INTRINSYC

WhitePaper

Highlighting Intrinsyc's Technologies:

Intrinsyc J-Integra

Bi-Directional Pure Java-COM Bridge

Intrinsyc Software, Inc.
700 West Pender Street, 10th Floor
Vancouver, British Columbia
Canada
V6C 1G8

Table of Contents

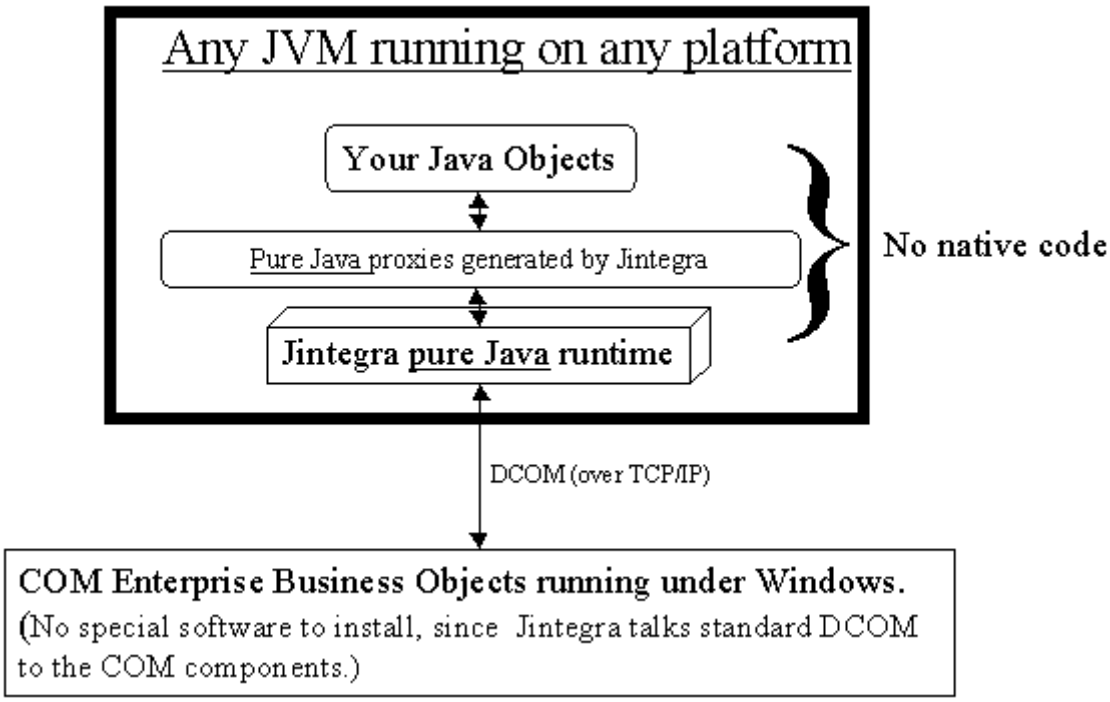
Chapter 1 - Introducing J-Integra

J-Integra is a COM-Java bridging tool. Using J-Integra, you can access ActiveX Components as though they were Java Objects, and you can access pure Java objects as though they were ActiveX Components.

J-Integra is used throughout the world, including in many European and US banks and financial institutions. If you work for a major bank, pharmaceutical company, IT company, or in telecommunications, chances are that someone in your company has purchased J-Integra licenses.

J-Integra's Java-COM bridging capabilities are infinitely flexible, allowing COM access to and from Java objects that are running on any operating system environment. The kit comes with step-by-step tutorial examples leading you through the most common usage scenarios.

- ***J-Integra works with any Java Virtual Machine, on any platform, and requires no native code (no DLLs).***



J-Integra's pure Java runtime talks to COM components using Distributed COM (DCOM) layered over Remote Procedure Calls (RPC), which are themselves layered on TCP/IP. So at the lowest level, J-Integra uses the totally standard Java networking classes.

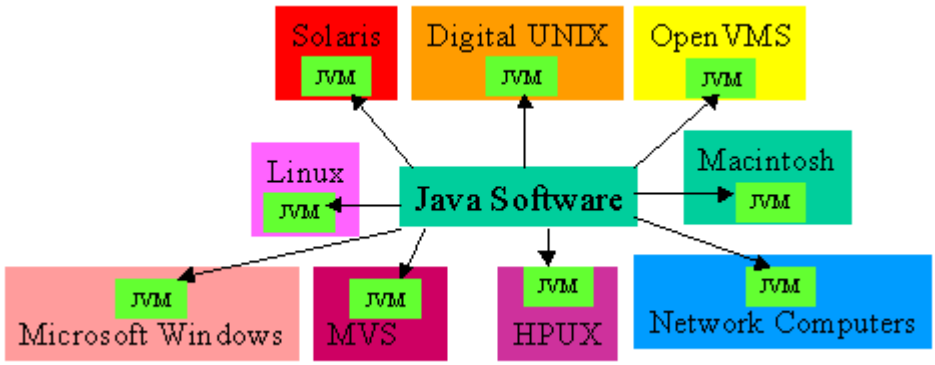
To the Java programmer, J-Integra makes COM components look just like Java objects, presenting COM properties, methods and events as Java properties, methods and events.

The J-Integra bridge is bi-directional. COM developers can make callbacks into Java objects. J-Integra dynamically "remote-enables" any Java object, making all of its public methods and member variables accessible from COM.

- ***J-Integra: No compromise bridging between pure Java and ActiveX.***

1.1 Java: Write Once, Run Anywhere.

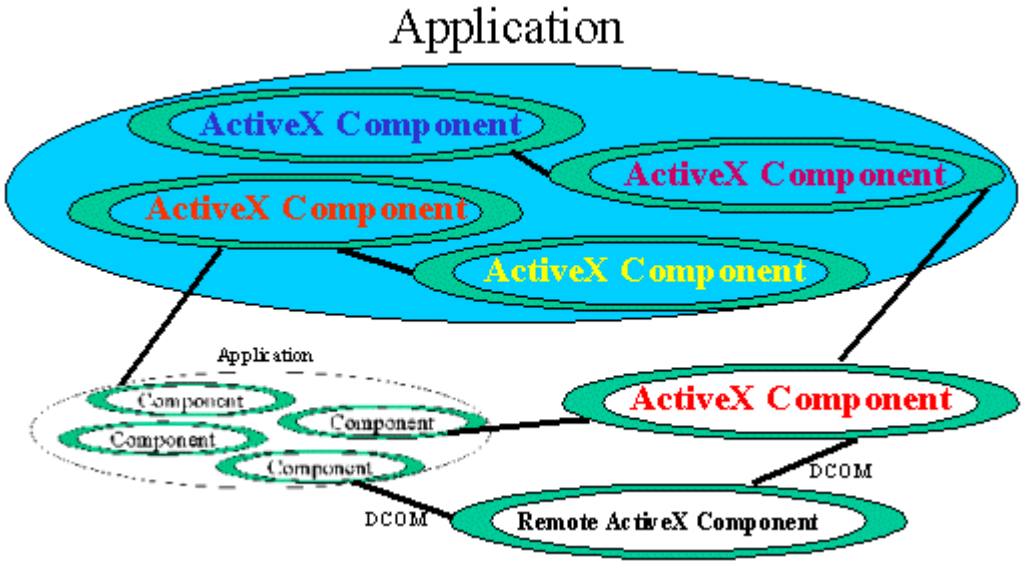
One of the reasons why so many companies are using Java is that it allows them to dramatically reduce the costs of software development and deployment, because Java programs can execute on any platform that supports a standard Java Virtual Machine (JVM).



The principle of developing "pure" Java software is a very important one, and many software developers are (justifiably) quite concerned about the use of any software which locks them into a specific platform.

1.2 ActiveX: Component Re-Use.

Software Components are a natural evolution of Object-oriented software development, enabling the isolation of parts of an application into separate components. Such components can be shared between applications, and since components are only accessed through a rigidly defined interface, their implementation can be changed without impacting applications that use them. Microsoft's widely used Component Object Model (COM) defines a binary standard for component integration, allowing COM components created using Visual BASIC (for example), to be accessed from an application created using Visual C++.
A massive number of ActiveX (COM) Components have been created and are available for purchase. Indeed, any software which has been recently developed under Microsoft Windows will almost certainly have been created using such components.



Modern software design under Microsoft Windows practically mandates that an application be designed using an ActiveX Component-based approach. One of the benefits of doing so is that parts of an application's functionality can be made accessible to other applications, not only on the same host, but also remotely using Distributed COM (DCOM).

1.3 The Conflict

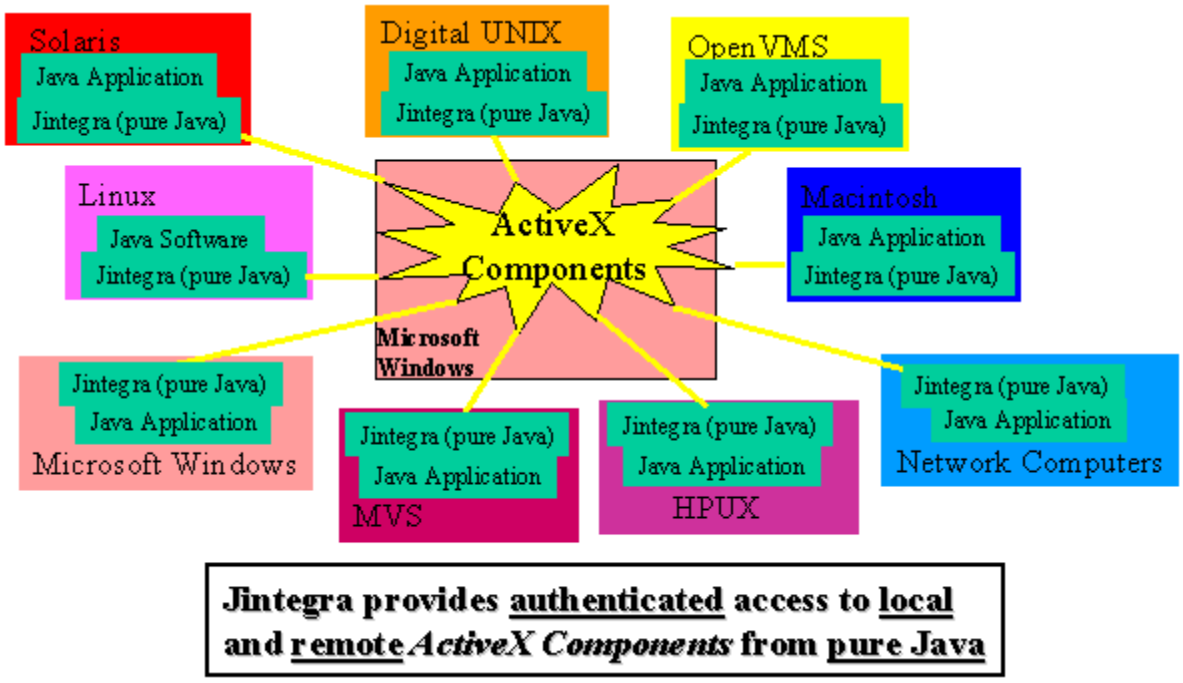
There may be conflict between the desire to create Java software which runs anywhere, and the need to re-use ActiveX software components.

This conflict often manifests itself in companies where there are two clearly divided camps -- the Java developers, who dislike anything which will limit their pure Java software to a specific platform, and the Windows developers, who are urging the use and re-use of ActiveX components.

Nevertheless, there is frequently a business need to access existing applications from new applications developed in pure Java, running in a standard JVM. Many of these existing applications were developed under Microsoft Windows, and because they were well designed, they expose their functionality to other applications by exposing ActiveX Components.

1.4 The Solution

- ***Write once, run anywhere, and benefit from ActiveX Component re-use***



J-Integra offers a solution. Internally, it uses a standard platform independent mechanism that Microsoft has defined for accessing ActiveX Components.

Chapter 2 - Technical information

2.1 DCOM-Based

Internally, J-Integra uses the Distributed COM network protocol to provide access to both local and remote ActiveX components from a pure Java environment. DCOM is an integral part of Windows NT and Windows 98, and is available for Windows 95 free from Microsoft.

This means that there is **no special software to install** on the machine hosting the ActiveX Component.

2.2 Access to Automation Components

J-Integra provides access to the most widely used form of ActiveX Component -- ActiveX Automation Component (as well as IDispatch derived 'dual' interfaces).

Almost all standard Microsoft products (such as Excel) expose Automation interfaces, as does most third party software.

When your software developers create ActiveX Components using tools such as Visual C++ and Visual BASIC, they will almost definitely be creating Automation Components.

2.3 Low Overhead

The J-Integra Java runtime (jintegra.jar) is approx 220K in size. J-Integra has been engineered using a pragmatic approach, bearing in mind that one of the uses for the product may be to provide Java Applet-based interfaces to existing ActiveX Component-based applications.

2.4 ActiveX Data Types are Totally Hidden

Although ActiveX Components deal with data types such as Variants, SAFEARRAYS, and OLE Dates, J-Integra completely hides the existence of such data types, dynamically mapping between ActiveX types and the most appropriate Java types.

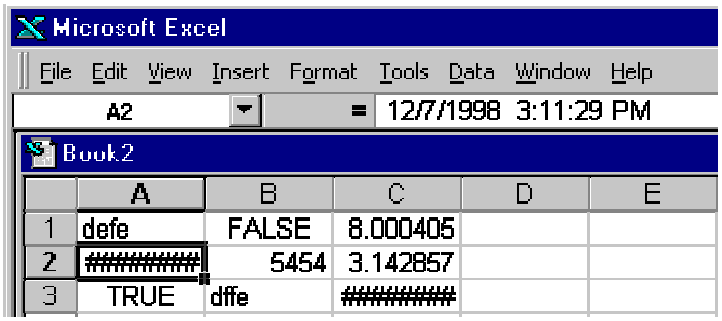
This small code excerpt from an Excel Example demonstrates this point. The *Range* ActiveX Component (part of Excel) has a *Value* property, which can be set to update the values in a range of spreadsheet cells.

In ActiveX terms, the Value property is a two dimensional SAFEARRAY of Variants. In Java, you simply create a two dimensional array of Objects, and populate it with standard Java objects (instances of java.util.Date, Float, Boolean, etc.). J-Integra does the rest:

```
            . . .
 Range range       = sheet.getRange("A1:C3", null);

 // New contents for the range -- notice the standard Java types
 Object[][] newValue = { { "defe",          new Boolean(false), new
Double(98765.0/12345.0)},
                  { new Date(),       new Integer(5454),  new Float(22.0/7.0)       },
                  { new Boolean(true), "dffe",          new Date()              } };

 range.setValue(newValue); // Update the spreadsheet
            . . .
```



2.5 Event Handling

If an ActiveX component can generate events (in COM terms, has a source interface, perhaps called *xyz*), J-Integra will allow Java objects to implement the event *xyz* event interface, and then subscribe to that event using the standard Java semantics of add*XYZ*Listener and remove*XYZ*Listener.

2.6 Bi-Directional

You may pass a reference to a Java object as a parameter to a method call on an ActiveX Component. That component may then make calls back into the Java object as though it were a standard ActiveX Component.

You can also create an instance of a Java object from Visual BASIC using a simple Moniker based mechanism:

```
Set anObject = GetObject("YourJvm:com.yourcompany.YourClass")
```

J-Integra supports both early binding and late binding, and includes examples of accessing Java objects from Visual BASIC (both late and early binding), from Visual C++, and from Excel VBA. J-Integra can also be used to access **Enterprise Java Beans** from COM.

2.7 Authentication

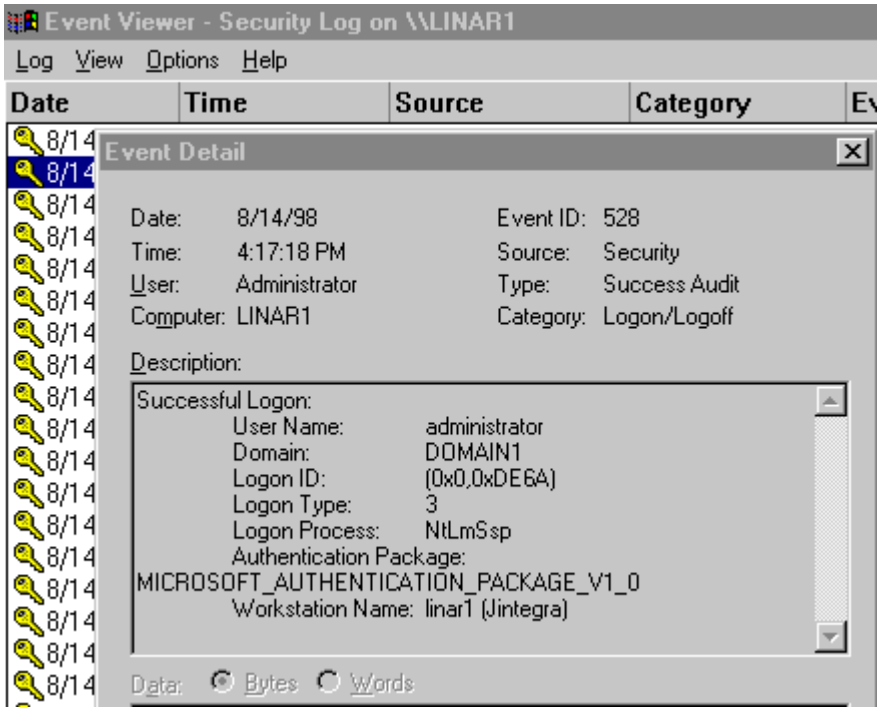
When accessing an ActiveX Component from Java using J-Integra, specify the domain, username, and password to be used to Authenticate access to the component:

```
        . . .
AuthInfo.setDefault("NT DOMAIN", "NT USER", "password");
excel.Application app = new excel.Application();
        . . .
```

If no host name is passed to the constructor, J-Integra assumes you wish to create a component on the local host.

J-Integra supports both NT Challenge-Response, and the legacy LanMan authentication (which can be disabled if not required). These authentication mechanisms do not cause passwords to be sent across the network.

By enabling Auditing of logons, it is possible to see that the user name and password have been verified through the standard Windows NT Authentication mechanism (in this case the computer running the Java software is called *linar1*):



Microsoft's standard DCOM configuration tool is used to specify exactly who may access a component.