

# CONNECTING EIB COMPONENTS TO DISTRIBUTED JAVA APPLICATIONS

Robert Ott\*, Heinrich Reiter\*\*

\* Professional JNet Solutions AG, Postfach,  
CH-9542 Muenchwilen TG, Switzerland,  
Phone: +41 71 960 08 10, Fax: +41 71 960 08 14, E-mail: ott@jnet.ch

\*\* Dr. Heinrich Reiter, EIB Association s.c.,  
Avenue de la Tanche, 5, B-1160 Brussels, Belgium,  
Phone: +32 2 6631443, Fax: +32 2 6755028, E-mail: hreiter@eiba.com

**Abstract** - This paper describes an architecture called 'EIBAgents for Java' that enables Java applications to access EIB (European Installation Bus) components. The architecture is held abstract to allow open and flexible extensions of the EIB Agent architecture. It is divided into three major Java base classes that are called EIBAgent, EIBReceiver and EIBEiblet. The EIBAgent is the actual interface that provides methods for accessing EIB group objects. EIBAgents can be implemented to access an EIB system via communication mechanisms such as serial communication (RS-232) or TCP/IP socket communication. EIBReceivers are described as request forwarder to EIBAgent implementations and can be used to forward requests across different EIBAgent servers. EIBEiblets are applet style pieces of Java code that implement logical behavior or graphical user interfaces for the EIB system. The paper also shows a sample of a distributed Java application that uses the EIB Agent architecture to access an EIB system from various platforms.

## 1. INTRODUCTION

The EIB (European Installation Bus) [1] has been a successful system to facilitate modern electrical installations for homes as well as commercial buildings. The standardization through the European Installation Bus Association (EIBA) allows today's interoperability between EIB components of different manufacturers.

On the other hand, the Java technology [2] and its unique and flexible architecture provides platform independent development and deployment of Java applications. Deployment of applications on systems such as Personal Computers, Unix servers, Settop-Boxes and even smart cards has never been easier as today using Java's portable byte code. Java applications cannot only run on almost all platforms, they can also easily communicate between these platforms.

However, why do today's EIB components not use Java as a technology to communicate to each other? Some critics claim that the architecture of these two systems is too different to bring them together. Others wait skeptically because the complexity of joining the two systems might be too high.

The architecture of the EIB Agents for Java shows a way how EIB and Java can be integrated seamlessly.

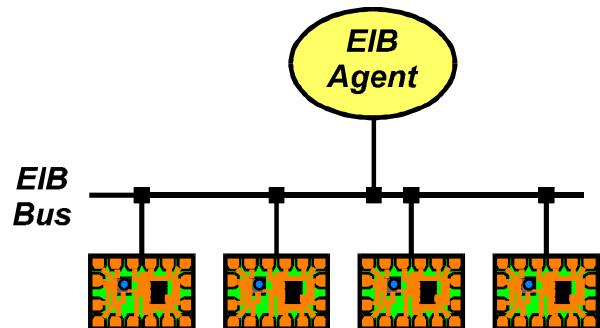


Figure 1: An EIBAgent connected to an EIB system.

The architecture can be applied with standard hardware components available on today's market. Furthermore, the architecture is flexible enough to bring some parts of the Java Agent components right into the EIB hardware components in the future.

## 2. THE EIB AGENT ARCHITECTURE

The architecture of the EIB Agent system can be divided into three major components. These components are the EIBAgent itself, an EIBEiblet component and an EIBReceiver component.

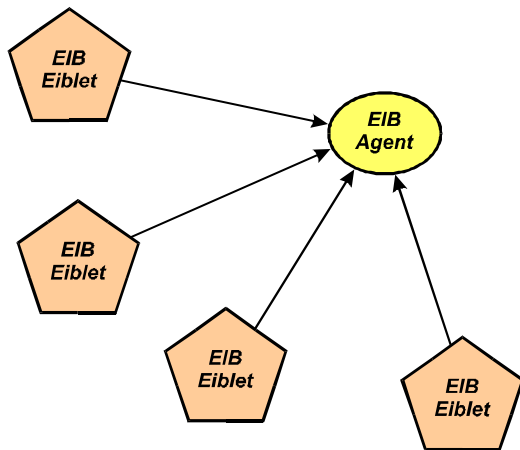


Figure 2: EIB Eiblets using an EIB Agent.

An EIBAgent is an abstract Java base class that defines how to access EIB group objects. This abstract base class can be implemented to access an EIB system directly using a serial interface or it can be implemented using additional hardware interfaces such as the EIB-ComClient from Schlaps & Partner [3] to access an EIB system via TCP/IP. Another implementation of an EIBAgent could also talk to other EIBAgents via a network using an EIBReceiver (see below). Figure 1 shows an EIBAgent object that is connected to an EIB system.

An EIBEiblet is a portion of Java code that uses an EIBAgent object to access EIB group objects. In Java terms, an EIBEiblet is a kind of applet that can be loaded into a running EIB Agent system. Figure 2 shows an EIBAgent to which several EIBEiblets are connected. Some EIBEiblets could implement some logic behaviors of an EIB system while others could be used as a graphical user interface for the EIB system.

EIBReceivers shown in Figure 3 allow the EIB Agent system to be distributed across platforms. Implementations of EIBReceivers could use socket communication, CORBA [4] service implementations, or even RS-232 serial communication (using the Java extension javax.comm [5]) to wait for EIBAgent commands. An EIBReceiver object always uses an EIBAgent component to access an EIB system.

The following section shows a prototype system that uses the EIB Agent architecture running some EIBEiblets within the system.

### 3. A DISTRIBUTED EIB-AGENT SYSTEM

The EIBAgent application shown in Figure 4 uses standard hardware components that are available on the market.

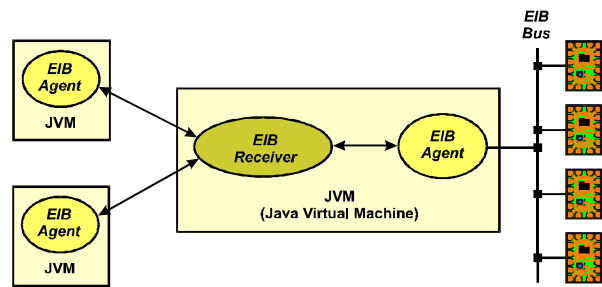


Figure 3: EIBReceiver for distributed EIBAgents.

### 3.3 The Communication to an EIB System

The interfaces to access an EIB system are currently very limited. The standard BCU-1 (Bus Coupling Unit; first generation) can be used to access an EIB system using serial communication (RS-232). However, the current specification of the serial interface of the BCU-1 defines to use 'special' hardware handshake mechanisms (RTS/CTS) for each byte transferred to the BCU-1. This specification requires very deep access to the serial controller of the operating system. That was easily achieved in operating systems such as DOS, Windows 3.1 and Windows 95. However, it is hard to implement the serial communication to the BCU-1 when the access to the hardware is limited on operating systems such as Windows NT or higher programming languages such as Java.

The manufacturer Schlaps & Partner from Germany provides a hardware product called EIB ComClient [3]. The EIB ComClient can be connected to an EIB system on one side and provides a TCP/IP interface to access EIB group objects on the other side. The communication to the EIB ComClient can be implemented using UDP (universal datagram packages) on IP. UDP style of network communication is also available in the core class library of Java (package java.net). The sample EIB Agent application uses the EIB ComClient to access an EIB system.

### 3.4 The EIBComClientAgent

### 3.5 The Communication to an EIB System

The abstract base class EIBAgent had to be specialized to use the EIB ComClient to access an EIB system. This class is called EIBComClientAgent. It uses UDP style of communication to talk to the EIB ComClient. The EIBComClientAgent implementation replicates the state of all available EIB group objects. As a result, the state of all EIB group objects can be accessed using this agent implementation. In addition, EIB group objects can be changed using the EIBComClientAgent. The agent then forwards such changes to the EIB ComClient using UPD commands which again forward

the changes to an EIB system. Figure 4 shows a EIBComClientAgent running on a Linux server [6].

### 3.3 The EIBServerSocketReceiver and the EIBSocketAgent

Now that the EIB group objects are available in the world of Java using the EIBComClientAgent, an EIBReceiver implementation is needed to distribute EIBAgents across platforms. The ServerSocket class of the standard Java package java.net is quite suitable to implement an EIBReceiver. We call this implementation EIBServerSocketReceiver.

The EIBServerSocketReceiver listens on a specified port and can handle multiple socket connections. The receiver forwards all requests to its EIBAgent, which is an EIBComClientAgent shown in Figure 4 (on the Linux server).

An EIBServerSocketReceiver is accessed via socket communication from another EIBAgent implementation, the EIBSocketAgent. The EIBSocketAgent implementation forwards all agent requests to the EIBServerSocketReceiver on the other side of the network, which again forwards the requests to its EIBAgent.

This EIBSocketAgent-EIBServerSocketReceiver pair can be used as a kind of request forwarder or request relays for EIBAgents.

### 3.4 The Server Side Eiblets

Figure 4 shows two server side eiblets. One is an eiblet for controlling light scenes called EIBLightControlEiblet and another eiblet controls the heating system and is called EIBHeatingEiblet.

Java is a multi-threaded environment by nature. As a result, eiblets can easily be active components using their own threads to act as control system for an EIB installation.

### 3.5 EIB Agents in Java Applets within Browsers

Java got its breakthrough not only because of its excellent conception and portability, the breakthrough came rather because of the ability to download Java applets into browsers and the easy deployment of these applets via the network. Even through there are still incompatibilities between the Java virtual machines within browsers of different vendors, the Java market has been grown enormously. Today, Java has also reached great success on the server side and in middleware.

Nevertheless, Java applets are also becoming more and more portable within browsers. Therefore Java can also

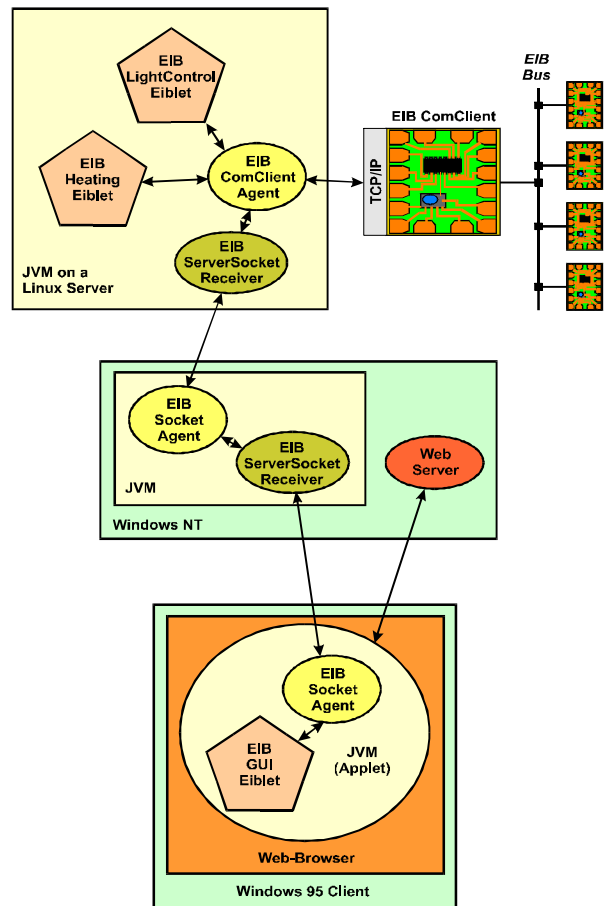


Figure 4: A Sample EIB Agent Application.

be an important means to provide graphical user interfaces for EIB installations.

Nevertheless, Java applets are also becoming more and more portable within browsers. Therefore Java can also be an important means to provide graphical user interfaces for EIB installations.

Figure 4 shows a Windows 95 client that runs an EIBAgent to provide a user interface eiblet. Ordinary Java applets within browsers are restricted by the security manager of the browser in which they are running. This restriction allows only socket connections to the server where an applet has been loaded from. The figure also shows that the applet within the browser uses a socket connection to the web server where the applet has been loaded from. The EIBSocketAgent and the EIBServerSocketReceiver on the Windows NT server provide a sort of gatekeeper functionality to leave the applet within its sandbox.

The sample EIBAgent application described in this section uses standard EIB hardware components available on today's market to access the EIB system. The next section will show how some parts of an EIBAgent application could be moved right into EIB hardware components.

#### 4. EIB AGENTS ON EIB HARDWARE COMPONENTS?

The previous section described a sample of a distributed application that uses the EIBAgent architecture. It is quite suitable for simple control tasks and for providing graphical user interfaces to control an EIB installation.

The major limitation of this sample application is, however, that some sort of server must always be up and running, except if the system is only used for graphical user interface purposes (in the sample application it is the Linux server).

The next step to extend the power of the EIBAgent architecture is to create a hardware component for the EIB system that runs a Java virtual machine. The EIBAgent system could then easily be put directly into this Java EIB component.

Figure 5 shows an EIB hardware component that runs a Java virtual machine. In contrast to the sample application described in the previous section, the actual server and logic eiblets of the application can be loaded directly into the EIBAgent hardware component. EIB applications could now easily be written using a standard Java development kit such as the JDK from Sun Microsystems or Symantec Visual Café [7]. It is even conceivable that eiblets move from one EIB component to another across the normal EIB twisted pair wires.

#### 5. FUNCTIONAL EXTENSIONS

The EIBAgent architecture basically only defines the abstract base classes with which the EIBAgent software components communicate to each other. The shown implementations such as EIBServerSocketReceiver, EIBSocketAgent, EIBComClientAgent are only some implementations that can be used to implement an EIBAgent system.

The architecture leaves open to implement additional EIBAgent and EIBReceiver classes that could use CORBA or other means to communicate. Additional implementations may regard secure socket layers (SSL) communication and authentication to limit the access to an EIB system and its group objects.

#### 6. SUMMARY

The EIBAgent architecture described in this article shows a way how Java can be used as a programming environment to access EIB hardware components. The architecture is held abstract to allow specialization of the partially abstract base classes of the EIBAgent system. Thus, additional communication styles between

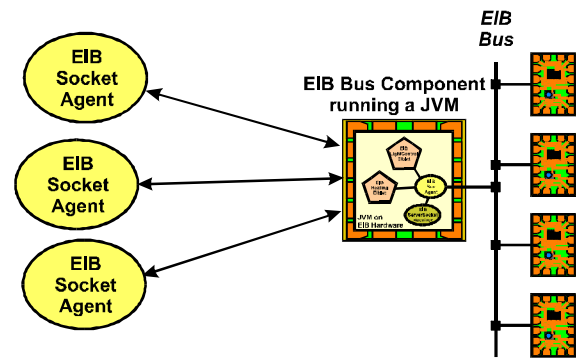


Figure 5: An EIB Agent running direct on EIB hardware.

the EIBAgent architecture can easily be implemented by subclassing.

The shown architecture can be divided into three major base classes that are EIBAgents, EIBReceivers and EIBeiblets. The EIBAgent and EIBReceiver base classes are used for communication purposes while EIBeiblets define a common way to implement logic components or graphical user interfaces for an EIB system.

The article also describes a sample application using the EIBAgent architecture to access EIB hardware components. The hardware used to exchange messages between Java and the EIB system of the shown example uses the EIB ComClient available from Schlaps & Partner in Germany. The sample also shows how the EIBAgent architecture can be applied for Web based applications that could run eiblets in Java applets within browsers. An implementation could also use Java servlets to provide an HTML interface to access EIB components.

#### REFERENCES

- [1] EIB Handbook, Rev. 2.21, EIB Association, Brussels, Belgium, 1998.
- [2] <http://java.sun.com/>
- [3] <http://www.schlaps-automation.de/>
- [4] <http://www.corba.org/>
- [5] <http://java.sun.com/products/javacomm>
- [6] <http://www.linux.org/>
- [7] <http://symantec.com/>