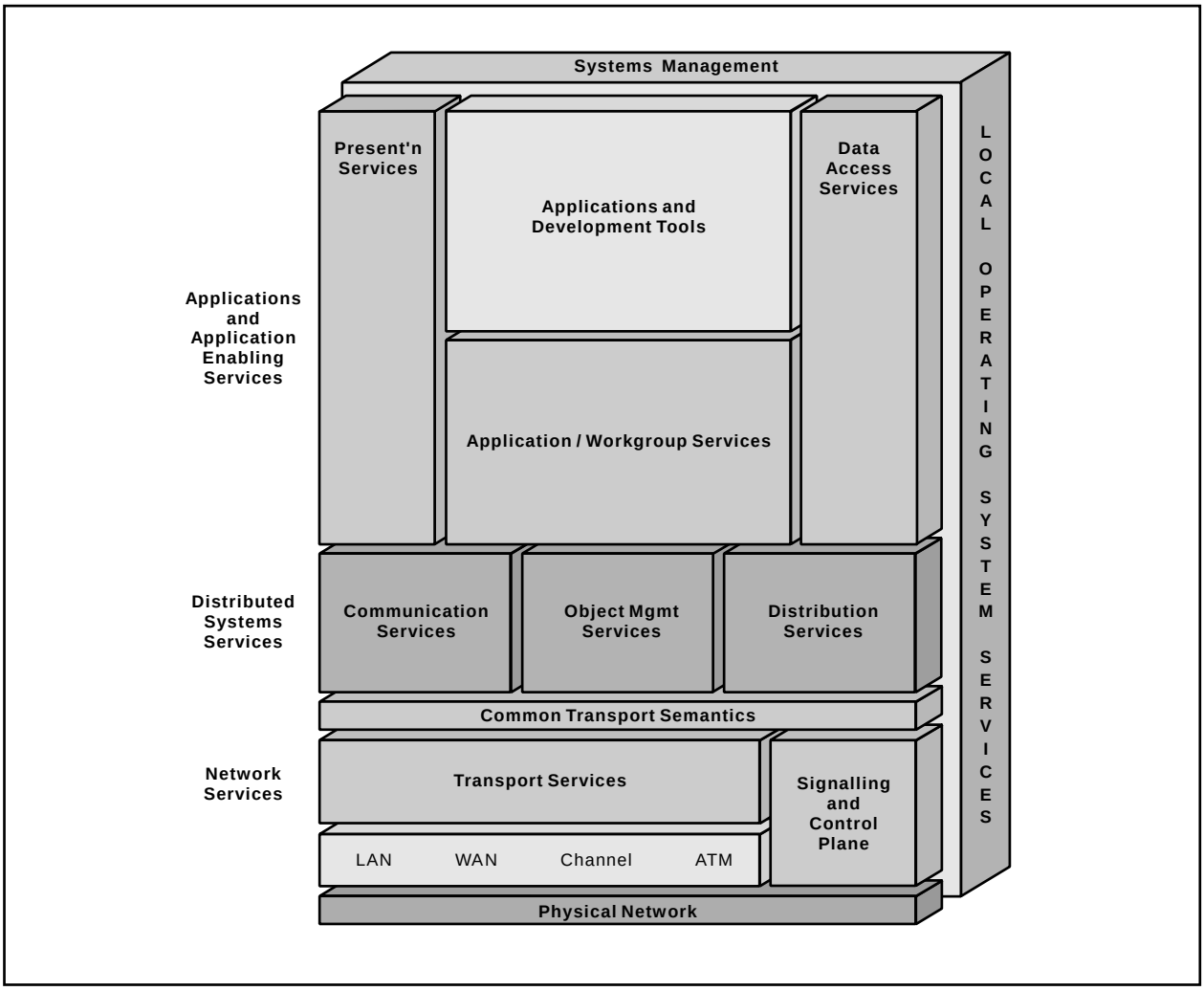
Open Blueprint

Virtual Machine Resource Manager



G325-6588-00

Open Blueprint

Virtual Machine Resource Manager

G325-6588-00

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Virtual Machine resource manager component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the Virtual Machine Resource Manager in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Open Blueprint Virtual Machine Resource Manager

The Virtual Machine resource manager implements the industry standard, object-oriented, Java execution environment defined by the JavaSoft division of Sun Microsystems Inc. It provides a platform-independent execution environment for Java programs such that the same binary programs will execute on all Java-compliant hardware and software configurations. The Java object model, instruction set, and standard interfaces are the same everywhere. The implementation is provided primarily by a mapping onto the other resource managers and services of an Open Blueprint distributed system.

Background: The technology on which Java is based began life in 1989 as software for consumer electronics. Java was released for public use in 1995 as Internet software that dramatically extends what a PC or Unix workstation can do while browsing the World Wide Web (WWW).

Java-enabled applications have more animation and "sizzle"; and they do some of their computation in the client system, for example checking input fields for valid data. The Web Browser resource manager is a remote viewing program; it uses the HyperText Transfer Protocol (HTTP) resource manager to access a Web server system across the Internet and displays a HyperText Markup Language (HTML) file or the results of executing an application on a server. Using Java applets[1] and the Virtual Machine resource manager, this passive role changes; the Web browser can *fetch* (access and download) applets and execute them in the local system as live, interactive elements within the current Web page.

The Java technology has evolved to become a binary-compatible cross-platform execution environment for network applications on commercial client and server systems. Developers can create applications and content with the same tools, in the same language, and using a consistent programming model on both client and server; the resulting binary programs will execute on any Java-compliant virtual machine.

The Virtual Machine resource manager is a program that emulates a computer. This virtual computer has its own set of instructions referred to as *bytecodes*. It also has libraries of predefined functions for some frequently used utility functions such as accessing the network, accessing files, and creating dialogs with the user using a mouse or a keyboard and windows.

The Virtual Machine resource manager design is:

- **Small and fast**: so that it fits in set-top boxes, personal digital assistants (PDAs), desktop and laptop PCs, and high-end parallel servers; the virtual machine is *scalable*.
- **Safe**: providing a base on which to build a secure environment that has a measure of protection against viruses and programming errors.
- **Portable**: so that efficient implementations can be developed for a wide variety of systems. There are implementations for every major client and server platform.
- **Object-oriented**: the virtual machine and its predefined functions consist of methods and data grouped into classes that are similar to, but simpler than, C++.
- **Compact**: Java programs are compact so that they travel quickly as they are fetched, just when needed, across the Internet.

Java is the name for JavaSoft's new language, which is designed to fit the object model and function of the virtual machine. Java is derived from C++ but it is much simpler and is designed to disallow the sort of programming that directly manipulates memory addresses using pointers. The Java language, like the virtual machine itself, is *type-safe*. The Java language was initially designed for ultra-reliable software to be run in consumer electronics devices. It is now evolving rapidly into a full application development language. The default compiler[3] for the Java language is written in Java and runs on the virtual machine. It's a good example of the less visible advantages of using Java. It has low development costs, because only one executable form of the program is used on all the different hardware and operating system

platforms. It becomes reliable faster, because more of us are using the one identical binary executable program.[2]

Main Features of the Virtual Machine Resource Manager

The two main features of the Virtual Machine resource manager are the interpreter and a set of class libraries.

- **The Interpreter**. Interprets the industry standard set of Java bytecodes. The interpreter also provides multitasking, exception handling, and the loading and execution of methods on objects. These mechanisms implement the Java object model, a dynamically linking, single-inheritance model where the objects, variables, and methods are all *typed,* that is, they are identified as belonging to specific Java classes. The typing is enforced at execution time.
- **Set of Classes** (libraries). Provides the Java-defined implementations of functions for:
  - A graphical user interface - graphics and windows
  - Bytestream file input and output
  - A communications interface based on IP sockets
  - Collections, events, math, and other utility classes
  - A range of other functions that are currently under industry development led by JavaSoft

The Virtual Machine resource manager provides documentation in HTML that is hyperlinked for easy navigation using a Web browser, and a Java development kit (JDK) that includes a compiler and debugging program.

For systems that are instances of the Open Blueprint, the implementations of Java classes comprise mappings to other Open Blueprint resource managers and services wherever practical. For example, the graphical user interface maps to the presentation service of the host operating system such as the OS/2 Presentation Manager or the Solaris Motif implementation.

The Virtual Machine resource manager is language independent; in addition to Java, languages such as NetRexx[4] target the Virtual Machine resource manager. However, because the Java object model is built into the Virtual Machine resource manager and type-safety is enforced at execution time, not all languages fit naturally. In particular, C++ and other languages using multiple-inheritance or exposing pointers are unlikely to map efficiently to the Virtual Machine resource manager.

The Virtual Machine: A Network Application Execution Environment

Network applications, whether written to the client/server or peer-to-peer programming models, are distinguished by a reliance on industry-standard TCP/IP network connectivity. They typically implement an interaction that crosses administrative, security, and enterprise boundaries.

The Virtual Machine resource manager emphasizes integrated support for networking. It anticipates use of World Wide Web technology through the use of the HTTP resource manager and HTML. The Virtual Machine resource manager implements a distribution paradigm where the program to be executed is generally fetched on demand from a server elsewhere on the network.

Programs that are retrieved from servers elsewhere on the network can be less trustworthy than programs installed ahead of time under the supervision of enterprise software distribution processes. The Virtual Machine resource manager incorporates safety features, beginning with type-safety enforcement, to enhance the protection of the virtual machine and its underlying system under these conditions.

Supported Application Characteristics

The application characteristics that benefit from the Virtual Machine resource manager are described below:

- The application consists of *live content* (a program running on the client that is animated and interactive) as part of a Web page. This is generally referred to as a *Java applet* and is invoked through the HTML APPLET tag. The APPLET tag is processed by the Web browser; the applet is executed when the Web page is displayed.
- The application runs on many different sorts of clients, both personal computers and workstations. For example, client programs that access online services can, if written for the virtual machine, be compiled to a single binary program that can be executed on all the client systems. Such a program can provide a distinctive or application-specific user interface on all those clients, and perhaps provide continuing user dialog when the network is congested or unavailable.
- The application is distributed; different pieces run on multiple, different types of clients, or split between client and server. The same tools, languages, and binary program can be run on any client or server that has the Java-compliant bytecode interpreter and Java classes. The Java-to-Java distributed method call, *Remote Method Invocation* (RMI), is provided for invoking methods on a Java object running on a server across the network. Alternatively, by using Java Interface Definition Language (IDL) to communicate between distributed objects that use an industry standard Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) object request broker and its Internet Inter-ORB Protocol (IIOP), Java programs can invoke the services of other CORBA objects including other Java programs.

  Distribution techniques can effect economies in development by avoiding the need for different source and binary programs for different hardware and software platforms. Distribution techniques can decrease administrative costs by reusing the same binary program on all platforms. For Java programs downloaded just-in-time from server to client, these distribution techniques can reduce the cost of administering software installation and upgrade. The robustness of the application can improve more rapidly, because all users are exercising the same binary program; bugs are found more quickly.
- The application runs on a server, either as a user-supplied subroutine extending the function of a Web server (such as a CGI-BIN program), or as a Java application in its own right. Using the Virtual Machine resource manager on a server combines the productivity of simple, powerful languages like Java and NetRexx, (comparable with PERL or BASIC), with the higher performance and more complete set of classes defined for Java.
- The application uses distributed resources on the network. In this case if the distributed resources are accessed through the use of a CORBA Object Request Broker (ORB), then Java provides the IDL and classes for that.

Virtual Machine Functional Description

This section describes the interpreter and lists the supported Java object classes.

The Interpreter

The interpreter executes Java instructions that are known as *bytecodes*. The computer emulated by the Java interpreter is stack-based, multithreaded, and object-oriented. It implements an object model with single inheritance and exception handling similar to C++. A just-in-time (JIT) compiler can be used to enhance the execution time performance of the interpreted bytecodes. The JIT compiler is invoked once for each method in an application, the first time the method is used. It converts the bytecodes for the method into the native machine code of the executing system. As methods are repeatedly used during the execution of an application, overall performance can be significantly enhanced. The native machine code is discarded at the end of the execution. Because the compilation is done on and for the specific system and the results are later discarded, there is no impact on platform independence.

The Java Classes

The Java Virtual Machine resource manager contains three types of classes:

- **Core classes**, which are guaranteed to be in all virtual machine resource managers.
- **Standard extension classes**, which are not always present, but when present are accessible through a standard, predefined API.
- **User classes**, which are all other classes that, by definition, do not implement a JavaSoft endorsed Java API.

The list of core and standard extension classes will be extended over time; the classes described here have already been shipped, or are projected to ship before the end of 1997.

Java Basic Utility Classes

Java basic utility classes are core classes that were made available in the first product release of Java in early 1996. These include:

- Utility classes for typical operating system functions such as collections, exceptions, and file I/O.
- A set of classes providing sockets-like transport on top of TCP/IP.
- Presentation classes for containers, windows, graphics, and image. These classes are known as the abstract windowing toolkit (AWT).
- Classes for the implementation of applets.
- Properties - persistent global values.

Java Enterprise Access Classes

Java enterprise access classes are core classes for:

- **JDBC**. Provides Java database connectivity (JDBC). JDBC is analogous to Open Data Base Connectivity (ODBC), and it provides an interface that is supported by the Relational Database resource manager.
- **Java IDL**. Supports communications between a Java object and a CORBA object using an Object Request Broker (ORB) and the Object Management Group (OMG) language-neutral interface specification.

- **RMI and Object Serialization:**. Supports remote method Invocation (RMI) between peers, or between client and server, when both participants are Java applets or applications.

Java Beans (Software Components) Expected in 1997

Java Beans classes will be part of the core classes, providing portable, platform-neutral application programming interfaces (APIs) for software components. Java Beans defines a component model that describes how to implement software in small, reusable chunks. These chunks, knows as beans, can be combined with other beans to form applications. Components can also be stored for later use - Java Beans classes provide component persistence using the Open Blueprint Data Access Services.

Java Media Classes Expected in 1997

Core classes for media consist of: clocks, 2D, audio, and animation. Timing services (clocks) for synchronization and media players are the foundation for the other time-based media classes. Audio classes manage multiple audio streams and support a range of audio types including WAV and AU data formats. Animation classes provide ways to add motion to visual elements such as text, graphics, and images. The 2D graphics model and rendering services display and print color images from graphical descriptions.

The media standard extension classes consist of: video and MIDI multimedia; team use of Java objects through the Java Share collaboration services which enable team use of Java objects; telephony call control, and 3D modeling and rendering.

Java Security Expected in 1997

Java security core classes are used to provide digital signatures, encryption, and authentication in user-developed applications and applets.

Java Management

Java management standard extension classes provide services for building applets that can manage an enterprise network over the Internet or intranets.

Java System Integration Interfaces

Four core interfaces provide system-independent access to and from the Virtual Machine resource manager:

- **Virtual Machine Invocation**. Defines the way instances of the Virtual Machine resource manager are created and shut down. It provides a way for other resource managers and applications to create and access their own instance of the Virtual Machine resource manager.
- **Native Method Invocation**. Methods that execute outside the virtual machine. Usually these are object methods written in C++ or C. These APIs define how to call out and call back between a Java applet or application running in the Virtual Machine resource manager and other objects in the same process.
- **Debugging Tool Interfaces**. Provide ways for development tools to access running Java applications and applets. By using these APIs, development tools written in Java can be deployed on all implementations of the Virtual Machine resource manager.
- **Just-in-Time Compilation Hooks**. Defines the way the just-in-time compiler is invoked.

Virtual Machine Resource Manager Evolution

The Virtual Machine resource manager will continue to evolve rapidly, because the value of Java technology is based on platform independence, reducing system administration, and being a preferred environment for innovation on the World Wide Web. These are not static attributes; they require, and JavaSoft anticipates, a sustained agenda for creating and enriching standard extension classes, and for pushing additional functions into the core classes. In the future, new core functions will typically first appear as standard extension classes.

Relationships to Other Resource Managers and Services

The Virtual Machine resource manager uses the services of other Open Blueprint resource managers. Java applications and applets access these services through Java's platform-independent and binary-compatible APIs. The Virtual Machine resource manager enables Java applications to use:

- **Presentation Services**. The Java AWT classes map to the underlying presentation services of the host system and use the *look and feel* native to that system. The Virtual Machine resource manager does not have a system-independent look and feel; it adopts the look and feel of the host presentation services.
- **Data Access Services**. JDBC, the file I/O core classes, properties, and Java Beans persistence component services map to the Open Blueprint Data Access Services.
- **Transport Services**. The Virtual Machine resource manager provides a sockets-like network interface for basic communications, assuming that applets and applications communicate over the Internet. The Open Blueprint Transport Services can provide multiple physical transports that underpin the sockets API; the Virtual Machine resource manager maps its network classes onto that sockets interface.
- **Object Management Services**. Java programs can communicate with non-Java programs by using Java IDL definitions supported by the Object Request Broker (ORB) and its IIOP protocols.

Application Development Support

Applications and applets can be developed for the Virtual Machine resource manager with integrated development environment (IDE), and rapid application development (RAD) tools that are similar to, and in many cases based on, the familiar tools already in use for development of other applications.

Industry leaders integrate their development tools so that an application can benefit from compositions of existing platform-specific code and new Java code; so that significant composite applications can be crafted from Java components (Java Beans) and other components, in particular: LotusScript Extension (LSX) components from Lotus, ActiveX components from Microsoft, and OpenDoc components. The Java Beans APIs are designed to allow this hybrid composition; the system integration interfaces ensure that the beans interoperate in a platform-neutral manner.

---

1 Applets are Java programs written to run as macros or extensions to a Java-enabled Web browser.

2 The Java bytecode definition, the Java language definition, APIs, semantics, and applications and applets are documented in books published by Sun Microsystems.

3 The compiler does not execute at run time; rather, it is used at build time to prepare the Java applets or applications for later downloading and execution.

4 NetRexx is a derivative of the Rexx language that is optimized for use with the Virtual Machine resource manager. For additional information about NetRexx, visit the URL: http://mfc.hursley.ibm.com/netrexx/netrexx.htm

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM
IBMLink
Open Blueprint
OS/2
Presentation Manager

The following terms are trademarks of other companies:

ActiveX	Microsoft Corporation
C++	American Telephone and Telegraph Company, Incorporated
CORBA	Object Management Group, Incorporated
Java	Sun Microsystems, Incorporated
Lotus	Lotus Development Corporation
Microsoft	Microsoft Corporation
Motif	Open Software Foundation, Incorporated
OpenDoc	Apple Computer, Incorporated
Solaris	Sun Microsystems, Incorporated
Sun Microsystems	Sun Microsystems, Incorporated

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft is a registered trademark of Microsoft Corporation.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the U.S. and other countries.

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

G325-6588-00