract classes" and "Cross-platform overrides" in the part of the *Users' Guide* appropriate to .NET-to-Java or Java-to-.NET projects.

… embed Java GUI elements in a .NET Windows Forms or Windows Presentation Foundation application, or embed .NET GUI element in a Java application?

We provide demos showing how Java GUI elements (AWT or Swing) can be embedded in a .NET WinForms or WPF application, and also showing how .NET GUI components (WinForms and WPF) can be embedded in an AWT, Swing, or SWT application.

The demos walk through everything you need to do, and include all the necessary source code. They're installed as part of the JNBridgePro installation process, and can be found by drilling down into the Start menu.

… change the communication mechanism?

JNBridgePro supports several communications mechanisms based on the user's architectural or performance requirements. There is a shared-memory mechanism that allows the .NET and Java sides to run in the same process and to communicate through shared memory structures: it is much faster than the alternatives and can be used when the Java and .NET sides are on the same machine. There is a high-performance socket-based communications mechanism that employs TCP-based connections and a binary formatter that can be used when the .NET and Java sides must run in separate process or on separate machines. There is also a socket-based communications mechanism using HTTP and a SOAP formatter that can be used when the communication between the Java and .NET sides must pass through firewalls. In all cases, the communication mechanism is typically specified through the application configuration file on the .NET side, and through a Java properties file on the Java side. The basic demos included with the JNBridgePro installation discuss how the communications mechanism can be changed by changing the configuration information. They provide examples of how to change configuration parameters, where certain file segments can be commented out and other segments can be uncommented and edited to change the communications from TCP/binary to shared memory.

It's also possible to specify the communications mechanism programmatically, through an API that's discussed in the *Users' Guide* ("Configuring the .NET side: Configuring communications programmatically").

… use secure communications?

JNBridgePro supports secure communications between .NET and Java using SSL (Secure Sockets Library). SSL provides data encryption, message integrity, and authentication. (JNBridgePro currently supports server authentication only; client authentication will be supported in an upcoming

version. Secure communications can be used with both tcp/binary and http/soap communications, and in both the .NET-to-Java and Java-to-.NET directions. (Note that currently secure http/soap communications is not supported in the Java-to-.NET direction; it will be supported in an upcoming version.

The best way to experience secure communications is to work through one of the supplied demo programs, and then follow the instructions to secure the communications. The process is straightforward: no code need be changed, only configuration information.

The *Users' Guide* provides comprehensive instructions on secure communications in JNBridgePro.

Explore additional capabilities

Have you explored these top features of JNBridgePro?

- **Generate proxies from within an Integrated Development Environment.** In addition to the standalone proxy generation tool, it is now possible to generate proxies from within Visual Studio or Eclipse, and to use the results of that proxy build directly from a .NET or Java project, thereby greatly simplifying the build process. For more information, please see the document *Getting Started with the JNBridgePro Plug-ins for Visual Studio and Eclipse*.
- **Secure communications.** Socket-based communications between .NET and Java (through the tcp/binary and http/soap channels) can be secured using SSL (secure sockets library). SSL provides data encryption, message integrity, and authentication of the parties to the communication. You can secure your communications without changing your code; only the configuration information need be changed.
- **Generic classes.** The .NET Framework and Java support generic classes. JNBridgePro allows you to take advantage of this feature. You can generate proxies for generic classes on the other platform and instantiate them, then access their members. You can also instantiate and access generic methods of non-generic classes.
- **Mappings between enums.** .NET has had enum types from the beginning, but Java added them with Java SE 5.0. For efficiency and convenience, JNBridgePro automatically maps between Java and .NET enum classes.
- **Shared-memory communication**. It's possible to run the Java and .NET sides in the same process and have them communicate using shared memory structures. For many applications, this is the best alternative and is definitely the fastest. It's also possible to have the Java and .NET sides run in different processes and communicate via sockets. This allows the two sides to reside on different machines on a network.
- **Reference vs. value objects.** By default, JNBridgePro passes objects between Java and .NET by reference. Passing objects by reference is generally faster, since a reference is typically smaller than the actual object it represents. However, if you need to repeatedly access members of the object, accessing a reference object takes longer since each access is a round trip from the .NET side to the Java side and back (or vice versa). In order to address this issue, JNBridgePro allows you to pass objects by value. When an object is passed by value, a snapshot of the object is passed from one side to the other. While this will take a little longer than passing by reference (since an object is typically larger than its reference), subsequent accesses to members of the value object are much faster, since they are all local and don't require a round trip. See the *Users'*

Guide (“Reference and value proxies”) for instructions on how to use value objects in JNBridgePro.

- **Mapped classes**. For convenience, JNBridgePro automatically maps between a large number of equivalent .NET and Java classes. For example, if a Java method returns an integer, the .NET code that calls that method through JNBridgePro will automatically receive a .NET integer in return, even though the original method returned a Java integer. All the numeric classes are mapped, as are characters and Boolean values. Because they’re so frequently used, JNBridgePro automatically converts between Java and .NET strings, and between Java and .NET arrays. In addition, JNBridgePro allows the user to optionally specify that various collections (for example, Java vectors and .NET array lists, and Java and .NET hash tables), Java and .NET dates, and Java and .NET extended-precision values are automatically converted. See the sections on mapped classes in the *Users’ Guide* for more information.
- **Multiple Java sides**. If you have an server deployment license, it’s possible to have your .NET side call Java code in multiple Java sides. Those Java sides can even be on different machines. See the *Users’ Guide* (“Interacting with multiple Java sides”) for more information. It’s also possible for a Java side to call .NET code in multiple .NET sides. This feature is available in both the SE and EE versions. See the *Users’ Guide* (“Interacting with multiple .NET sides”) for more information.
- **Multiple proxy DLLs**. Your application can now include more than one proxy DLL. This can be useful if you are creating an application from several Java libraries, and you want the generated DLLs to stay separate – perhaps they are still under development and are being developed by different teams. Alternatively, you might have a proxy DLL that was generated previously, and now wish to generate proxies for new classes without having to touch the old proxy DLL. For more information, see the section “Multiple proxy DLLs” in the *Users’ Guide*.
- **Embedding GUI elements**. You can embed Java AWT and Swing components inside WinForms and WPF applications, and you can embed .NET GUI (WinForms or WPF) components inside AWT, Swing, *and* SWT applications (even Eclipse plug-ins!). See the *Users’ Guide* (“Embedding Java GUI elements inside .NET GUI applications,” and “Embedding .NET GUI elements inside Java GUI applications”) for more information.
- **Failover**. JNBridgePro provides a basic failover capability, so that if a primary Java side fails, the .NET side will automatically send all subsequent requests to a designated failover Java side. See the section on failover in the *Users’ Guide*. It’s also possible for JNBridgePro to take advantage of an application server’s clustering capability: a technical note on JNBridgePro and clustering can be downloaded from the JNBridgePro Web site.
- **Cross-platform transactions**. It is possible for transactional .NET and Java operations to be part of the same transaction, so that if either the participating .NET or Java operations fail, both the .NET and Java operations are rolled back. See the section on cross-platform transactions in the *Users’ Guide*.
- **Saving and restoring proxy generation projects**. Using the JNBridgePro GUI-based proxy generation tool, it is possible to save proxy generation projects to a file, and to open them and continue work on them later. Among other things, this allows for incremental development, where you generate proxies for a number of classes now, then add more classes later when the project’s requirements evolve.

- **Run proxy generation as part of a build script**. JNBridgePro comes with a command-line version of the proxy generation tool. This tool can be run as part of a script. It can be set up to generate proxies from a text file containing a list of classes. That class list can even be generated by the GUI-based version of the proxy generation tool. That way, the initial proxy generation can be done incrementally through the GUI-based tool, and subsequent builds can be done as part of a script using the command-line tool.
- **Access a COM component from Java**. You can use JNBridgePro to access a COM component from Java. You can use the .NET development tools to create a Runtime Callable Wrapper (RCW), which is a .NET assembly that acts as a wrapper for the COM component. Then, generate Java-side proxies from the RCW, and call them as you would in a conventional Java-to-.NET project.
- **Access Java from a COM component**. You can also use JNBridgePro to access Java code from a COM component. Use the JNBridgePro proxy generation tools to create a proxy DLL from the Java classes you want to call. Then, use the .NET development tools to create a COM Callable Wrapper (CCW), a wrapper for the proxy DLL that looks like a COM component, and call it from the COM component.
- **Access a Java RMI server or CORBA server from .NET.** Although JNBridgePro uses .NET remoting protocols for inter-platform communication, it's still possible to use it to access classes on an RMI or CORBA server. Simply generate the Java client RMI or CORBA stubs as one normally would, then generate JNBridgePro proxies for those stubs and use them to access the stubs from .NET. Use shared memory for the communication between the .NET code and the Java stubs.

Where can I find additional information?

The following sources of further information are included with the JNBridgePro installation, which you can typically find at C:\Program Files (x86)\JNBridge\JNBridgePro v10.0, although it can be installed elsewhere.

- **Users' Guide** explains how to install, configure, and license JNBridgePro, and how to build your Java/.NET applications.
- **Users' Guide for .NET Core** explains how to install, configure, and license JNBridgePro when using .NET Core, and how to build your Java/.NET Core applications.
- **Release Notes** contain the latest information on system requirements, recent changes, and known issues.
- **Demos** contains sample code and explanations for how to build and run the demos.
- **J2EE examples** demonstrate how to set up and configure JNBridgePro with the leading web application servers, including JBoss, Glassfish, WebLogic, and WebSphere.

Other sources of information include:

- The **JNBridge Web Site**, https://jnbridge.com, has the latest copies of white papers, case studies, documentation, demos, and other updated information, available for download or perusal at any time.
- **Contact us** at any one of the following addresses for information or assistance:

sales@jnbridge.com	pricing and licensing
support@jnbridge.com	technical assistance
registration@jnbridge.com	license key requests
info@jnbridge.com	general inquiries

What's the ROI (Return on Investment) story I can tell my boss?

Once you've successfully completed your JNBridgePro technical evaluation, you may need to justify the solution to your management team. We'd be delighted to help you with this, but in the meantime here are some thoughts you can use to start your presentation:

- Spend hours instead of months on solving your Java/.NET interoperability needs
- Leverage your investment in existing Java or .NET code and libraries with complementary .NET or Java capabilities
- Make use of heterogeneously skilled development teams with both Java and .NET expertise
- Migrate your heterogeneous application to the cloud, or develop a new cloud-based application to take advantage of various economies of cloud-based technologies
- Produce Java and .NET versions of your application that share a common code base, thereby decreasing your development and maintenance costs
- Perform Java/.NET co-development, creating applications that make use of the best features of both platforms.
- Your Java code remains cross-platform, and will run on any system that supports Java - not only Windows
- Your .NET code can run on both Windows and on Linux (using the Mono runtime).
- JNBridgePro will continue to work as both Java and .NET evolve, so you're not locked in to a particular version of Java or .NET