



SPANNING JAVA & .NET

White Paper

JNBridgePro™: A Technical Overview

www.jnbridge.com



Introduction

JNBridgePro is a Java & .NET interoperability tool that allows you to call anything Java from .NET, and anything .NET from Java. JNBridgePro allows classes written in .NET languages like C#, C++, and Visual Basic to seamlessly and transparently access Java classes as though Java were itself a .NET language. JNBridgePro supports J2SE or J2EE and the leading J2EE application servers, allowing .NET code to access all J2EE facilities including JMS, EJBs and JNDI.

JNBridgePro is designed so that the .NET objects interact with the Java objects through an automatically generated set of proxies. The Java objects behind the proxies may be on the same machine as the invoking .NET objects or on a different machine on the local area network or across the Internet; communication between the .NET and the Java objects uses .NET's remoting mechanism, and both HTTP/SOAP and a fast binary protocol are supported. In addition, an in-process, shared-memory communication channel is supported that allows the .NET CLR and the JVM to run inside the same process, and to communicate using shared memory structures, and not the socket-based mechanisms used by the other communication channels.

JNBridgePro can be deployed into cloud-based applications, in order to support Java-.NET interoperability within a cloud instance, between instances in the same cloud, between instances in different clouds, and between cloud-based and on-premises code.

JNBridgePro also supports calls from Java to .NET through a similar mechanism to that which supports calls from .NET to Java.

JNBridgePro provides the following benefits:

- All versions of Java, from JDK 1.3.1 onward, are supported, as are all versions of the .NET Framework.
- JNBridgePro works with both 32-bit and 64-bit applications.
- The Java code and the .NET code can run in the same process, on the same machine in different processes, or on different machines communicating over a network. To change from one case to another, the Java and .NET code need not be changed: only the configuration must be changed.
- The solution only requires Java binaries or .NET assemblies; source code is not necessary.
- .NET code can access J2EE facilities, including Java Message Service (JMS), Enterprise Java Beans (EJBs), and Java Naming and Directory Interface (JNDI).
- JNBridgePro can be used to bridge to Java code from standalone .NET applications, as well as from .NET-enabled Microsoft applications such as BizTalk Server, SharePoint Server, Excel, Word, and Outlook.
- .NET classes can register themselves with Java as listeners; Java classes can be used as .NET event handlers.
- Java UI elements (AWT, Swing, SWT) can be embedded in .NET Windows Forms, and WinForms elements can be embedded in Java UI applications.



- JNBridgePro can be used to integrate transaction handling on both the .NET and Java sides, so that the actions on both sides become part of the same transaction. In such integrated transactions, a failure on one side will call rollbacks on both the .NET and Java sides.

JNBridgePro is efficient, easy to use, and makes Java/.NET interoperability seamless and transparent.

The discussion below gives a technical overview of the mechanisms by which .NET code calls Java. The mechanism for Java calling .NET is similar.

Generating Proxies

For each Java class that will be accessed from .NET, a .NET-based proxy for that class must be generated. JNBridgePro provides a proxy generator application (Figure 1) that allows the developer to explore the interfaces and functionality of the available classes to determine which classes should have their functionality exposed, and then to generate those proxies. In addition to the standalone proxy generator, JNBridgePro includes proxy generators that are plug-ins for Visual Studio (Figure 2) and Eclipse (Figure 3). These classes can reside in individual class files, or in JAR files. If the developer is unsure which classes should have their functionality exposed through proxies, the proxy generator tool will make an accurate estimate of the set of classes needed to handle all method parameters, return values, fields, interfaces, and exceptions. Once the set of classes has been chosen, the tool is used to generate the set of proxies, which are written out to a .NET assembly DLL file.

Each class whose functionality is to be exposed will cause a proxy of the same name to be generated. The generated proxy's members (including constructors, methods, and fields) will correspond to the members of the Java class underlying the proxy.

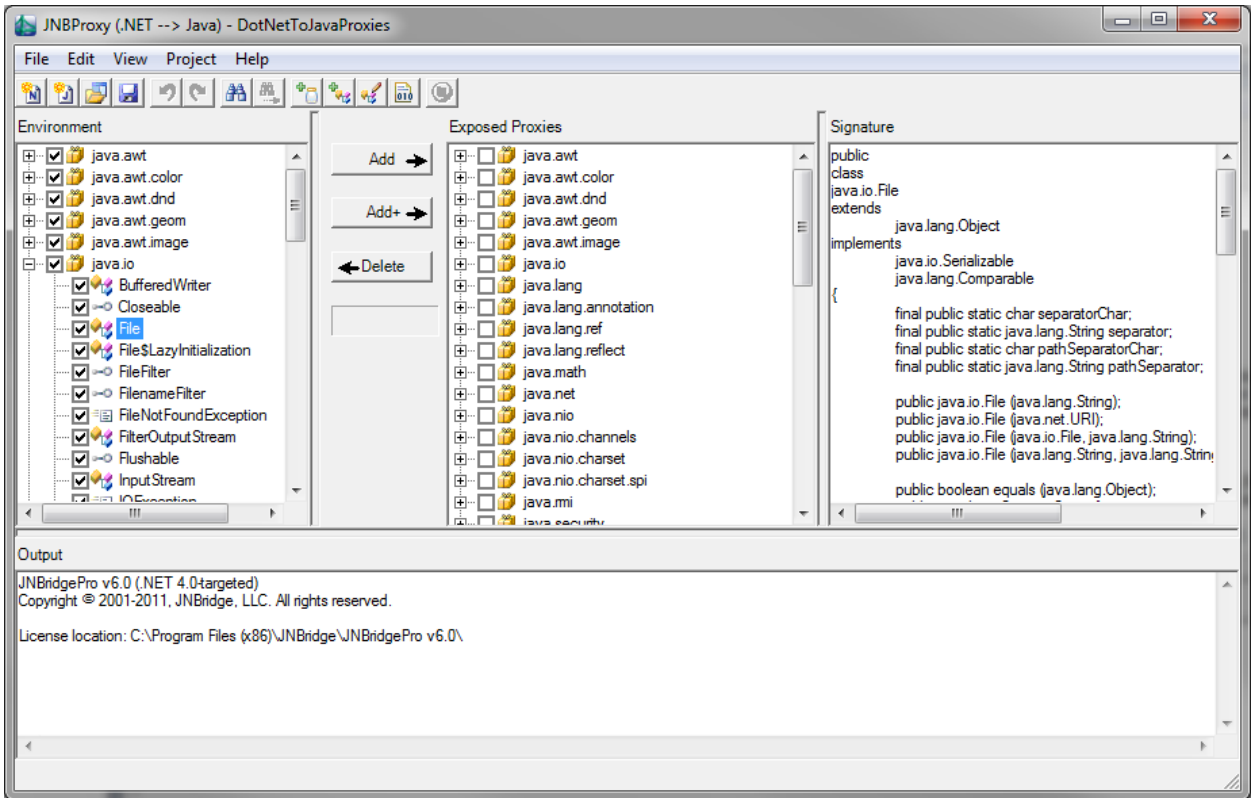


Figure 1. JNBridgePro Proxy Generation Tool

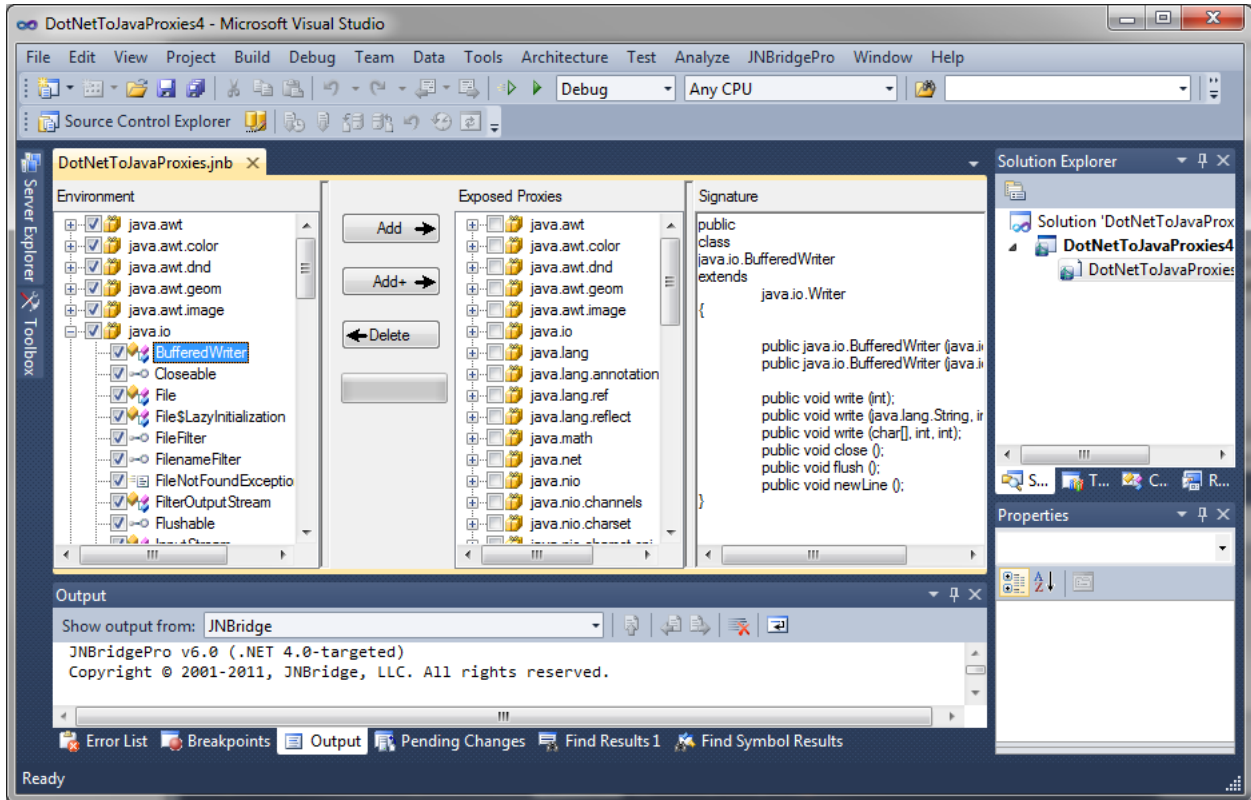


Figure 2. JNBridgePro plug-in for Visual Studio

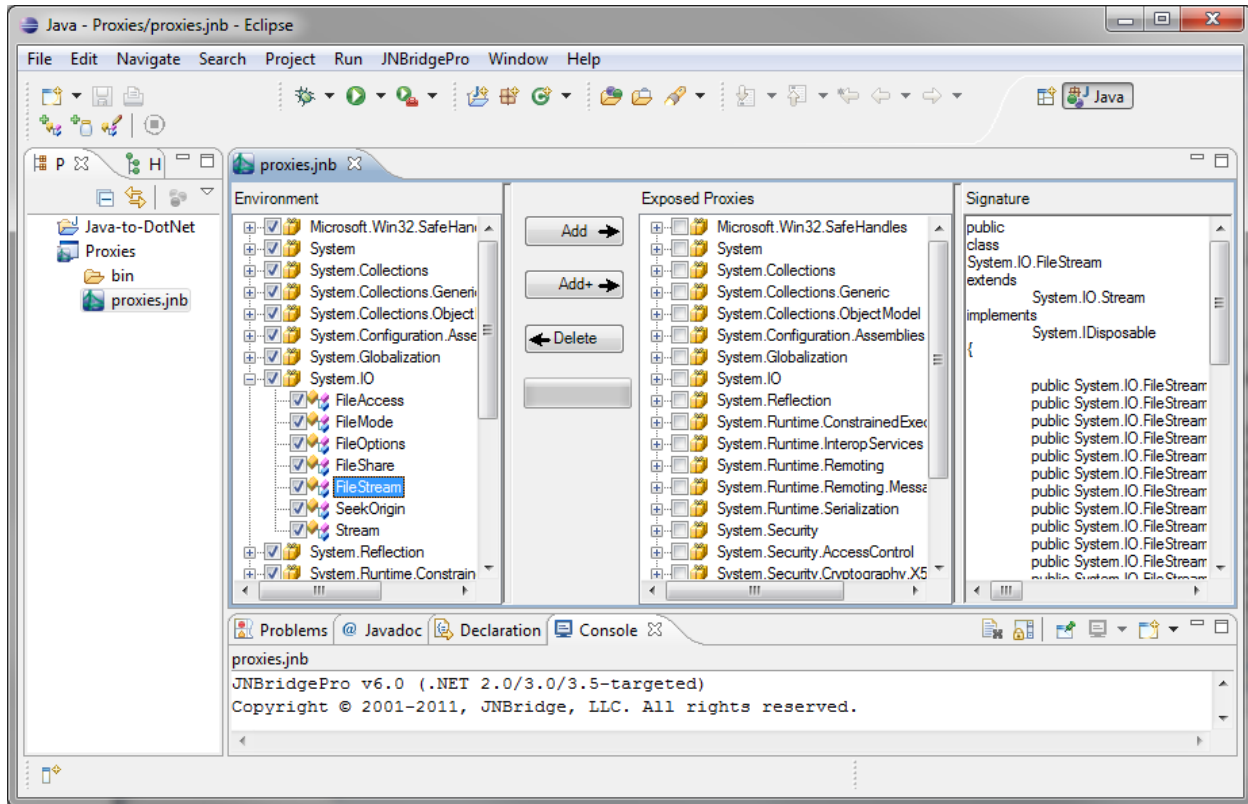


Figure 3. JNBridgePro plug-in for Eclipse

Java-.NET Integration

Once the proxy DLL is generated, the developer adds it to the current project, and now has the ability to access the underlying Java classes from .NET. To construct a Java object from .NET code, the .NET code simply calls a constructor (corresponding to the Java constructor with the same signature) of the corresponding proxy class. To invoke an instance method or field on a Java object, the .NET code calls the appropriate method or accesses the appropriate field on the corresponding proxy object. To invoke a static method or field on a Java class, the corresponding method or field is called or accessed on the corresponding proxy class. If a Java method throws an exception, the .NET method calling the Java method can catch the exception. Finally, if the proxy object is no longer referenced and is garbage collected, the corresponding Java object will be made eligible for garbage collection if there are no other references to it.

A .NET class can be made to inherit from a Java class simply by making the new class a subclass of the Java class's corresponding proxy class. Calls to any methods that the new class does not override will be directed to the underlying Java class. .NET classes can also implement Java interfaces simply by implementing the interfaces' proxies.

.NET code can be called from Java through the standard Java callback mechanism, by implementing Java listener interfaces and registering the .NET classes with the Java code as listeners. The Java code performing the callbacks does not need to be changed in any way.

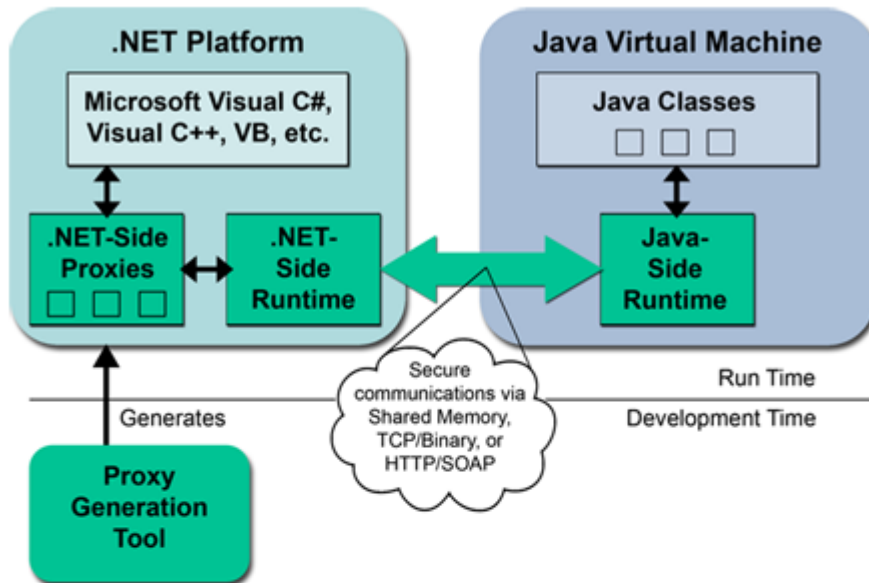


Java primitive values returned from calls to Java methods are automatically converted to the corresponding .NET primitives. Similarly, Java strings are automatically converted to native .NET strings, and Java arrays are converted to native .NET arrays. Conversely, .NET primitives, strings, and arrays can be passed as parameters to Java classes; they are automatically converted to their Java equivalents. JNBridgePro automatically maps between native Java and .NET collection objects. Java ArrayLists, LinkedLists, Vectors, and HashSets are converted to .NET ArrayLists when they are returned from a .NET call to a Java method, and are converted from .NET to Java when passed as parameters in a call from .NET to a Java method. Similarly, JNBridgePro automatically maps between Java Hashtables and HashMaps and .NET Hashtables.

JNBridgePro supports both reference and value objects. While it is faster to pass a value by reference than by value (a JNBridgePro reference is smaller than the corresponding object), subsequent accesses to members of a reference object require additional round trips between the .NET and Java sides. Conversely, passing an object by value between the .NET and Java sides may initially take longer because the value object is larger, but subsequent member accesses are much faster because the data is local and round trips between the .NET and Java sides are unnecessary. The choice of using a reference or value object depends on the performance requirements of the application being developed. JNBridgePro supports two types of value objects: generalized value objects in which the public fields of the Java object are copied over to the .NET side, and Java Bean-style value objects, where the results of get/set methods in the Java object are copied over to the .NET side. Java Bean-style value objects are well suited to efficient access of Enterprise Java Beans (EJBs).

Java-to-.NET integration: The description above covers the situation where .NET code calls Java code (and the Java code calls the .NET code back through a callback mechanism). JNBridgePro provides equivalent capability for projects where Java code calls .NET code, and can even be used to create bidirectional projects where the calls between Java and .NET go in both directions.

.NET to Java



Java to .NET

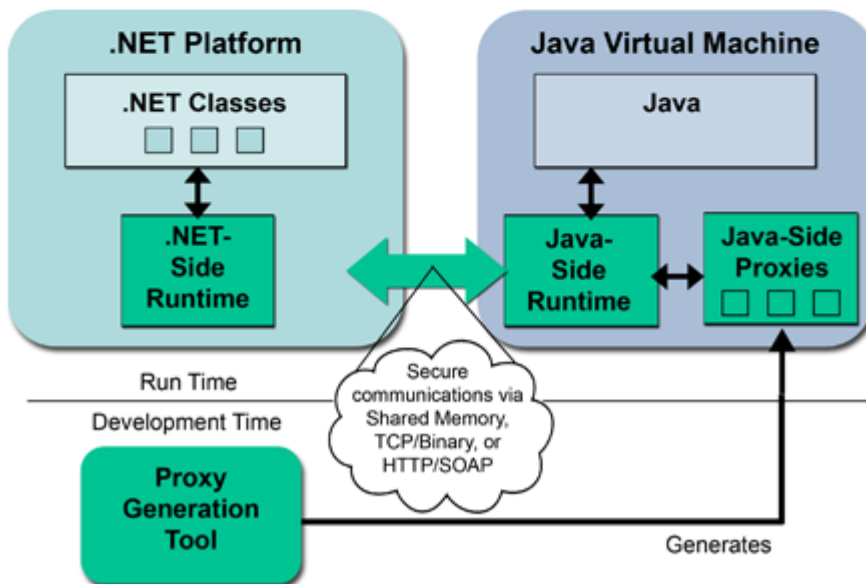


Figure 4. JNBridgePro Architecture



JNBridgePro Architecture

A system using JNBridgePro consists of components on both the Java side and the .NET side (Figure 4). On the .NET side are the classes of the driving .NET application, the assembly consisting of the generated proxies, and a runtime component containing a set of core proxies and classes to manage Java-.NET communication and references to Java objects. On the Java side are the Java classes (either as individual class files or JAR files), and a Java runtime component that manages communication, method dispatch, and object references on the Java side.

If in-process, shared-memory communication is used, the .NET CLR and JVM are both started automatically when the application is started. (If the main application is a .NET application, it will automatically start the JVM inside the .NET process, and if the main application is a Java application, it will automatically start the CLR inside the Java process.) Alternatively, if the socket-based TCP/binary or HTTP/SOAP communications channels are used, the Java side can run inside a separately-started standalone Java Virtual Machine (JVM), or in any application server that contains a J2EE servlet container. The .NET side can be a standalone application, an ASP.NET Web application, or a Web service. JNBridgePro supports communication between Java and .NET sides using a fast binary protocol based on .NET remoting or HTTP/SOAP. No change to the code is required to switch between communication protocols, only changes to configuration files.

JNBridgePro in the cloud

In addition to supporting .NET-Java interoperability in conventional on-premises applications, JNBridgePro can be used to support interoperability in cloud-based applications. For example, .NET and Java code can interoperate inside a process in a single cloud instance using the shared-memory communication mechanism, or they can interoperate in separate processes in the instance using the TCP/binary communications mechanism. Similarly, .NET code can reside in one cloud instance and can interoperate via TCP/binary with Java code in an instance in the same cloud (for example, Windows Azure or Amazon EC2), or the .NET and Java sides can be in different clouds (for example, one in Windows Azure and the other in Amazon EC2). Finally, JNBridgePro can be used to bridge between .NET code running in the cloud and Java code running in an on-premises machine (or vice versa).

Summary

JNBridgePro provides benefits to any organization or developer wishing to leverage an investment in existing Java code and libraries while pursuing new .NET initiatives. In addition, it can help development teams with both Java and .NET expertise to combine their skills in joint projects. It also allows development teams to produce Java and .NET versions of a product that share a common code base. JNBridgePro is simple to use, allowing developers to integrate .NET and Java classes as though Java were a .NET language. The .NET classes seamlessly interact with Java classes through proxies that expose the Java classes' functionality. Generation of proxies for the Java classes is done using a simple yet powerful graphical tool.

JNBridgePro offers a number of powerful and unique capabilities supporting Java/.NET interoperability:



- JNBridgePro enables .NET code to seamlessly access Java objects and methods, and vice-versa, without having any knowledge that the target objects are written for the other platform. Capabilities include the construction of objects, the passing and return of objects, throwing of exceptions, and inheritance.
- JNBridgePro allows Java UI elements to be freely embedded in .NET Windows Forms, and allows WinForms elements to be embedded in Java GUI applications.
- JNBridgePro provides a callback mechanism that allows .NET classes to be registered as Java listeners, and allows Java classes to be registered as .NET event handlers. No source code needs to be present to accomplish this; it is done solely with binaries.
- JNBridgePro can be used to interoperate with Java code from standalone .NET applications, or from .NET-enabled Microsoft products such as BizTalk Server, SharePoint Server, and Office applications like Excel, Word, and Outlook.
- JNBridgePro supports JDK 1.3.1 and later, and all versions of the .NET Framework. JNBridge is committed to support all future versions of Java and .NET as they are introduced.
- JNBridgePro allows Java and .NET code to communicate directly in the same process, and also supports the construction of distributed systems through both a high-speed binary protocol and SOAP.
- JNBridgePro supports interaction with all Java objects including: EJBs, JMS, and JNDI.
- JNBridgePro transparently and seamlessly integrates transactions on the .NET and Java sides, thereby allowing .NET and Java code to participate in the same distributed transaction.
- Applications using JNBridgePro can be deployed into the cloud using the same communications mechanisms, so that the Java and .NET sides can be in the same cloud instance (in the same or different processes), in different instances in the same cloud, in different clouds, or one side in the cloud and the other side in an on-premises application.

For more information about JNBridgePro, or to download a free full-featured trial version, visit the JNBridge website at www.jnbridge.com.



COPYRIGHT © 2002-2011 JNBridge, LLC. All rights reserved. JNBridge is a registered trademark and JNBridgePro and the JNBridge logo are trademarks of JNBridge, LLC. Java is a registered trademark of Sun Microsystems, Inc. in the United States and other countries. Microsoft, C#, Windows, and other Microsoft product names are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries. Other terms and product names may be trademarks or registered trademarks of their respective owners and are hereby acknowledged.

June 2011