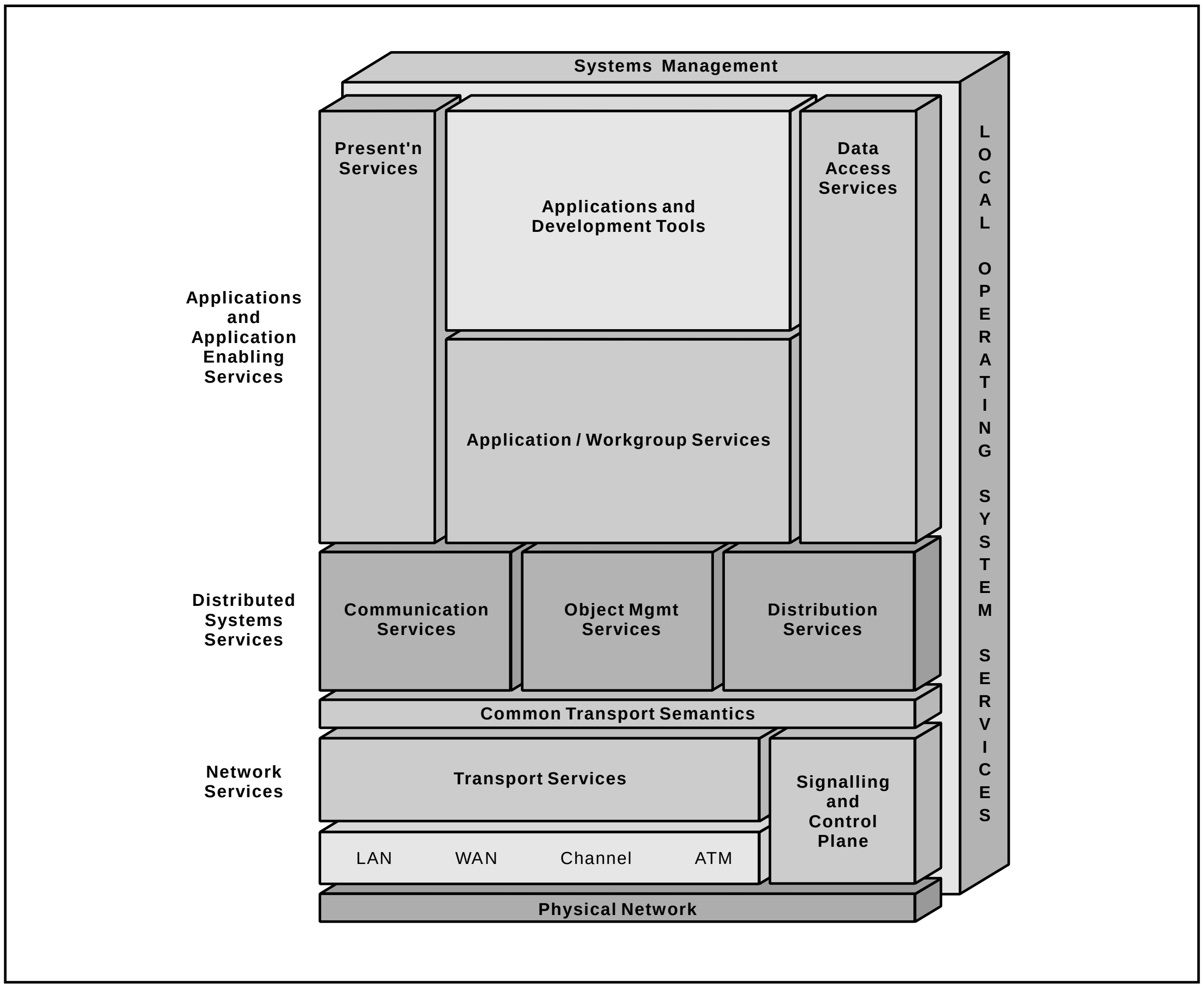
Open Blueprint

Object Request Broker Resource Manager



Systems Management
Present'n Services
Applications and Development Tools
Data Access Services
LOCAL OPERATING SYSTEM SERVICES
Applications and Application Enabling Services
Application / Workgroup Services
Distributed Systems Services
Communication Services
Object Mgmt Services
Distribution Services
Common Transport Semantics
Network Services
Transport Services
Signalling and Control Plane
LAN
WAN
Channel
ATM
Physical Network

GC28-1213-01

Open Blueprint

IBM

Object Request Broker Resource Manager

GC28-1213-01

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 Object Request Broker 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 Object Request Broker 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

Figures

Summary of Changes

The focus of this paper moved from explaining and positioning object-oriented technology to functional specifics of the Object Request Broker in the Open Blueprint structure.

The paper focuses on the distributed object environment and the support of the Object Management Group (OMG) CORBA standards rather than on intra-process System Object Model support.

Object Request Broker Resource Manager

This paper provides descriptions of the functions in the Open Blueprint Object Request Broker resource manager. It also discusses how these services relate to other components in the Open Blueprint structure, and how the Object Request Broker relates to industry standards.

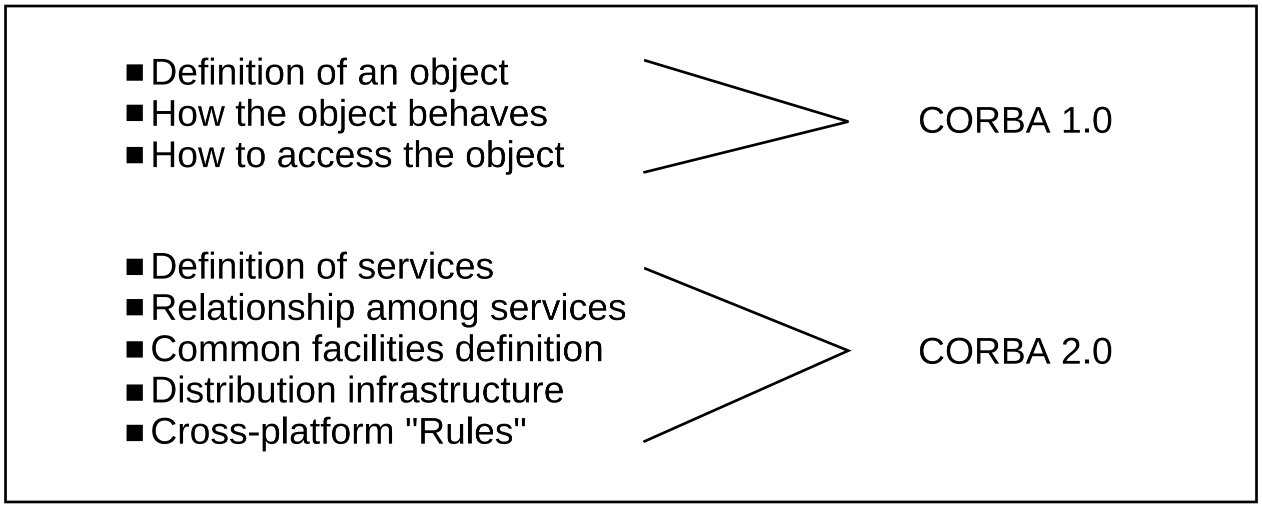
The Object Request Broker resource manager provides distributed object support functions that are available to both Open Blueprint resource managers and applications in the Open Blueprint environment. The Object Request Broker (ORB) resource manager is part of the Open Blueprint Object Management Services which supports basic object services and the System Object Model (SOM).

Overview of the Object Request Broker

Communications between objects can take place within a process (address space), between two processes in the same system, or between two processes running in separate systems. Objects in different processes use the ORB to communicate. Objects written in different languages that are executing within a process use SOM.

The ORB resource manager (also known as Distributed SOM) supports inter-process object communications. The ORB enables object distribution by brokering method invocations to remote objects across processes or machine boundaries. The ORB supports a language-neutral object model, which enables object method invocations across different language environments. The ORB also enables release-to-release binary compatibility so that parts of an application can be updated independently. Using the ORB, objects can be accessed independent of location, across processes, or across distributed systems.

The ORB resource manager is an implementation of the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA), which is published by X/Open. CORBA consists of two standards, CORBA 1.0 and CORBA 2.0, which together define the object and its behavior and the local and distributed object environment. CORBA 2.0 does not replace CORBA 1.0; it expands the scope of the standard into the distributed computing environment. (The contents of CORBA 1.0 and CORBA 2.0 are illustrated in Figure 1 below.)



■ Definition of an object
■ How the object behaves
■ How to access the object
CORBA 1.0
■ Definition of services
■ Relationship among services
■ Common facilities definition
■ Distribution infrastructure
■ Cross-platform "Rules"
CORBA 2.0

Figure 1. OMG CORBA Standards

CORBA 2.0 defines Common Object Services and Common Facilities, which are required to support objects in a distributed environment. These services define the infrastructure needed to successfully implement distributed object environments and to enable location and platform independence as objects are deployed across networks.

The ORB uses many Open Blueprint services including Security Services, the Directory resource manager, and Communications Services. CORBA Common Services are performed by one or more of the Open Blueprint resource managers, or by the Object Request Broker basic object services. A further discussion of the relationships of the Open Blueprint services and the basic object services appears in Sections 3 and 4.

System Object Model (SOM)

SOM is based on the CORBA 1.0 standard and is used as an object request broker for objects within the same process. SOM supports a language-neutral object model, which enables objects developed using different source languages to communicate at runtime across the different language environments. SOM supports C++, Java and SmallTalk; other languages will be added in the future. SOM also enables release-to-release binary compatibility for separately developed objects within a process. SOM provides the language, release, and location independence (within a process) required of a local, CORBA-compliant ORB.

Basic Object Services

The Object Request Broker Basic Object Services provide functions that enable object-oriented (OO) application development in the Open Blueprint environment. These services are also used by the ORB runtime and are considered fundamental services in the ORB environment. Because of the pervasive use of these services, they are provided as part of Object Management Services and are discussed in this paper with other ORB function.

The Object Request Broker (ORB)

The ORB enables application programs to access objects in other processes, even on different machines. Both the location and implementation of an object are hidden from a client, and the client accesses the object (using method calls) in the same manner regardless of the object's location.

The ORB:

- Uses the standard CORBA Interface Definition Language (IDL) Compiler, Interface Repository, and language bindings.
- Allows an application program to access a combination of local and remote objects. The location of an object is transparent to the program.
- Supports inter-process communication within the same system and remote transport across systems.
- Uses SOM inter-language interoperability capabilities to call objects written in different languages.

Figure 2 below illustrates the ORB resource manager's implementation of the CORBA architecture. The following sections provide an overview of both client and server functions.

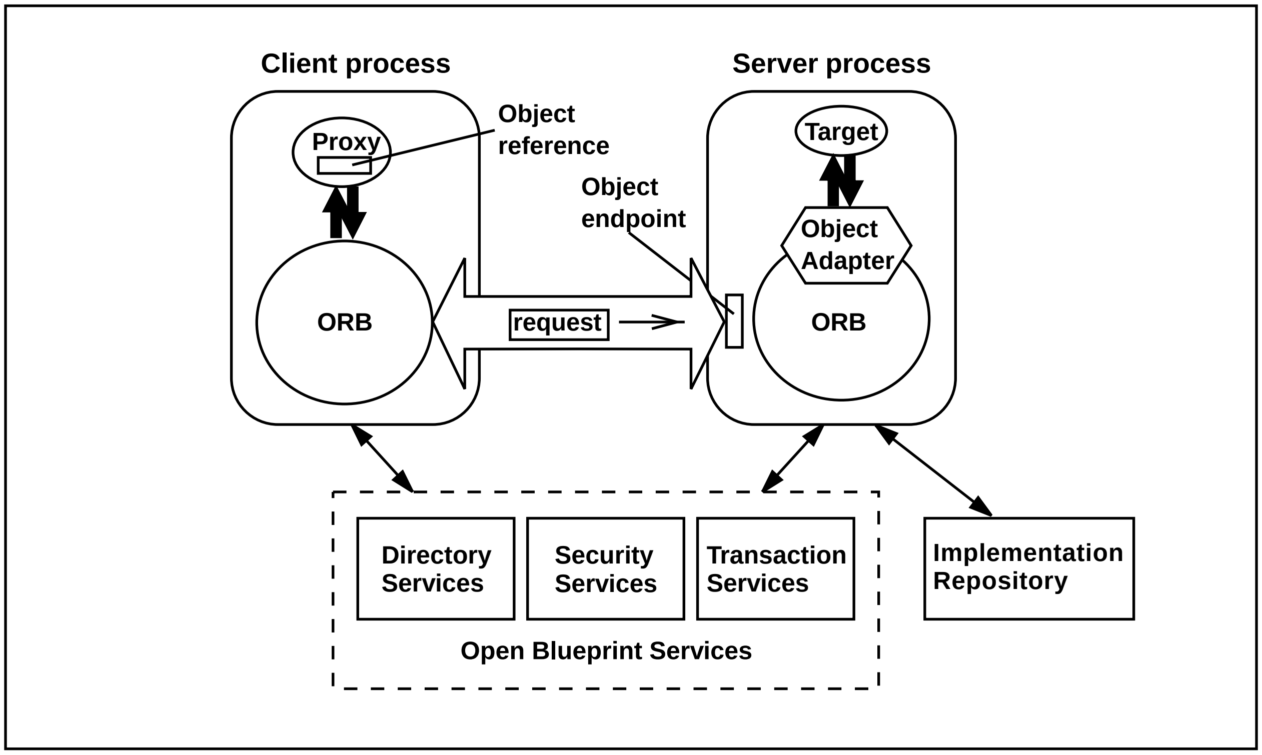


Figure 2. Object Request Broker High-Level Architecture

Client Overview

ORB clients can invoke methods on remote objects through local proxy objects. The ORB runtime finds or creates an instance of the proxy whenever a reference is made to a remote target object. To find the object reference (which is a machine-generated object ID) of a remote object, an application can query the Open Blueprint Directory resource manager with a user-defined object name.

To forward a request to a remote object, the proxy must:

- Obtain a binding to a communications endpoint that is associated with the remote object. The communications endpoint information is contained in the object reference. If the object reference is known, the communications endpoint information is extracted. If only the object name is known, the Open Blueprint Directory resource manager is used to match the name to an object reference.
- Compose a message to represent the request (marshaling)
- Send the request message to the server
- Receive the response message from the server
- Decompose a message (demarshal) and return results to the caller

If the remote object is under security protection, the credential provided by the client must be authenticated before a request can be sent. The ORB runtime calls the Security resource manager to obtain the user credential, and the Security resource manager returns the credential to the ORB runtime. The ORB sends the credential together with the request message to the server for authentication.

The ORB also uses the Open Blueprint Transaction Manager resource manager. Before the ORB sends a request, it determines whether the client is currently participating in a transaction. If it is, the ORB runtime calls the Open Blueprint Transaction Manager resource manager to get a *transaction context* and sends this transaction context to the server with the request message.

Server Overview

Each remote object is associated with a server. Each server has one or more communications endpoints that receive messages from clients on behalf of objects in that server. Each server contains an Object Adapter (OA) that serves as the link between the client and a remote object. The OA accepts request messages from an endpoint, performs necessary demarshaling, locates or activates the target object in the server, and dispatches the request to the target object.

To activate an object, the OA consults the Implementation Repository (IR), which contains information about each object in the network. Information needed about the objects includes the object implementation such as dynamic load library (DLL) name, process structure, and so on. (The Implementation Repository is a data store that is only needed by the server.) To receive and dispatch each request, the OA must do the following:

- Obtain the request message
- Demarshal the request
- Pass the request to the target object using SOM inter-language capabilities if the target object is implemented in a different language environment
- Marshall the response
- Send the response message to the client

After receiving a message, the ORB runtime passes the security credential it received from the client to the Security resource manager for authentication. The ORB then sets up the *principal name* (which is the user ID provided by the client) in the server so it can be used for authorization checks.

After receiving a message with an attached transaction context, the ORB runtime calls the object transaction service, an object oriented interface to the Open Blueprint Transaction Manager resource manager. The transaction context notifies the object transaction service that the client is operating within a unit of work and that the server is participating in that unit of work. When the response is ready for return, the ORB server obtains a return transaction context from the object transaction service and sends this transaction context back to the client. The ORB client runtime uses the return transaction context to notify the object transaction service that the server is participating in the same unit of work. At transaction commit time, the two transaction managers have enough information to communicate with each other to coordinate transaction commit or rollback.

ORB Use of Open Blueprint Communications

The CORBA 2.0 standard Internet Inter-ORB Protocol (IIOP) is a TCP/IP-based protocol. The ORB implements IIOP through Open Blueprint Network Services using Common Transport Services (CTS), enabling the ORB resource manager to control the transport protocols supported by CTS. The supported transports include TCP/IP, NetBIOS, IPX and SNA.

Synchronous and Asynchronous Method Calls

The ORB supports synchronous and asynchronous method calls. Synchronous method calls are used to:

- Send the request
- Wait for the reply
- Process the reply

Synchronous method calls are similar to remote procedure calls. Synchronous methods are implemented by using the Open Blueprint Common Transport Semantics (CTS), the Conversational services, and the Remote Procedure Call (RPC) resource managers.

Asynchronous method calls are used to:

- Put a request object on a queue for transport
- Obtain a response from a queue and process the response

The asynchronous calls are implemented using the Open Blueprint Messaging and Queueing resource manager.

Basic Object Services

Included in the Open Blueprint Object Management Services is a set of basic object services. These object services are essential for writing distributed object applications. The basic object services included are *life cycle service, externalization service*, and *collections service*. All basic object services support the OMG CORBA Common Object Services specification standard.

Life Cycle Service

The life cycle service defines object creation, deletion, move and copy .

- Object creation defines how objects are created and how instances of objects and object references are created. It incorporates the concept of a factory object which can create new objects.
- Object deletion deletes an object.
- Object copy creates a new instance of the same object.
- Object move changes the location of an object.

The life cycle service is essential for creating objects remotely and moving objects to a different location.

Externalization Service

The externalization service defines a standard method for externalizing and internalizing objects. Externalization and internalization are used to package and transport objects. For example, objects can be stored in a database until needed by a specific runtime environment. When stored in the database, the object is flattened, or transformed, into a database format (*externalization*). When retrieved from the database, the object is transformed into a runtime object (*internalization*).

Collections Service

The collections service supports the grouping of objects and the manipulation of objects as a group. It provides a uniform way to generically create and manipulate the most common collections such as queues, sets, bags, maps, and so on.

In addition to the object services above, the Open Blueprint Basic Object Services also support Object Identity. Object Identity is a service used to determine whether two objects are actually the same object.

Relationship to Other Open Blueprint Resource Managers

The ORB resource manager uses Open Blueprint resource managers to carry out many of the CORBA-defined object services. These services include:

Conversational Resource Manager: The ORB runs in a conversational environment using the services provided by the Conversational resource manager, which enables SNA addressing, security and other services to be used to optimize the SNA environment.

Messaging and Queuing Resource Manager: The Messaging and Queueing resource manager in the Open Blueprint provides decoupled client/server interactions and reliable message delivery semantics. The ORB uses the Messaging and Queuing resource manager to provide asynchronous method calls. This allows the user to take advantage of both the distributed functions of the ORB and the flexible delivery features of the Messaging and Queuing resource manager.

Directory Resource Manager: The ORB utilizes the Open Blueprint Directory resource manager services to locate servers and object instances.

Security Resource Manager: The ORB is a user of Open Blueprint Security services to support authentication and the movement of security credentials between clients and servers.

Transaction Manager Resource Manager: The ORB interfaces with the Transaction Manager resource manager to support transaction processing semantics. The Open Blueprint Transaction resource manager uses the ORB resource manager to deliver its transaction contexts. The ORB calls the Open Blueprint Object Transaction Service which in turn uses the Transaction Manager resource manager to obtain the context and to ensure transaction coordination.

Virtual Machine Resource Manager: The IIOP protocol is used to support object messaging between Java objects executing in the Virtual Machine resource manager and objects executing outside the Virtual Machine resource manager.

See the component description papers for each of the above resource managers for additional information.

Standards Support

The Open Blueprint Object Request Broker resource manager supports industry standards for distributed objects defined by the Object Management Group (OMG) and published by X/Open.

- The ORB supports OMG CORBA Interface Definition Language standards
- SOM supports OMG standard language bindings such as C++ bindings and Java bindings
- The ORB complies with the CORBA 1.0 and 2.0 standards, including:
  - IIOP and DCE-CIOP
  - Interface Repository
  - Object services

ORB Future Directions

Use of the Remote Procedure Call (RPC) resource manager is a future objective. CORBA 2.0 currently defines an optional Distributed Computing Environment (DCE) Common Inter-ORB Protocol (CIOP) for the DCE environment. This definition enables the DCE RPC to be used as the transport. There are several advantages to using a DCE RPC-based protocol from the ORB resource manager:

- The ORB can provide multiple levels of message security by using functions in RPC services such as message integrity, message signatures and message encryption.
- The ORB can exploit the error detection and recovery built into RPC.
- The ORB can interoperate with other ORB implementations with full naming and security interoperability.

Appendix A. Bibliography

Object Management Group CORBA Specifications

The Common Object Request Broker: Architecture and Specifications (CORBA 1.1). December, 1991

The Common Object Request Broker: Architecture and Specifications (CORBA 2.0). December, 1995 revision.

CORBA Object Services Specifications

Concurrency Service. Final submission August 5, 1994.

Event Notification Service. Final submission July 3, 1993. Updated January 1, 1994.

Externalization Service. Final submission September 9, 1994.

Lifecycle Service. Final submission July 4, 1993.

Naming Service. Final submission May 2, 1993.

Object Query Service. Final submission January 1, 1994.

Object Transaction Service. Final submission August 4, 1994.

Persistent Object Service. Final submission October 7, 1994

Relationship Service. Final submission May 5, 1994.

Time Service. November 8, 1995.

Security Service. Final submission December 1, 1995.

Appendix B. 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
SOM
System Object Model

The following terms are trademarks or service marks of other companies:

C++	American Telephone and Telegraph Company, Incorporated
CORBA	Object Management Group, Incorporated
DCE	The Open Software Foundation
Java	Sun Microsystems, Incorporated
IPX	Novell, Incorporated

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

Appendix C. 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.

GC28-1213-01