



Gatekeeper Guide



VERSION 4.5

VisiBroker® for Java™

Inprise Corporation, 100 Enterprise Way
Scotts Valley, CA 95066-3249

Inprise may have patents and/or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

Copyright © 1997, 2000 Inprise Corporation. All rights reserved. All Inprise and Borland brands and product names are trademarks or registered trademarks of Inprise Corporation. Java and all Java-based marks are trademarks or registered trademarks in the United States and other countries. Other brands and product names are trademarks or registered trademarks of their respective owners.

Printed in the U.S.A.

VJD0045WW21003

PDF

Contents

Chapter 1

Preface

1-1

What's new	1-1
Installing Gatekeeper and Gatekeeper samples	1-1
Typographic conventions	1-2
Platform conventions	1-2
Where to find additional information	1-3
Contacting developer support	1-3

Chapter 2

Gatekeeper

2-1

Introducing the Gatekeeper.	2-1
Gateway between the client and server	2-2
Firewall support	2-2
Dual-homed machine	2-2
Port filtering.	2-2
NAT device	2-3
Web server integration.	2-3
Access control.	2-3
Built-in access controllers	2-3
Scalability and fault tolerance.	2-4
Fault tolerance.	2-4
Load balancing	2-4
Gatekeeper properties	2-4
Gatekeeper scalability and performance guidelines	2-5
Gatekeeper performance and scalability profile	2-5
Impact of configuring number of connections	2-5
Impact of asynchronized invocation of Gatekeeper.	2-5
Planning your Gatekeeper deployment	2-6
Configuring the VisiBroker runtime to work with Gatekeeper	2-6
Configuring client-centric runtime.	2-7
Configuring Gatekeeper	2-7
Configuring the client	2-7
Using Gatekeeper as a proxy	2-7
Requiring HTTP tunneling	2-7
Requiring secure connections	2-8
Creating pass-through connections.	2-8
Configuring and deploying Gatekeeper with Java applets	2-9

Configuring and deploying Gatekeeper with server-side firewalls.	2-9
Using the Gatekeeper's proxy properties	2-9
Using Gatekeeper behind firewalls and as an IIOP proxy	2-11
Configuring the server	2-11
Configuring a single POA	2-11
Configuring the firewall policy for all POAs associated with a server	2-12
Configuring the server at runtime.	2-12
Using Gatekeeper with client-side firewalls	2-13
Gatekeeper chains	2-14
Support for Bidirectional Communication	2-14
Gatekeeper support for bidirectional IIOP	2-15
About the example.	2-15
The client	2-16
The server.	2-16
Security considerations	2-17
Gatekeeper and the Smart Agent	2-18
Gatekeeper's security features.	2-18
Starting the Gatekeeper	2-19
Gatekeeper command.	2-19
Options	2-20
Installing Gatekeeper as an NT service	2-20
Gatekeeper properties file	2-20

Chapter 3

Gatekeeper Management Console

3-1

Introducing the Gatekeeper Management Console.	3-1
Understanding how the Gatekeeper uses IP addresses and ports	3-2
Starting the Gatekeeper Management Console	3-2
Configuring basic properties	3-3
Configuring IIOP Proxy Services	3-5
Configuring general IIOP Proxy Services.	3-5
Request Forwarding Service	3-5
Callback Service	3-6
Location Service	3-7
Request Listeners	3-8
Callback Listeners	3-9
Configuring HTTP Service.	3-10
Configuring Administrative Service	3-11
Configuring Security	3-12
Configuring basic Security properties	3-12

Managing access control. 3-13

Managing access rules 3-15

Configuring properties for performance . . . 3-16

Configuring Gatekeeper performance
properties 3-16

Configuring ORB performance properties . 3-17

Configuring Load Balancing performance
properties 3-18

Creating and editing Gatekeeper property files 3-19

Exiting the Gatekeeper Management Console . 3-19

Appendix A

Gatekeeper properties **A-1**

Exterior Server Engine properties. A-1

ex-hiop listener properties A-2

ex-iiop listener properties A-2

Interior Server Engine properties A-3

Properties that support bidirectional
communications. A-3

Properties that support pass-through
connections A-4

Administration properties A-5

Index **I-1**

Tables

2.1 2-15	2.2	Files configured to support the bidirectional connection 2-17
-----	----------------	-----	--

Figures

2.1	Gatekeeper and firewall software on dual-homed host 2-10	2.3	Firewall architecture with a router in front of and behind the Gatekeeper . . 2-11
2.2	Firewall architecture with a router in front of the Gatekeeper 2-10	2.4	Chaining a series of Gatekeepers 2-14
		2.5	SSL Connections to Gatekeeper 2-19

Preface

This Preface lists the contents of the VisiBroker for Java *Gatekeeper Guide* and describes typographical and syntax conventions used throughout the manual. It also provides references for more information about CORBA.

What's new

The Gatekeeper 4.5 new features and enhancements include:

- Gatekeeper now provides support for bidirectional communications. See “Support for Bidirectional Communication” on page 2-14 for more information.
- You can create pass through connections between clients and servers. For more information, see “Creating pass-through connections” on page 2-8.
- An appendix listing all Gatekeeper-related properties is provided.

Installing Gatekeeper and Gatekeeper samples

To install Gatekeeper, refer to the instructions provided in the VisiBroker for Java *Installation Guide*. Samples of Gatekeeper configurations are also provided with this release. See the readme file for more information.

Note: The Gatekeeper supports both HTTPS and HTTP traffic for JDK plug-ins 1.1.x, 1.2.x, and 1.3 on both the Netscape and Internet Explorer browsers.

Typographic conventions

This manual uses the following conventions:

Convention	Used for
boldface	Bold type indicates that syntax should be typed exactly as shown. For UNIX, used to indicate database names, filenames, and similar terms.
<i>italics</i>	Italics indicates information that the user or application provides, such as variables in syntax diagrams. Also used to introduce new terms.
<code>computer</code>	Computer typeface is used for sample command lines and code.
UPPERCASE	Uppercase letters indicate SQL statements and terms. For Windows, used to indicate database names, filenames, and similar terms.
[]	Brackets indicate optional items.
{ }	Curly brackets are used in the more complex syntax statements to show a required item.
...	An ellipsis indicates the continuation of previous lines of code or that the previous argument can be repeated.
	A vertical bar separates two mutually exclusive choices.

Platform conventions

This manual uses the following conventions to indicate that information is platform-specific:

Windows	All Windows platforms including Windows 3.1, Windows NT, and Windows 95
WinNT	Windows NT only
Win95	Windows 95 only
Win98	Windows 98 only
Win2000	Windows 2000 only
UNIX	All UNIX platforms
Solaris	Solaris only
AIX	AIX only
HP-UX	HP-UX only
IRIX	IRIX only
Digital UNIX	Digital UNIX only

Where to find additional information

For more information about VisiBroker for Java, refer to the following sources:

- VisiBroker for Java *Programmer's Guide* provides information on developing distributed object-based applications in Java for Windows and UNIX platforms.
- VisiBroker for Java *Reference* contains the information on the VisiBroker for Java commands and Java interfaces.
- VisiBroker for Java *Installation Guide* contains the instructions for installing VisiBroker for Java on Windows and UNIX, and information for system administrators who are deploying distributed applications built using VisiBroker for Java.
- The Inprise website also provides a variety of useful information to developers and others who are interested in evaluating our products and ORB technology. The website contains a variety of white papers concerning distributed computing and VisiBroker for Java, and information specific to developers such as Frequently Asked Questions (FAQ) and their answers. There is also a Gatekeeper FAQ that answers common questions specific to Gatekeeper. Visit Inprise's website at <http://www.borland.com/>.

For more information about the CORBA specification, refer to the following sources:

- *The CORBA 2.4 Specification*. This document is available from the Object Management Group at <http://www.omg.org/> and describes the architectural details of CORBA.
- *The OMG IDL/Java Language Mapping Specification*. This document is available from the Object Management Group at <http://www.omg.org/> and describes the IDL-to-Java mapping specifications.
- *The CORBA/Firewall Security Specification*. This document is available from the Object Management Group at <http://www.omg.org/> and provides detailed information about GIOP Proxy.

Contacting developer support

Inprise offers a variety of support options. These include free services on the Internet, where you can search our extensive information base and connect with other users of Inprise products. In addition, you can choose from several categories of telephone support, ranging from support on installation of the Inprise product to fee-based consultant-level support and detailed assistance.

For more information about Inprise's developer support services, please see our website at <http://www.borland.com/devsupport>, call Inprise Assist at 800-523-7070, or contact our Sales Department at 800-632-2864. For customers outside the United States of America, please see our website at <http://www.borland.com/bww/intlcust.html>.

When you contact developer support, you will be asked to provide the following information:

- Your Access ID number
- Product name and version (for example, VisiBroker for C++, version 4.0)
- Operating system and version (for example, Windows NT Server 4.0 with Service Pack 5)
- Your desired priority (low, medium, high)
- Brief description of the problem
- Details of any error messages or exceptions raised

For information about year 2000 issues and our products, see the following URL:
<http://www.borland.com/about/y2000/>.

Gatekeeper

This chapter describes how you can use the VisiBroker Gatekeeper to enable VisiBroker client applications and applets to communicate with servers across networks, while still conforming to the security restrictions imposed by web browsers, firewalls, or a combination of the two. It also describes how to perform load balancing, fault tolerance, and access control.

Introducing the Gatekeeper

The Gatekeeper enables VisiBroker clients to communicate with servers across networks, while still conforming to the security restrictions imposed by web browsers and firewalls. The Gatekeeper serves as a gateway from clients to servers when security restrictions imposed by Java sandbox security or firewalls prevent clients from communicating with servers directly. The Gatekeeper is a GIOP proxy server fully compliant with the OMG CORBA Firewall Specification.

In addition, the Gatekeeper provides the following functionality:

- Bootstrapping
- Location transparency
- Callback enabling
- HTTP tunneling
- Acts as a simple web server to load classes
- Customizable IP-based access control
- Load balancing
- Fault tolerance

The Gatekeeper comes with a Management Console that enables you to set the Gatekeeper's properties to meet the requirements of your networked system environment. The Gatekeeper's properties reside in a *properties file* that the Gatekeeper references at startup time. For a full description of the Gatekeeper Configuration Manager, see Chapter 3, "Gatekeeper Management Console."

Gatekeeper works with two major types of security restrictions: Java applet sandbox security and firewalls. With Java applet sandbox security, an applet can talk only to the server from which it was downloaded. Firewalls usually restrict communication to certain hosts, ports, or protocols by using various hardware or software, such as routers, firewall software, and so on. The following sections discuss how Gatekeeper works with firewalls.

Gateway between the client and server

When a distributed system based on VisiBroker is deployed over the Internet or Intranet, many security restrictions can apply to the system, including the following:

- Java sandbox security prevents unsigned Java applets from communicating with servers other than the ones running on the host machine from which the applets were downloaded.
- Server-side firewalls can prevent the client from accessing certain hosts.
- Client-side firewalls can prevent incoming connections or prohibit protocols other than HTTP.

The Gatekeeper, along with the VisiBroker ORB, provides mechanisms to work with these restrictions based on the OMG CORBA Firewall specification by acting as a gateway between the client and the server.

When certain restrictions prevent the clients from connecting directly to the server, the client can choose to connect to Gatekeeper if the server object reference has the necessary information. The clients can send messages to Gatekeeper; Gatekeeper will forward the messages to the server.

When certain restrictions prevent the server from connecting back to the client to do callbacks, the server can choose to connect to Gatekeeper if the callback object reference has the necessary information. The server can send callback messages to Gatekeeper; Gatekeeper will forward the messages to the client.

Firewall support

Dual-homed machine

A dual-homed machine is a machine with two network interface cards (NICs), and therefore, two IP addresses. Usually one NIC is connected to the outside network; the other is connected to the inside network. A proxy server is often deployed on a dual-homed machine to proxy the messages from one network to the other network. To support this type of configuration, Gatekeeper allows the user to set the different listeners/dispatchers to listen and accept messages on two different IP addresses.

Port filtering

Another common firewall configuration is to filter the IP packets based on the IP source and destination information at the IP router/bridge. For example, the router can be set to allow only IP packets with a specific destination port number or

destination IP address. To support this type of configuration, Gatekeeper allows the user to control both the port number where it receives the packets and the port number where it sends the packet.

NAT device

In order to hide the internal network information, many firewalls use a Network Address Translation (NAT) device to change the source and destination information in the IP packet when the packet is going from one network to another network.

There is usually a one-to-one mapping between a fake IP/port and a real IP/port. When the packet is sent from the protected network, the source IP/port of the packet is changed to the corresponding fake IP/port. When the packet is received from the outside network by the NAT device, the destination IP/port of the packet is changed to the corresponding real IP/port.

There are two ways Gatekeeper supports this type of configuration:

- Permanently modify the Gatekeeper object reference to use the fake IP/port. This hides the internal network information fully. This also has some performance advantages at bind time since it simplifies the client's logic. However, this configuration prevents internal clients that don't recognize the fake IP/port from accessing the Gatekeeper.
- Add a `TcpFirewallMechanism` to the server object reference. This is less secure than the solution discussed previously because it exposes the internal IP address. However, this supports a configuration that allows Gatekeeper to be used by clients regardless of whether or not they recognize the fake IP/port.

Web server integration

The Gatekeeper can be run as a servlet inside any Web server that supports servlets. Gatekeeper is started with a special HIOP listener whose purpose is to generate the right HIOP component in the Gatekeeper's IOR. The HIOP component should contain the Web server's host, port and the path to the Gatekeeper servlet. The client will send HIOP requests to the Gatekeeper as specified in the HIOP component.

Note Because you are running Gatekeeper as a servlet instead of from the Console, you will lose some administrative capabilities as well as Gatekeeper output capabilities.

Access control

Built-in access controllers

Gatekeeper provides a rules-based controller. This controller can deny or grant accessibility based on:

- Operation
- Signed by
- Server's host/port
- Server's subnet
- Client's host/port
- Client's subnet

All the rules are evaluated in the order in which you specified them. Action is taken based on the first matched rule. If there is no matched rule, the default action you specify is taken.

Scalability and fault tolerance

Because the Gatekeeper is most often used to provide a single point of access to the internal network, it can become congested or become the single point of failure. Better scalability and fault tolerance are provided through Gatekeeper clustering.

Fault tolerance

A master Gatekeeper and one or more backup Gatekeepers can be clustered together to be viewed as a single Gatekeeper by the client. There are several ways to cluster the Gatekeepers.

- Cluster the Gatekeepers as different firewall paths to the server. This doesn't require any changes to the Gatekeeper configuration; you only need to configure the server to include all the backup Gatekeepers as a firewall path on the server listener. This makes server configuration more complex.
- Fold all the backup Gatekeeper's object references (profiles) into the master Gatekeeper's object reference. When the master Gatekeeper fails, the client would rebound to one of the other backup Gatekeepers automatically. This approach can make the Gatekeeper's object reference very large.
- Fold all the backup Gatekeeper's information as a component in the master Gatekeeper's object reference. When the master Gatekeeper fails, the client would rebound to one of the backup Gatekeepers specified in the component. This reduces the master Gatekeeper's object reference size but requires changes to the client.

Load balancing

A master Gatekeeper and one or more slave Gatekeepers can be clustered together. The master Gatekeeper is responsible for balancing the load among the slave Gatekeepers. The server should export the master Gatekeeper object reference only.

The master Gatekeeper can balance the load between slave Gatekeepers on a per object level. On the object level, each client will be redirected to one of the slave Gatekeepers based on the load. In general, this will balance the load more evenly but potentially use more resources and be slower. The default load balance policy is round-robin.

Gatekeeper properties

Use the following command to start Gatekeeper:

```
gatekeeper -props <property file>
```

Where property file indicates the name of the Gatekeeper's properties file. You can include the entire path when you specify the file name. The default location for this

file is the directory where you started Gatekeeper. The default name for this file is `gatekeeper.properties`.

If you don't specify the property file name in the command line, the Gatekeeper looks for `gatekeeper.properties` in the current directory.

See the Release Notes for more information on Gatekeeper properties.

Gatekeeper scalability and performance guidelines

When assessing Gatekeeper performance it is useful to compare a Gatekeeper scenario (Client-Gatekeeper-Server) to a direct scenario (Client-Server).

Note Here performance is represented as response time and scalability is represented as throughput.

The Gatekeeper scenario requires two connections and thus two invocations. As a result:

- **Throughput is reduced:** It may be reduced by as much as 50 percent when compared to the direct scenario.
- **Response Time is slowed:** Response time will take longer when compared to the direct scenario. In some cases, it may take up to 200 percent longer.

Gatekeeper performance and scalability profile

Gatekeeper does not introduce any new performance threshold or throughput threshold. This means that Gatekeeper will have the same performance and throughput profile as the VisiBroker ORB.

Impact of configuring number of connections

The number of connections between Gatekeeper and Server is a configurable parameter for performance and throughput. However, this parameter should only be modified in situations where the server is intensely accessed.

When Multiple Connections are enabled:

- Performance is improved when the number of clients is small.
- Performance is decreased when the number of clients is large.

For more information on configuring Gatekeeper performance properties, see "Configuring properties for performance" on page 3-16.

Impact of asynchronized invocation of Gatekeeper

Asynchronized invocation of Gatekeeper does not have a very significant impact on performance and scalability.

Planning your Gatekeeper deployment

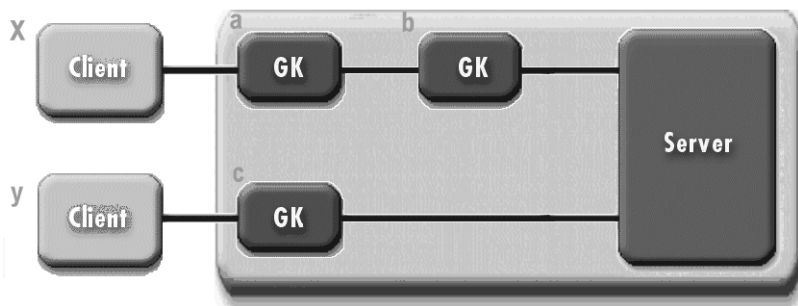
Before you configure your client and server computers you should take the following steps:

- Understand your firewall configuration
- Understand your network configuration
- Develop a deployment plan.

When examining your firewall configuration, you need to decide where to put Gatekeeper. Will you put Gatekeeper inside or outside the firewall? In either case, you'll need to determine on which port to run Gatekeeper.

Configuring the VisiBroker runtime to work with Gatekeeper

You should configure the server to include Gatekeeper information. See the VisiBroker for Java *Programmer's Guide* for detailed information on how to configure the server. The following illustration shows two firewall configurations. Configuration X has a chain of two Gatekeepers. Configuration Y has a single Gatekeeper.



To configure the server for these configurations, you would have to put the following information into the server's properties file.

First, you have to declare all firewall paths.

```
Firewall paths = x,y
```

Next, identify the components for each path.

```
firewall-path.x = a,b
firewall-path.y = c
```

For each component, specify whether its type is proxy or TCP.

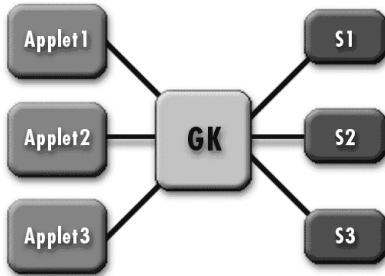
```
vbroker.firewall.n.type=<PROXY | TCP>
```

For each firewall path, specify an IOR. You can specify either an IOR string or an HTTP URL.

```
vbroker.firewall.n.ior=<IOR string| URL (HTTP)>
```

Configuring client-centric runtime

When you want to configure Gatekeeper to access certain servers, you can set `vbroker.orb.gatekeeper.ior` to the Gatekeeper's IOR using URL naming or using a stringified IOR.



Configuring Gatekeeper

Configuring the client

You can set how a client communicates with a server by using Quality of Service (QoS) APIs. In particular, using policies and properties, you can specify whether:

- Client use a gatekeeper as a proxy for the real server;
- Client communicate to servers in the HTTP-tunneled channel;
- Clients talk to servers through IIOP/SSL or IIOP/HTTPS;
- Messages pass between client and server connections through a gatekeeper entirely unexamined by the gatekeeper;
- Servers can asynchronously connect to clients. (For more information, read “Support for Bidirectional Communication” on page 2-14.)

Using Gatekeeper as a proxy

If the `vbroker.orb.alwaysProxy` property is set to `true`, clients are forced to use Gatekeeper to proxy requests to servers. This property is optional. If you do not set it, Gatekeeper determines whether or not the object is hidden behind a server-side firewall and traverses the firewall accordingly. When a client invokes both local objects inside the trusted network and remote objects hidden behind the firewall, the `vbroker.orb.alwaysProxy` property should not be set.

Requiring HTTP tunneling

If the `vbroker.orb.alwaysTunnel` property is set to `true`, clients will communicate to servers in the HTTP-tunneled channel, which means that the client applet

communicates to the Gatekeeper through HTTP and the Gatekeeper relays the request to the actual server object through IIOP. Replies from the server object to Gatekeeper are communicated through IIOP. Gatekeeper then forwards those replies to the client through HTTP.

Requiring secure connections

If the `vbroker.orb.alwaysSecure` property is set to `true`, clients talk to servers through IIOP/SSL or IIOP/HTTPS.

Creating pass-through connections

The pass-through feature is useful in situations where you do not want the gatekeeper to terminate connections or interpret messages. For example, the Gatekeeper may not have SSL or the associated Certificates to establish trust with the client. In such a case, the pass-through connection helps the client and server negotiate their SSL connection without the gatekeeper participating. Since the gatekeeper does not have SSL, it will not be able to read any messages (which will be encrypted), but it doesn't have to, since (by definition) the gatekeeper doesn't look at messages in a pass-through connection.

Two properties determine whether pass-through connections can be established:

The `vbroker.orb.proxyPassthru` property sets the value of the ORB-level `PROXY_MODE_POLICY` property. If set to `true`, it dictates that all objects using proxy on the client will request pass-through connections. Alternatively, you can set the `PROXY_MODE_POLICY` on specific objects so that only those particular objects request pass-through connections.

The `vbroker.gatekeeper.enablePassthru` property instructs a gatekeeper to accept pass-through connections. This is a property that is global to the gatekeeper and affects the gatekeeper's behavior only.

If the `vbroker.orb.proxyPassthru` property is set to `true`, client will try to acquire pass-through connections through the gatekeeper. However, Gatekeeper grants pass-through connections only if the `vbroker.gatekeeper.enablePassthru` property on the Gatekeeper is set to `true`. If the client obtains a pass-through connection, the Gatekeeper will not examine any messages that pass between server and client. If the Gatekeeper disallows pass-through connections (i.e., `vbroker.gatekeeper.enablePassthru` is set to `false`), clients obtain normal (non-pass-through) connections to the server and messages exchanged between the client and server are examined by the Gatekeeper.

The following properties are provided to help configure pass-through connections:

```
vbroker.gatekeeper.passthru.blockSize
vbroker.gatekeeper.passthru.connectionTimeout
vbroker.gatekeeper.passthru.logLevel
vbroker.gatekeeper.passthru.streamTimeout
```

See “Properties that support pass-through connections” on page -4 for more information about these properties.

Caution: The pass-through feature heavily taxes the resources of the Gatekeeper. If you choose to use this feature, be sure to configure Gatekeeper with sufficient memory and increased sockets.

You can use different combinations of these policies to determine how you want clients talk to servers.

Configuring and deploying Gatekeeper with Java applets

You can deploy Gatekeeper on the same host as the web server. Start Gatekeeper in the applet code base if Gatekeeper is acting as the HTTP server for downloading applet classes.

Applets should set `vbroker.orb.alwaysTunnel` if the client will be doing HTTP tunneling. Applet clients must include the property `vbroker.orb.gatekeeper.ior=""` and not set `vbroker.locator.ior`.

Configuring and deploying Gatekeeper with server-side firewalls

There are many different architectures and configurations that are called firewalls whose purpose is to restrict access to a network. In this section, several examples are given showing you how to use Gatekeeper with some of the firewall configurations.

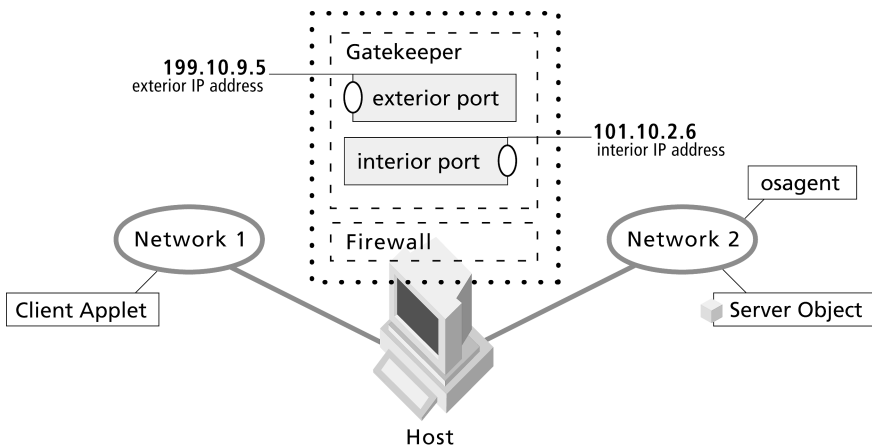
Using the Gatekeeper's proxy properties

Certain firewall configurations use the Network Address Translation (NAT) device to map IP addresses or port numbers. You can deploy NAT before, behind, or both before and behind the Gatekeeper, and then set the appropriate Gatekeeper proxy properties. For example, if you have a NAT that maps address 199.20.1.6 to 199.10.9.5 and the NAT is deployed before the Gatekeeper, you set the listener's request channel IIOP host to 199.10.9.5 and the IIOP proxy host to 199.20.1.6. When the Gatekeeper hands out an object reference, it will use 199.20.1.6 instead of 199.10.9.5. Similarly, you can set the TCP mechanism as specified in the OMG CORBA specification.

Dual-homed hosts

You can use the Gatekeeper with a firewall that has dual-homed host computers. Dual-homed computers have at least two network interfaces to bridge separate networks. In order to make the Gatekeeper forward messages between the different networks, you must specify the exterior and interior host IP addresses for the Gatekeeper.

Figure 2.1 Gatekeeper and firewall software on dual-homed host



For dual-homed hosts, the Gatekeeper's exterior host IP address must be the same as the web server's IP address.

Router filtering

You can use the Gatekeeper with a firewall that has router filtering. Usually the Gatekeeper is running on the bastion host with a router in front, behind, or both in front and behind the Gatekeeper.

Figure 2.2 Firewall architecture with a router in front of the Gatekeeper

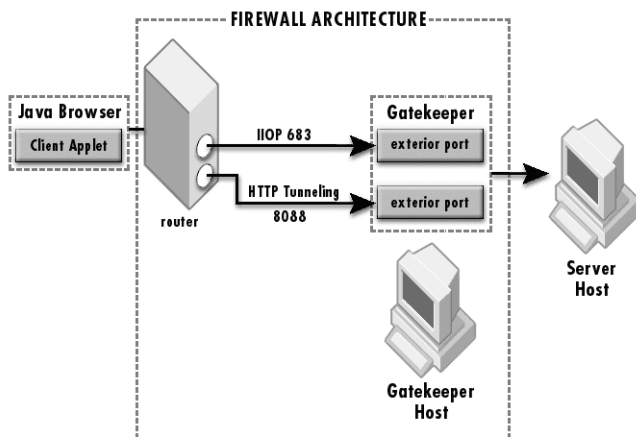
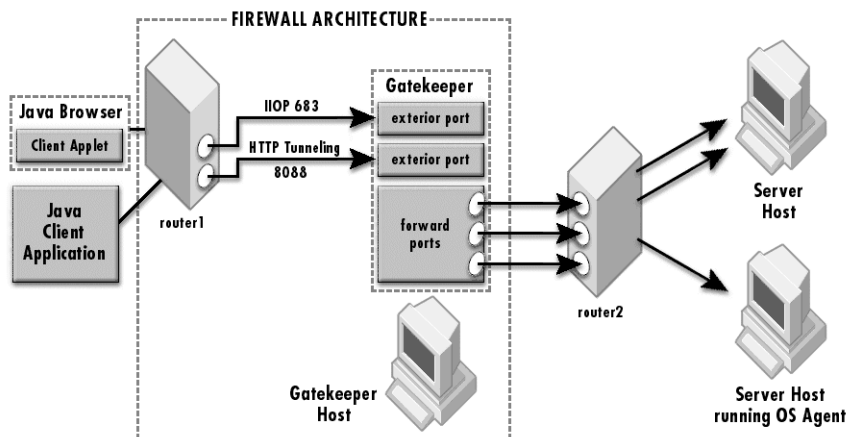


Figure 2.3 Firewall architecture with a router in front of and behind the Gatekeeper

Using Gatekeeper behind firewalls and as an IIOP proxy

When a server is running behind a firewall, all clients running outside the firewall will fail to bind to this server because the firewall does not allow IIOP requests to pass through it. In this case, Gatekeeper can run on the firewall and help clients to bind to the server. To do this, Gatekeeper intercepts the bind request from the clients and binds to the server on behalf of the clients, sending its IOR to the client so that all IIOP requests from clients can be forwarded to the server.

You need to configure both programming and runtime environments so that Gatekeeper can work behind firewalls. The following examples use the Bank example as a basis.

Configuring the server

In order to allow a server to traverse a firewall, you must specify a firewall policy on the Portable Object Adapter (POA) where the servant is activated. You can configure POAs individually or globally.

In particular, the following code must be added to the server.

Configuring a single POA

The following code creates the firewall policy:

```
org.omg.CORBA.Any fw_policy_value = orb.create_any();
com.inprise.vbroker.firewall.FirewallPolicyValueHelper.insert(
    fw_policy_value, com.inprise.vbroker.firewall.EXPORT.value);
org.omg.CORBA.Policy firewall_policy = orb.create_policy(
    com.inprise.vbroker.firewall.FIREWALL_POLICY_TYPE.value,
    fw_policy_value);
org.omg.CORBA.Policy[] policies = {
```

Configuring the server at runtime

```
        firewall_policy,  
        rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT)  
    };
```

Then apply the policy to the POA on which the servant will be activated.

```
POA bankPOA = rootPOA.create_POA(  
    "bank_agent_poa", rootPOA.the_POAManager(), policies);
```

Only the root POA takes the default policy, so it can be used to activate any servants that might need to be accessed behind a firewall. You will need to create another POA to activate the Account servant. Since the Account servant should not be bound by clients directly, you create the POA as a transient POA

```
policies = new org.omg.CORBA.Policy[] {  
    firewall_policy,  
    rootPOA.create_lifespan_policy(LifespanPolicyValue.TRANSIENT)  
};  
  
POA accountPOA = rootPOA.create_POA(  
    "account_agent_poa", rootPOA.the_POAManager(), policies);
```

Configuring the firewall policy for all POAs associated with a server

If you use the following syntax, you set the firewall policy for all POAs associated with a server:

```
- D vbroker.orb.exportFirewallPath=true
```

If you specify the `exportFirewallPath` property, you do not need to add a firewall policy when creating a POA, so you do not have to make modifications to the source code.

```
org.omg.CORBA.Policy[] policies = {  
    rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT)  
};
```

Then apply the policy to the POA on which the servant will be activated.

```
POA bankPOA = rootPOA.create_POA(  
    "bank_agent_poa", rootPOA.the_POAManager(), policies);  
policies = new org.omg.CORBA.Policy[] {  
    rootPOA.create_lifespan_policy(LifespanPolicyValue.TRANSIENT)  
};  
  
POA accountPOA = rootPOA.create_POA(  
    "account_agent_poa", rootPOA.the_POAManager());
```

Configuring the server at runtime

At runtime, several properties need to be loaded when initializing the ORB, that is, when the following method is invoked:

```
org.omg.CORBA.ORB.init(String[] args, java.util.Properties property);
```

These properties can be loaded either from the command line or through a file stream.

The following property causes the firewall package to be loaded into the ORB. This property is necessary for both client and server.

```
vbroker.orb.dynamicLibs=com.inprise.vbroker.firewall.Init
```

When using Gatekeeper as an IIOP proxy across a firewall, the following properties also need to be specified:

```
vbroker.se.iiop_tp.firewallPaths=internet
vbroker.firewall-path.internet=proxy
vbroker.firewall.proxy.type=PROXY
vbroker.firewall.proxy.ior=http://www.inprise.com/GK/gatekeeper.ior
```

The first property defines all internet IIOP requests that might be forwarded across firewall. The second property defines one firewall path for internet IIOP requests, and the path is called “proxy.” The third property defines the firewall as a PROXY type, which identifies Gatekeeper as an IIOP proxy to forward all IIOP requests. The fourth property defines the IOR of the IIOP proxy. In this case, URL naming is used to find the IOR of the Gatekeeper.

When using a router to forward IIOP message at the TCP level, the following properties need to be specified as well:

```
vbroker.se.iiop.firewallPaths=intranet
vbroker.firewall-path.intranet=tcp
vbroker.firewall.tcp.type=TCP
vbroker.firewall.tcp.host=router1
vbroker.firewall.tcp.iiop_port=12345
vbroker.firewall.tcp.hiop_port=0
```

The first property defines all intranet IIOP messages that might be forwarded across firewall. The second property defines one firewall path for intranet IIOP messages, and the path is called “tcp.” The third property defines the firewall path as a TCP type, which provides a predefined port to forward all IIOP messages on a router or other network device. The fourth property defines the host name of the network device. The fifth property defines the port for forwarding all IIOP messages. The sixth property disables HIOP (HTTP Inter ORB Protocol) forwarding by specifying the value as 0.

Using Gatekeeper with client-side firewalls

Gatekeeper supports both standard HTTP and HTTP tunneling. If your client is running behind a firewall which only allows HTTP traffic, the client must use HTTP tunneling and the Gatekeeper to talk to the server object.

The Gatekeeper supports HTTP tunnelling. If there's a client-side firewall that only allows HTTP protocol, the ORB in the client detects the failure when trying IIOP and automatically switches to HTTP protocol.

Note HTTP tunneling support is not backward-compatible with earlier versions of Gatekeeper.

In certain network configurations, it may take some time for the ORB to determine that IIOP is not allowed and then to switch to HTTP tunneling. To avoid this delay,

you can force the ORB to always use HTTP tunneling by specifying the `vbroker.orb.alwaysTunnel` param tag in your applet tag:

```
<param name=vbroker.orb.alwaysTunnel value=true>
```

For an application, use the `-D` syntax:

```
-D vbroker.orb.alwaysTunnel=true
```

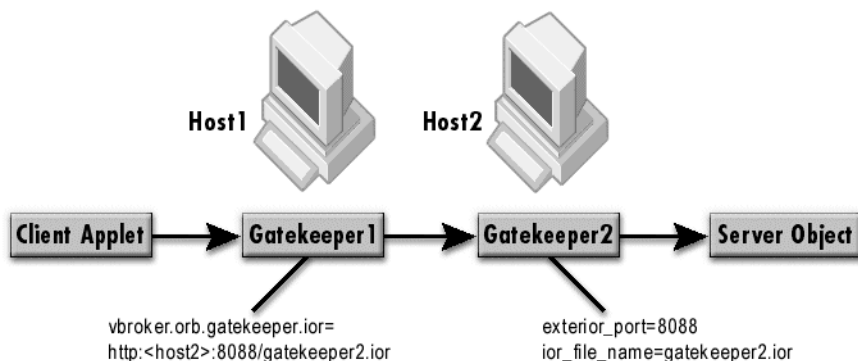
Using HTTP tunneling, the listener can act as a simple web server to download simple classes.

Note You cannot use callbacks while HTTP tunneling.

Gatekeeper chains

In complex network environments, it can be useful to chain multiple Gatekeepers together. The Gatekeeper supports this advanced functionality.

Figure 2.4 Chaining a series of Gatekeepers



Gatekeeper chaining can be used to communicate across multiple firewalls. For example, it takes two chained Gatekeepers to communicate across a series of three firewalls. To create a chained configuration, use the Gatekeeper Management Console.

Support for Bidirectional Communication

Most clients and servers that exchange information via the Internet are typically protected by corporate firewalls. In implementations where requests are initiated only by the clients, the presence of firewalls is usually transparent to the clients. However, there are cases where clients need information asynchronously, that is, information must arrive that is not in response to a request. Firewalls usually prevent servers from initiating connections back to clients. Therefore, if a client is to receive asynchronous information, it usually requires additional configuration.

In earlier versions of GIOP and VisiBroker, the only way to make it possible for a server to send asynchronous information to a client was to use a client-side Gatekeeper to handle the callbacks from the server.

If you use bidirectional IIOP, rather than having servers open separate connections to clients when asynchronous information needs to be transmitted back to clients (these would be rejected by client-side firewalls anyway), servers use the client-initiated connections to transmit information to clients. The CORBA specification also adds a new policy to portably control this feature.

Because bidirectional IIOP allows callbacks to be set up without a Gatekeeper, it greatly facilitates deployment of clients. For information about bidirectional communications exclusive of the Gatekeeper, see the *VisiBroker for Java Programmer's Guide* for more information.

Gatekeeper in Visibroker for Java version 4.5 also provides new support for bidirectional IIOP.

Gatekeeper support for bidirectional IIOP

The gatekeeper acts as a server for the client and as a client for the server. You can set either or both of these connections to be bidirectional.

If you configure the gatekeeper to both publish the listen points and accept the listen points, the client-gatekeeper and gatekeeper-server connections will both be bidirectional.

You can also selectively make bidirectional connections. If the client defines `vbroker.orb.enableBiDir=client` and the server defines `vbroker.orb.enableBiDir=server`, the chart below depicts the connection state for different values of `vbroker.orb.enableBiDir` at the gatekeeper.

Table 2.1

<code>vbroker.orb.enableBiDir=</code>	Client-Gatekeeper	Gatekeeper-Server
client	unidirectional	bidirectional
server	bidirectional	unidirectional
both	bidirectional	bidirectional
none	unidirectional	unidirectional

About the example

An example that demonstrates Gatekeeper's support for bidirectional connections is located in the `examples/gatekeeper/bank_bidir` subdirectory under the Gatekeeper for Java installation.

The Bank BiDir example is similar to the Bank Callback example, except that in the BiDir example, bidirectional connections are established between the client, gatekeeper, and server. In other words, in the bidirectional implementation, the same connection is used for both forward invocations as well as callbacks.

This example demonstrates how to:

- Configure the client to enable bidirectional connections via the property file.
- Program the client to create callback objects that can be passed as arguments to invocations on server objects.
- Configure the server to set up firewall paths containing the server side inbound firewall via property file. It also demonstrates how to configure the server so that it can accept bidirectional connections.
- Program the server to export the firewall path in server objects IOR.
- Configure the gatekeeper so that it supports bidirectional connections.

The client

In this example, the client `Client.java`:

- 1 Creates a callback object on the POA named `callback_poa`. (This callback object will be invoked by the server through the gatekeeper.)
- 2 Binds to the **AccountManager** object.
- 3 Sends the object reference of this callback object to the server by opening a bank account by invoking `open()` and passing the callback object as an argument.
- 4 Queries the **Account** object reference obtained for the balance, again passing the callback object (this time it is passed to the balance method).

The server

In the example, the server `Server.java`:

- 1 Creates a persistent POA named `bank_poa` and a transient POA named `account_poa` with firewall policy value of `EXPORT`.
- 2 Creates an instance of the **AccountManager** servant.
- 3 Activates that servant on `bank_poa`.
- 4 Starts waiting for client requests.
- 5 Responds to the requests by invoking a method on the client-initiated callback object through the Gatekeeper.

Table 2.2 Files configured to support the bidirectional connection

File name	Description
<code>server.properties</code>	Property file used to configure the bank server. In this example, the server is configured to accept listen points so that the connection between the gatekeeper and the server will be bidirectional. To make the connection unidirectional, either remove the property <code>vbroker.orb.enableBiDir</code> or set the value of this property to none. The other properties in this file are for loading the firewall package and then setting the firewall path so that server-side objects can be bound and called by the client.
<code>client.properties</code>	Property file used to configure the bank client. In this example, the client is configured to publish its listen points so that the connection between the client and the gatekeeper will be bidirectional. To make the connection unidirectional, either remove the <code>vbroker.orb.enableBiDir</code> property or set its value to none. As with the server, the <code>vbroker.orb.dynamicLibs</code> property is set to load in the necessary firewall library so that the client request can traverse the gatekeepers.
<code>gatekeeper.properties</code>	Property file used to configure the gatekeeper. In this example, the gatekeeper is configured to both publish the listen points and accept the listen points. Hence, both the client-gatekeeper and gatekeeper-server connections will be bidirectional. These connections can be converted into unidirectional by either removing the <code>vbroker.orb.enableBiDir</code> property or by setting this property to the value none.

Caution: Security considerations

Use of bidirectional IIOP may raise significant security issues. In the absence of other security mechanisms, a malicious client may claim that its connection is bidirectional for use with any host and port it chooses. In particular, a client may specify the host and port of security-sensitive objects not even resident on its host. In the absence of other security mechanisms, a server that has accepted an incoming connection has no way to discover the identity or verify the integrity of the client that

initiated the connection. Further, the server might gain access to other objects accessible through the bidirectional connection. This is why use of a separate, bidirectional SCM for callback objects is encouraged. If there are any doubts as to the integrity of the client, it is recommended that bidirectional IIOP not be used.

For security reasons, a server running VisiBroker for Java will not use bidirectional IIOP unless explicitly configured to do so. The property `vbroker.se.<sename>.scm.<scmname>.manager.importBiDir` gives you control of bidirectionality on a per-SCM basis. For example, you might choose to enable bidirectional IIOP only on a server engine that uses SSL to authenticate the client, and to *not* make other, regular IIOP connections available for bidirectional use. (See the Properties section for more information about how to do this.) In addition, on the client-side, you might want to enable bidirectional connections only to those servers that do callbacks outside of the client firewall. To establish a high degree of security between the client and server, you should use SSL with mutual authentication (set `vbroker.security.peerAuthenticationMode` to `REQUIRE_AND_TRUST` on both the client and server).

Gatekeeper and the Smart Agent

The Gatekeeper is a server to the client applet; the Gatekeeper is a client to the server object. In other words, the Gatekeeper acts like a proxy server in relation to the applet and it acts like a proxy client in relation to the server object. Therefore, the Gatekeeper uses all of the features a client would normally use—it uses the Smart Agent to locate server objects.

If Gatekeeper and the OS Agent are running on different subnets, use the following syntax:

```
-D ORBagent=[IP address of the osagent]
```

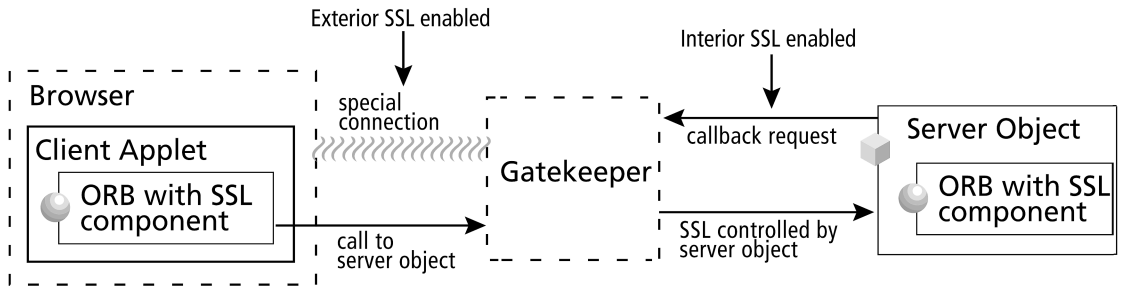
Gatekeeper's security features

Gatekeeper provides the following security features:

- Relay IIOP/SSL connection between client and server
- Perform authentication on behalf of server
- Support HTTPS tunneling
- Enable IIOP/SSL callback
- Forward credentials

Note To make use of these Gatekeeper security features, the Inprise Security Service must be installed.

Gatekeeper can listen on secured connections (SSL) and do authentication on the exterior side to provide protection for interior servers.

Figure 2.5 SSL Connections to Gatekeeper

For more information on how to set security-related properties on Gatekeeper, see Chapter 3, “Gatekeeper Management Console.”

Starting the Gatekeeper

The choice of directory in which to start the Gatekeeper is determined by how you are using the Gatekeeper:

- As a web server
- In combination with a separate web server
- As IIOP proxy server for a firewall

If you use the Gatekeeper as a web server, then it’s better to start the Gatekeeper in the same directory as the Java applets’ code base.

If you use the Gatekeeper in combination with a separate web server, then you can start the Gatekeeper on the same machine as the web server.

If you use the Gatekeeper as an IIOP proxy, consult your Firewall Administrator.

Gatekeeper command

Use the following command to start the IIOP Gatekeeper:

```
prompt>gatekeeper
```

Syntax `gatekeeper [-props <properties_file_name>]`

Example `gatekeeper -props devprops`

When you start the Gatekeeper, you will see a start up message followed by a series of messages indicating the services that being started.

Options

When using the `gatekeeper` command, you can specify the name and location of the Gatekeeper’s properties file:

Option	Description
<code>-props file_name</code>	Indicates the name of the Gatekeeper’s properties file. You can include the entire path when you specify the file name. The default location for this file is the directory where you started the Gatekeeper. The default name for this file is <code>gatekeeper.properties</code> .
<code>-h, -help, -usage, -?</code>	Displays usage information.
<code>-quiet</code>	Specifies that Gatekeeper not generate output.

Installing Gatekeeper as an NT service

You can install Gatekeeper as an NT service, though make sure you can run Gatekeeper from a DOS prompt.

To install Gatekeeper as an NT service, type the following command at a command line, where *servicename* is the name of the Gatekeeper you’re installing.

```
gatekeeper -install "servicename"
```

If you use the `-props` option to specify a properties file, make sure you include the full path name of the properties file you specify.

After you’ve installed Gatekeeper as an NT service, you can start it using the standard Services control panel.

To remove Gatekeeper, use the following syntax, where *servicename* is the name of the Gatekeeper you’re removing:

```
gatekeeper -remove "servicename"
```

Gatekeeper properties file

The properties for the Gatekeeper reside in a file built when you use the Gatekeeper Management Console. The default name of the file is `gatekeeper.properties`, but you can specify another name for the file using the `-props` command-line option with the `gatekeeper` command.

For more information, see Chapter 3, “Gatekeeper Management Console.”

Gatekeeper Management Console

This chapter describes how you can use the Gatekeeper Management Console to configure the properties of the Gatekeeper.

Introducing the Gatekeeper Management Console

The Gatekeeper Management Console is a tool that enables you to configure the behavior of the Gatekeeper to meet the needs of your network environment. You can use it to create and edit a property file for the Gatekeeper before running it for the first time.

You can also use it to configure the Gatekeeper dynamically, when it's running, allowing you to modify the properties and behavior of the Gatekeeper at runtime without shutting down the Gatekeeper.

You cannot perform the following functions if the VisiBroker Security Service is not run with Gatekeeper (if you are configuring Gatekeeper dynamically) or with the VisiBroker Console (if you are configuring Gatekeeper while it is not running).

- Enabling all the SSL listeners
- Enabling the firewall policies
- Selecting peer authentication
- Selecting SSL identity
- Adding new access controllers and rules

If the property changes require you to restart Gatekeeper, you can do so from within the Console. You will need to close and expand the Gatekeepers folder to refresh its content before you can select and configure it again.

Understanding how the Gatekeeper uses IP addresses and ports

There are several IP addresses you can set:

- Exterior
- Interior, if dual-homed

There are several ports you can set:

- Exterior
- Interior
- Forward
- Callback

An endpoint is a combination of a specific IP address and port. There are four TCP endpoints used by the Gatekeeper to communicate with clients and servers. The exterior IP address and exterior port TCP endpoint are used by the Gatekeeper to accept the client's connections. The exterior IP address and callback port TCP endpoint are used by the Gatekeeper to accept the applet's special connections for callback objects. The interior IP address and interior port TCP endpoint are used by the Gatekeeper to accept the server's connection to invoke the client's callback objects. The interior IP address and a range of forward ports are used by the Gatekeeper to forward calls from the applet to server objects.

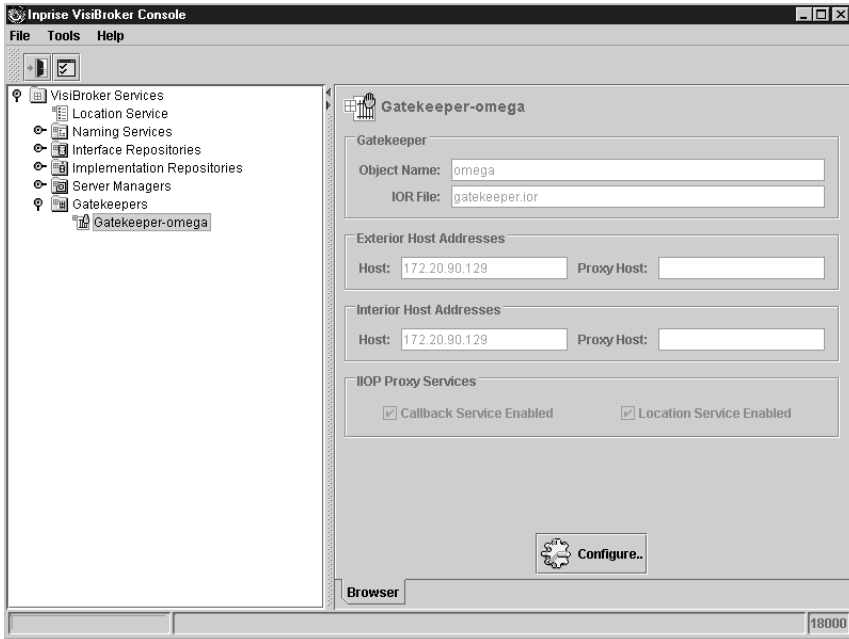
Starting the Gatekeeper Management Console

The Gatekeeper Management Console is an integrated component of the VisiBroker Console. To start the Gatekeeper Management Console, you have to start the VisiBroker Console first.

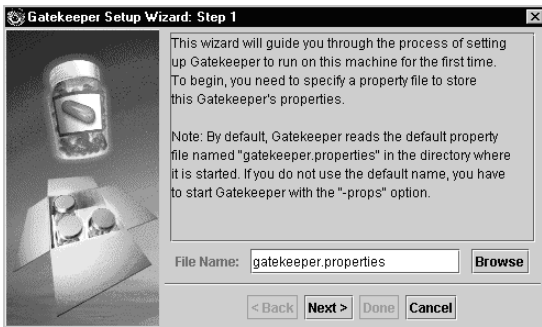
To start the Gatekeeper Management Console:

- 1 Start the VisiBroker Console.
- 2 Expand the VisiBroker Services folder.
- 3 Expand the Gatekeepers folder.
- 4 Select the Gatekeeper you want to view and configure. The VisiBroker Console displays a list of basic properties for the selected Gatekeeper in the right pane.

- 5 Click the Configure button to start the Gatekeeper Management Console.



If this is the first time you've run Gatekeeper, a setup wizard appears, prompting you for information. After you specify the appropriate information, the Gatekeeper Management Console appears.



Configuring basic properties

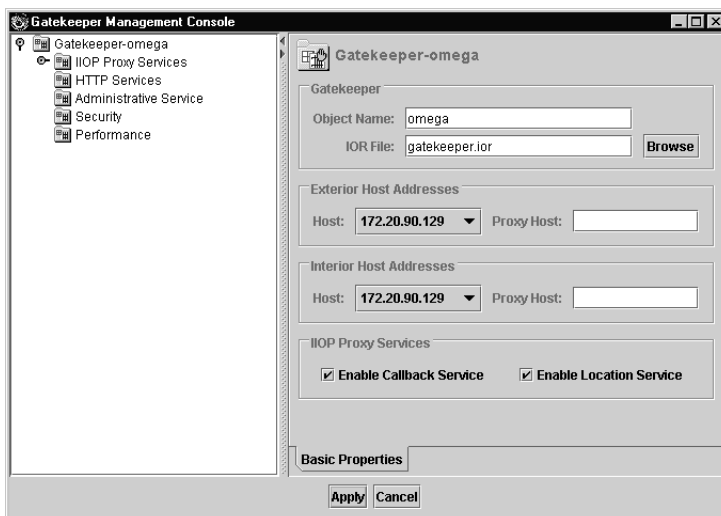
After you've clicked the Configure button, the Gatekeeper Management Console displays other Gatekeeper services and features you can configure.

- IIOP Proxy Services
- HTTP Tunneling Services
- Administrative Service
- Security Service
- Performance and Scalability

You can configure the properties associated with these features, by selecting the feature in the list, then setting properties shown in the Console.

To configure basic properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure.



- 2 Click the Configure button, then specify any of the following properties:
 - **Gatekeeper Object Name:** The name of the Gatekeeper you are configuring. This property is used by the Gatekeeper Manager as the object name to differentiate other Gatekeepers.
 - **IOR File:** Enter a file name by typing a name, or click Browse to locate and specify a file. Entering a file name causes the Gatekeeper to put the Gatekeeper IOR in the specified file, instead of in the default file, `gatekeeper.ior`.
 - **Exterior Host Addresses:** You can enter an IP address at which all the exterior listeners listen when Gatekeeper is running on a dual-homed host.
 - **Exterior Host:** Sets the exterior IP address to the IP address you specify. If you have two IP addresses and do not specify which is to be the exterior, it is dynamically assigned by the operating system. For dual- or multi-homed hosts, you must set the Gatekeeper's exterior IP address to be the same as the web server's IP address.
 - **Exterior Proxy Host:** Sets an exterior proxy IP address for Gatekeeper to the IP address you specify when a NAT device is running in front of Gatekeeper. When you specify the proxy IP address, any IOR generated by the Gatekeeper will use this IP address. If this is not specified, the IOR uses the exterior IP address.
 - **Interior Host Addresses:** The interior listeners usually listen at the Intranet side to forward messages from the Intranet to the Internet. You can enter an IP address for the Interior Host and Interior Proxy Host.

- **Interior Host:** Sets the interior IP address to the IP address you specify when Gatekeeper is running on a dual-homed host. If you have two IP addresses and do not specify which is to be the interior, it is dynamically assigned by the operating system.
- **Interior Proxy Host:** Sets a proxy IP address for Gatekeeper to the IP address you specify when a NAT device is running behind Gatekeeper. When you specify the proxy IP address, any IOR generated by the Gatekeeper will use this IP address. If this is not specified, the IOR uses the interior IP address.
- **IIOP Proxy Services:** Checking these boxes enables the Callback Services and Location Services, respectively. After you've enabled these services, you can set their properties.
- **Enable Callback Service:** This property disables or enables the callback functionality in the Gatekeeper.
- **Enable Location Service:** This property enables or disables the location service in the Gatekeeper. The location service (not to be confused with the CORBA Location Service) in Gatekeeper is provided for clients, like applets, that are not able to talk OS Agent, to do a Bind operation. If this property is set to false, the client gets NO_PERMISSION when it tries to perform a Bind operation through Gatekeeper.

3 Click Apply.

Configuring IIOP Proxy Services

Under the IIOP Proxy Service, you can configure general IIOP Proxy services.

Configuring general IIOP Proxy Services

The General IIOP Proxy Services are categorized as follows: Request Forwarding Service, Callback Service, and Location Service.

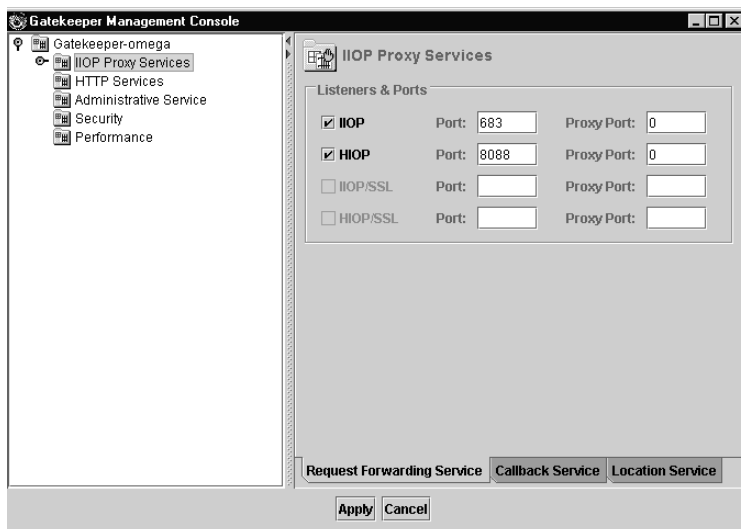
Request Forwarding Service

You can specify the ports and proxy ports for IIOP, HIOP, IIOP/SSL, and HIOP/SSL. Checking the box preceding a protocol, enables it. You can configure its properties after you've enabled it.

To configure Request Forwarding Service properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure.
- 2 Click the Configure button, then select IIOP Proxy Service from the list of Gatekeeper services.

3 Select the Request Forwarding Service tab.



4 Specify the appropriate properties.

If the Listener is an IIOP listener, the default port is 683, which is the default port for IIOP traffic, assigned by IANA. If the Listener is an HIOP listener, the default port is 8088.

5 Click Apply.

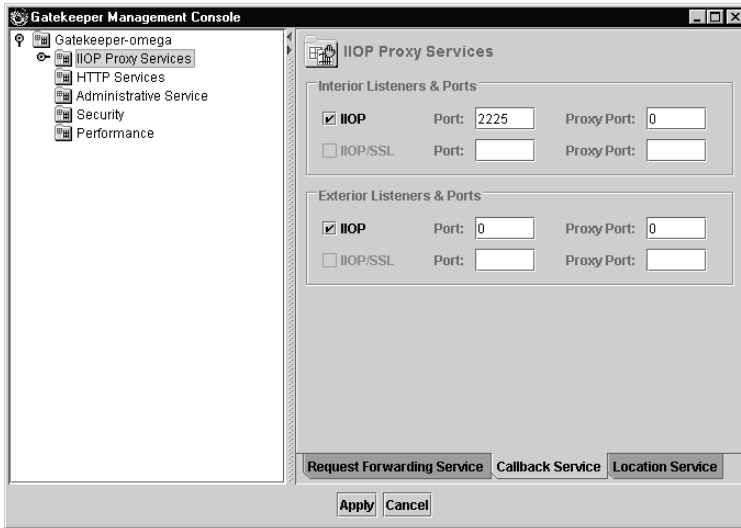
Callback Service

You can specify the interior and exterior ports and proxy ports at which the Callback Listeners listen for IIOP and IIOP/SSL. Checking the box preceding a protocol, enables it. You can configure its properties after you've enabled it.

To configure Callback Service properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure.
- 2 Click the Configure button, then select IIOP Proxy Service from the list of Gatekeeper services.

3 Select the Callback Service tab.



4 Specify the appropriate properties.

5 Click Apply.

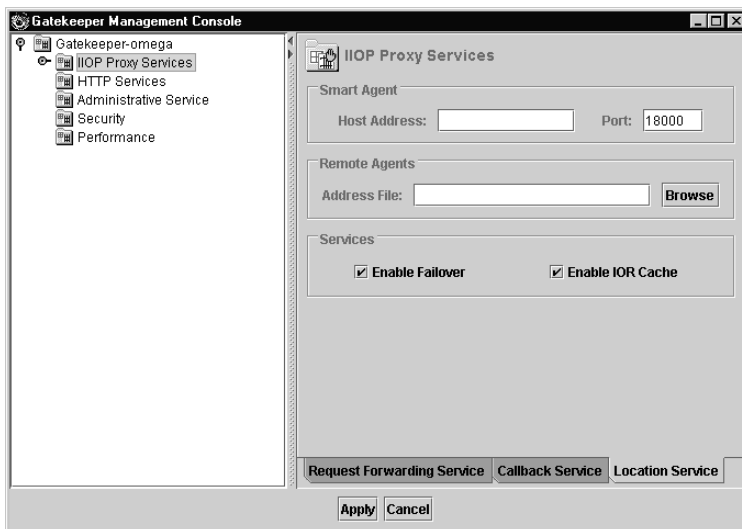
Location Service

The Gatekeeper provides location services for clients that are unable to communicate directly with an OS Agent because of Java sandbox security or existing firewalls. Disabling the location service facilities prevents clients from “binding” to the server through Gatekeeper; it also affects Gatekeeper’s ability to create connections to servers. It is recommended that you never disable the location service.

To configure Location Service properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure.
- 2 Click the Configure button, then select IIOP Proxy Service from the list of Gatekeeper services.

3 Select the Location Service tab.



4 Specify the appropriate properties:

- **Smart Agent:** Defines with which Smart Agent Gatekeeper registers (as an IP address and port) when the Gatekeeper is running outside the firewall or on another subnet of inter-servers.
- **Remote Agents:** Defines with which Smart Agent Gatekeeper registers (as an address file) when the Gatekeeper is running outside the firewall or on another subnet of inter-servers.
- **Services:** Controls the behavior of the Bind action. When Enable Failover is checked, it indicates that Gatekeeper should talk to another OS Agent if the current one crashes. Checking Enable IOR Cache directs Gatekeeper to cache the IOR of any server object it binds.

5 Click Apply.

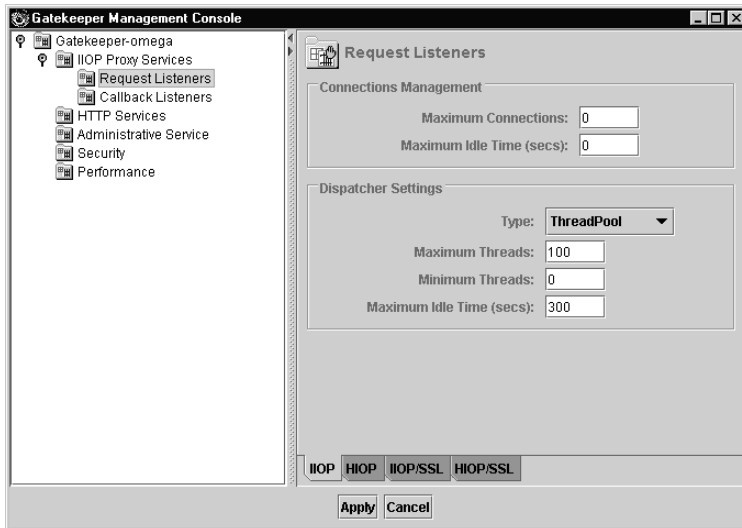
Request Listeners

If you have enabled IIOP, HIOP, IIOP/SSL, or HIOP/SSL Request listeners, you can further configure them. For IIOP and IIOP/SSL, you can specify connection management properties and Dispatcher settings. For HIOP and HIOP/SSL, you can specify connection management properties.

To configure Request listeners:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.

- 2 Double-click IIOP Proxy Service from the list of Gatekeeper services, then choose Request Listeners from the list.



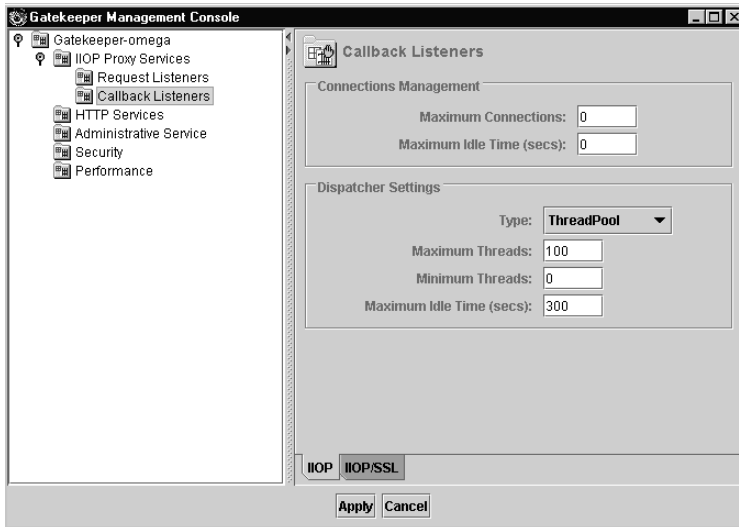
- 3 Choose the appropriate tab: IIOP, HIOP, IIOP/SSL, or HIOP/SSL.
- 4 Specify the appropriate properties.
 - **Connections Management Maximum Connections:** Specifies how many network connections Gatekeeper can accept.
 - **Connections Management Maximum Idle Time:** Specifies the amount of time a connection can be idle before it is considered stale and dropped.
 - **Dispatcher Settings Type:** Specify Thread Pool or Thread Session. If you specify Thread Pool, you can specify the thread pool's maximum and minimum threads and maximum idle time.
 - **Dispatcher Settings Maximum Threads:** Specifies the upper limit of the thread pool.
 - **Dispatcher Settings Minimum Threads:** Specifies the lower limit of the thread pool.
 - **Dispatcher Settings Maximum Idle Time:** Specifies the amount of time a thread can be idle before it is returned to the thread pool.
- 5 Click Apply.

Callback Listeners

If you have enabled Callback listeners, you can further configure them. For IIOP and IIOP/SSL Callback listeners, you can specify connection management properties and Dispatcher settings.

To configure Callback listeners:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.
- 2 Double-click IIOP Proxy Service from the list of Gatekeeper services, then choose Callback Listeners from the list.



- 3 Choose the appropriate tab: IIOP or IIOP/SSL.
- 4 Specify the appropriate properties.
- 5 Click Apply.

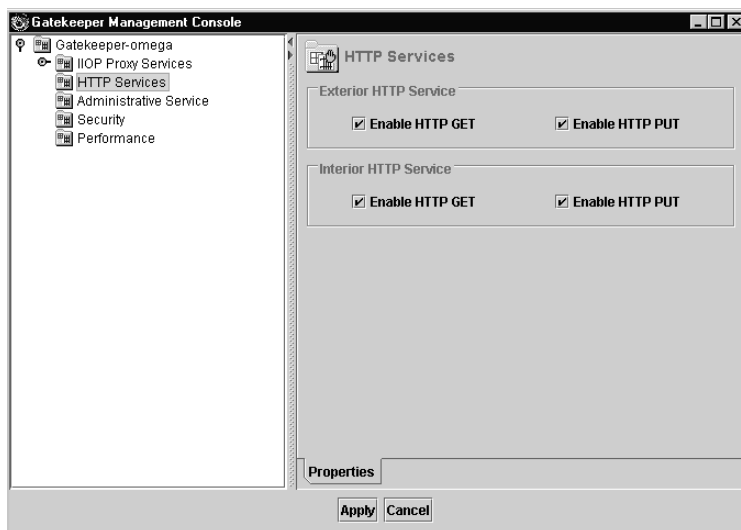
Configuring HTTP Service

The way you configure HTTP Service properties determines Gatekeeper's web access policy. You can enable GET and PUT statements for exterior and interior HTTP Services.

To configure the HTTP Service:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.

- 2 Choose HTTP Service from the list of Gatekeeper services.



- 3 Specify the appropriate properties.
- 4 Click Apply.

Configuring Administrative Service

The Administrative Service listens to the console itself. You can specify the address for the host and proxy host for the Administrative POA as well as the Listeners and Ports for IIOP and HIOP Administrative POA.

To configure the Administrative Service:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.

- 2 Choose Administrative Service from the list of Gatekeeper services.



- 3 Specify the appropriate properties.
- 4 Click Apply.

Configuring Security

You can set basic Security properties, manage access certificates, and create access rules.

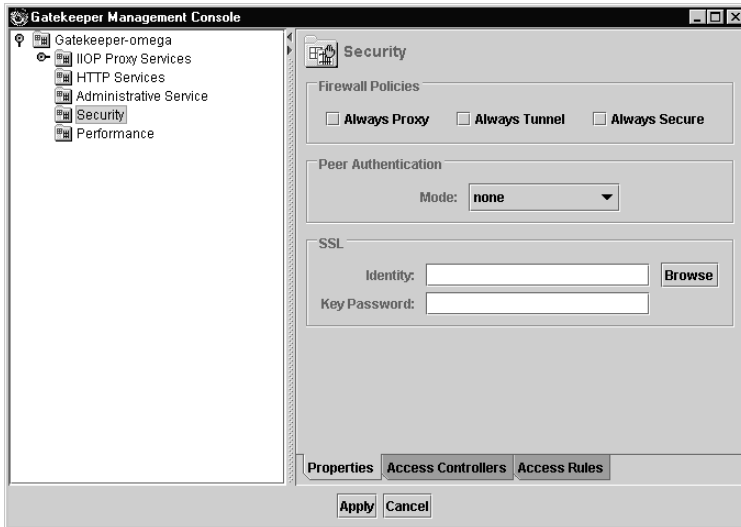
Configuring basic Security properties

Under basic properties, you can configure firewall policies, peer authentication modes, and manage identity/key password pairs.

To configure basic Security properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.

- 2 Choose Security from the list of Gatekeeper services, then choose the Properties tab.



- 3 Specify the appropriate properties. The SSL section allows you to install certificate chains on Gatekeeper.

You can specify the following Peer Authentication mode:

- **None:** No authentication.
- **Request:** The server wants to see the client's credential, but if the server doesn't receive the credential, it is not a Security error.
- **Request_and_trust:** If the server receives the client's credential, it will do the authentication based on the known trustpoint server. If the authentication fails, it is considered an error. If the credential is not received, it is not consider an error.
- **Require:** The server must receive the client's credential, otherwise, it is a Security error.
- **Require_and_trust:** The server must receive the client's credential and it must match the server trustpoint, otherwise it is an error.

- 4 Click Apply.

Managing access control

Gatekeeper provides rule-based access control based on an access control list (ACL). This ACL can deny or grant accessibility based on the following:

- Operation
- Signed by
- Server's host/port

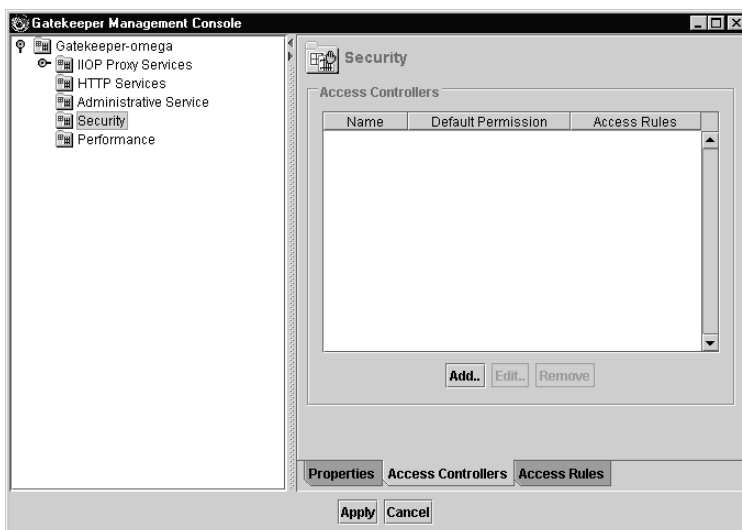
- Server's subnet
- Client's host/port
- Client's subnet

All the rules are evaluated in the order in which you specified them. Action is taken based on the first matched rule. If there is not a matched rule, the default action you specify is taken.

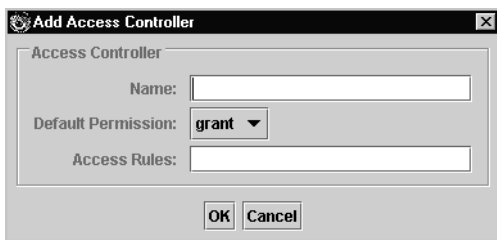
You can add, edit, and remove access controllers using this panel.

To manage access controllers:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.
- 2 Choose Security from the list of Gatekeeper services, then choose the Access Controllers tab.



- 3 To add an access controller, choose the Add button, then specify the appropriate properties.



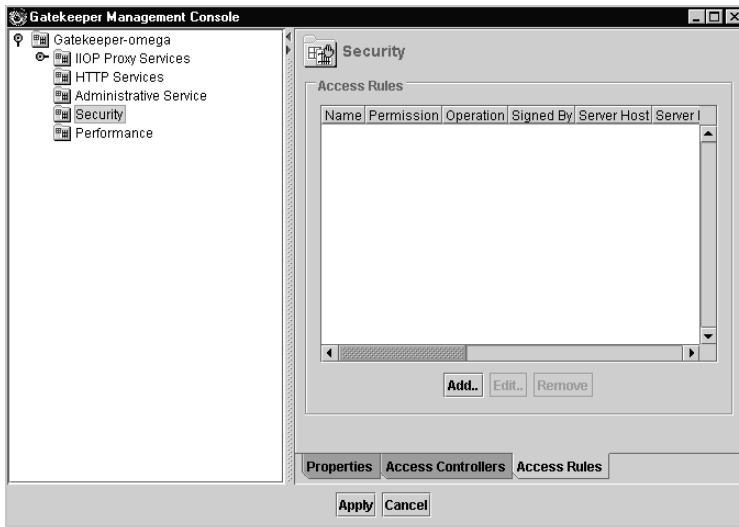
- 4 To edit or remove an access controller, select it from the list and choose the appropriate button.
- 5 Click Apply.

Managing access rules

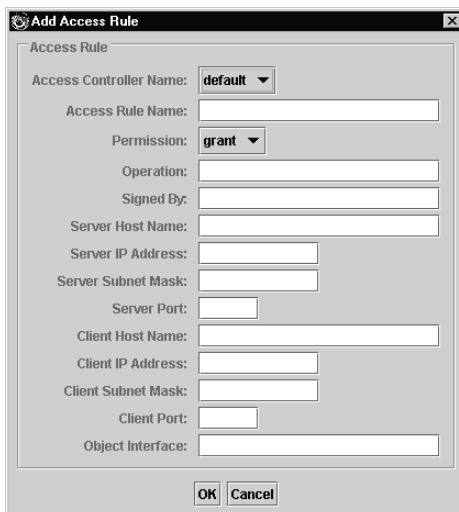
You can add, edit, and remove the access rules for the access controllers.

To manage access rules:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.
- 2 Choose Security from the list of Gatekeeper services, then choose the Access Rules tab.



- 3 To add an access rule, choose the Add button, then specify the appropriate properties.



- 4 To edit or remove an access rule, select it from the list and choose the appropriate button.
- 5 Click Apply.

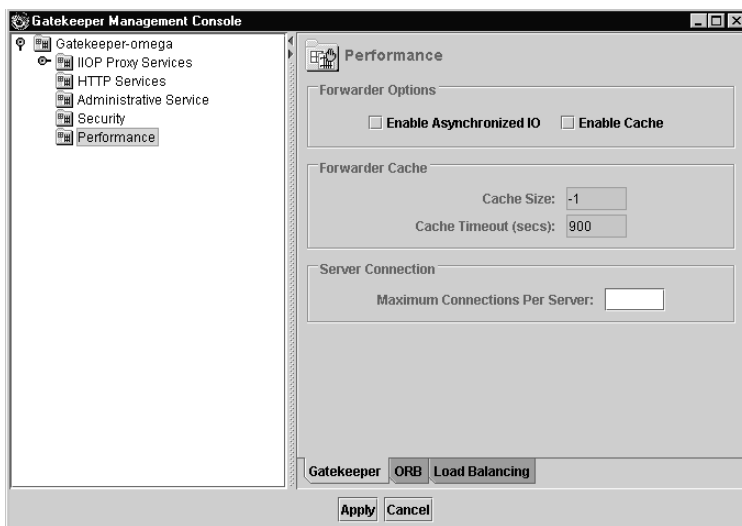
Configuring properties for performance

You can set Gatekeeper, ORB, and Load Balancing properties to improve performance.

Configuring Gatekeeper performance properties

To configure Gatekeeper performance properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.
- 2 Double-click Performance from the list of Gatekeeper services, then choose the Gatekeeper tab.



- 3 Specify the appropriate properties, then click Apply when you're finished.
- **Forwarder:**
 - **Cache Size:** Specifies the size of the cache. The value, -1, disables the cache. The value, 0, indicates an unlimited cache size. Enabling the cache enhances performance, however, in some cases, it might decrease the throughput.
 - **Cache Timeout:** Specifies how long Gatekeeper holds the idled cached information. That is, if the cached information has not been used for the specified period of time, it will be garbage collected. The unit is seconds.

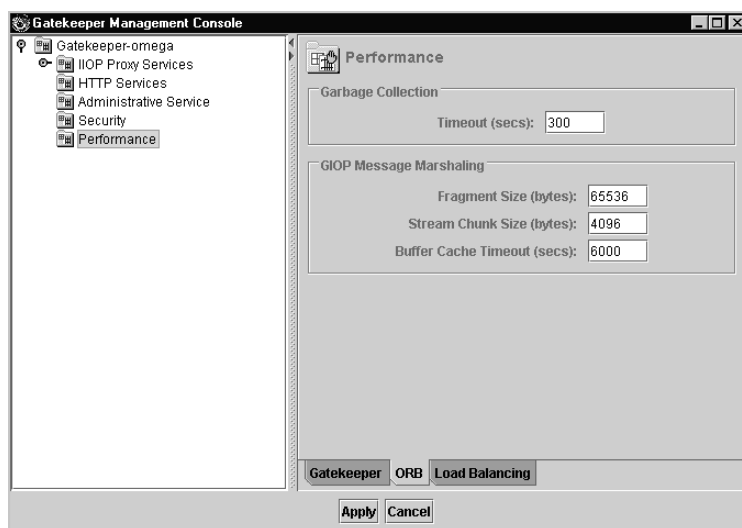
- **Enable Asynchronized IO:** Enables or disables the asynchronized input/output feature in the Gatekeeper. This feature increases the throughput for large number of clients, however, it might reduce the performance.
- **Server Connection:** Specifies how many connections Gatekeeper can have for one server object. This property may improve the bandwidth between Gatekeeper and a heavily referenced server object. The default value is one.

You can tune these properties at runtime to achieve the best configuration for your environment.

Configuring ORB performance properties

You can set the ORB garbage collection and GIOP message marshalling properties of the ORB on which Gatekeeper is running. To configure ORB performance properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.
- 2 Double-click Performance from the list of Gatekeeper services, then choose ORB from the list.



- 3 Specify the appropriate properties.

The Fragment Size and Stream Chunk Size must be a power of two. Their minimum value is 32. There is no maximum value. Fragment Size must be greater than or equal to Stream Chunk Size; must be a multiple of Stream Chunk Size.

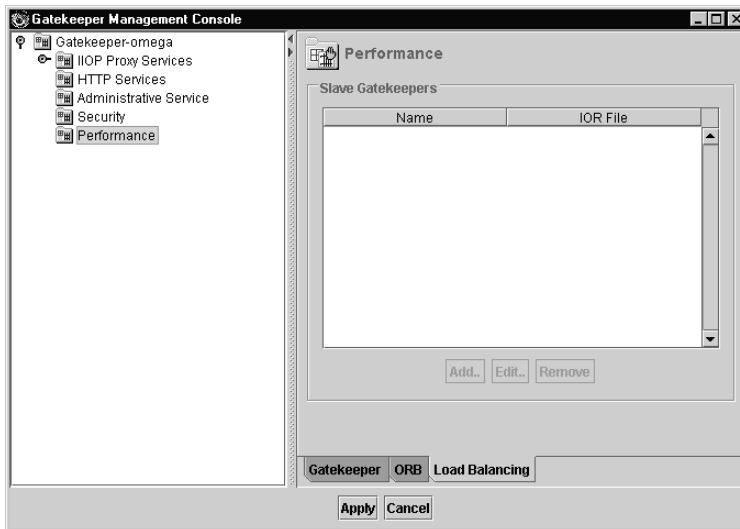
- 4 Click Apply when you're finished.

Configuring Load Balancing performance properties

You can configure Gatekeeper so that there's a master Gatekeeper (running as Dispatcher) and one or more slave Gatekeepers (running as service-loaders) are clustered together. The master Gatekeeper is responsible for balancing the load among the slave Gatekeepers. Use this panel to Add, Edit, and Remove slave Gatekeepers.

To configure load balancing performance properties:

- 1 Start the Gatekeeper Management Console for the Gatekeeper you want to configure. Click the Configure button.
- 2 Double-click Performance from the list of Gatekeeper services, then choose Load Balancing from the list.



- 3 To add a slave Gatekeeper, choose the Add button, then specify its name and IOR file.



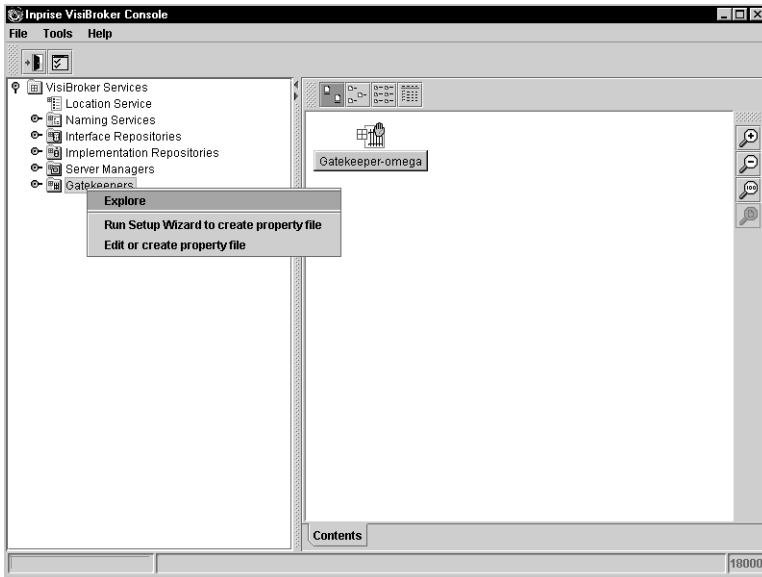
- 4 To edit or remove a slave Gatekeeper, select it from the list and choose the appropriate button.
- 5 Click Apply.

Creating and editing Gatekeeper property files

You can create and edit Gatekeeper property files using the Gatekeeper Management Console.

To create or edit a Gatekeeper property file:

- 1 Right-click the Gatekeepers folder. A menu appears.



- 2 Choose the Edit Or Create Property File command. A dialog appears, prompting you to select a property file. You can pick any property file to edit.
- 3 To create a new property file, enter a new property file name in the File name field.
- 4 Click the Apply button to save any changes.

Exiting the Gatekeeper Management Console

You can close the Gatekeeper Management Console at any time. If you have not saved your configuration information, the Gatekeeper Management Console prompts you to do so before exiting. The configuration you build with the Gatekeeper Management Console is not accessible to Gatekeeper unless and until you store it in a properties file.

Note If the Gatekeeper starts while you are modifying the configuration in an existing properties file, the Gatekeeper will start with the properties you most recently saved in that properties file.

To exit the Gatekeeper Management Console:

- From the File menu, choose Exit.

If there is unsaved information in the Gatekeeper Management Console, a dialog box appears prompting you to save before exiting. After you respond to this prompt, the Gatekeeper Management Console closes.



Gatekeeper properties

This appendix describes the properties that may be set for Gatekeeper.

Exterior Server Engine properties

This table lists the exterior properties.

Property	Default	Description
<code>vbroker.se.exterior.host=null</code>		The exterior server engine host address, which can be set in the <Basic Properties> Panel of the Gatekeeper.
<code>vbroker.se.exterior.proxyHost=null</code>		Network Address Translation (NAT) devices hide the actual IP address and/or port number in the network by changing the IP address and/or in the IP packet. You can set the Exterior Proxy Host in the Basic Property Panel to set the <code>vbroker.se.exterior.proxyHost</code> property to the value defined by the NAT. When you have callback enabled, the callback proxy host (<code>vbroker.gatekeeper.backcompat.callback.proxyHost</code>) will be set to equal this property.
<code>vbroker.se.exterior.scms=ex-iiop,ex-hiop</code>		
<code>vbroker.se.exterior.type=gatekeeper</code>		

ex-hiop listener properties

The following `vbroker.se.exterior.scm.ex-hiop` properties specify the behavior of the ex-hiop listener. The ex-hiop listener is an HIOP listener. The default port is 8088. The threading policy must always be `ThreadSession`.

Property
<code>vbroker.se.exterior.scm.ex-hiop.dispatcher.type</code>
<code>vbroker.se.exterior.scm.ex-hiop.listener.port</code>
<code>vbroker.se.exterior.scm.ex-hiop.listener.proxyPort</code>
<code>vbroker.se.exterior.scm.ex-hiop.listener.type</code>
<code>vbroker.se.exterior.scm.ex-hiop.manager.connectionMax</code>
<code>vbroker.se.exterior.scm.ex-hiop.manager.connectionMaxIdle</code>
<code>vbroker.se.exterior.scm.ex-hiop.manager.type</code>
<code>vbroker.se.exterior.scm.ex-hiop.root</code>
<code>vbroker.se.exterior.scm.ex-hiop.servlet.orb.class</code>
<code>vbroker.se.exterior.scm.ex-hiop.servlet.orb.GET</code>
<code>vbroker.se.exterior.scm.ex-hiop.servlet.orb.load</code>
<code>vbroker.se.exterior.scm.ex-hiop.servlet.orb.PUT</code>
<code>vbroker.se.exterior.scm.ex-hiop.servletList</code>

ex-iiop listener properties

The following `vbroker.se.exterior.scm.ex-iiop` properties specify the behavior of the ex-iiop listener. The ex-iiop listener is an IIOP listener. The default port is 683. The default threading policy is `ThreadPool`.

Property
<code>vbroker.se.exterior.scm.ex-iiop.dispatcher.threadMax</code>
<code>vbroker.se.exterior.scm.ex-iiop.dispatcher.threadMaxIdle</code>
<code>vbroker.se.exterior.scm.ex-iiop.dispatcher.threadMin</code>
<code>vbroker.se.exterior.scm.ex-iiop.dispatcher.type</code>
<code>vbroker.se.exterior.scm.ex-iiop.listener.type</code>
<code>vbroker.se.exterior.scm.ex-iiop.listener.port</code>
<code>vbroker.se.exterior.scm.ex-iiop.listener.proxyPort</code>
<code>vbroker.se.exterior.scm.ex-iiop.listener.giopVersion</code>
<code>vbroker.se.exterior.scm.ex-iiop.manager.connectionMax</code>
<code>vbroker.se.exterior.scm.ex-iiop.manager.connectionMaxIdle</code>
<code>vbroker.se.exterior.scm.ex-iiop.manager.type</code>

Interior Server Engine properties

This table lists the Interior properties.

Property	Default	Description
<code>vbroker.se.interior.type=normal</code>		
<code>vbroker.se.interior.host=null</code>		
<code>vbroker.se.interior.proxyHost=null</code>		You can set the Interior Proxy Host in the Basic Property Panel to set this property. It might be used if you have a NAT running between the GateKeeper and the server to hide the server host's address.
<code>vbroker.se.interior.scm.in-iiop.dispatcher.threadMax=100</code>		
<code>vbroker.se.interior.scm.in-iiop.dispatcher.threadMaxIdle=300</code>		
<code>vbroker.se.interior.scm.in-iiop.dispatcher.threadMin=0</code>		
<code>vbroker.se.interior.scm.in-iiop.dispatcher.type=ThreadPool</code>		
<code>vbroker.se.interior.scm.in-iiop.listener.type=IIOP</code>		
<code>vbroker.se.interior.scm.in-iiop.listener.giopVersion=1.2</code>		
<code>vbroker.se.interior.scm.in-iiop.listener.port=0</code>		
<code>vbroker.se.interior.scm.in-iiop.listener.proxyPort=0</code>		
<code>vbroker.se.interior.scm.in-iiop.manager.connectionMax=0</code>		
<code>vbroker.se.interior.scm.in-iiop.manager.connectionMaxIdle=0</code>		
<code>vbroker.se.interior.scm.in-iiop.manager.type=Socket</code>		
<code>vbroker.se.interior.scms=in-iiop</code>		

Properties that support bidirectional communications

This table lists the properties that support bidirectional communications.

Properties that support pass-through connections

NOTE: These properties are evaluated only once -- when the SCMs are created. In all cases, the `exportBiDir` and `importBiDir` properties on the SCMs are given priority over the `enableBiDir` property. In other words, if both properties are set to conflicting values, the SCM-specific properties will take effect. This allows you to set the `enableBiDir` property globally and specifically turn off bidirectionality in individual SCMs.

Property	Default	Description
<code>vbroker.orb.enableBiDir</code>		<p>You can selectively make bidirectional connections. If the client defines <code>vbroker.orb.enableBiDir=client</code> and the server defines <code>vbroker.orb.enableBiDir=server</code> the value of <code>vbroker.orb.enableBiDir</code> at the gatekeeper determines the state of the connection. Values of this property are: <code>server</code>, <code>client</code>, <code>both</code> or <code>none</code>.</p> <p>Note: Just as you can selectively enable bidirectional communication on a per-SCM basis, you can also selectively enable bidirectional communication on the Gatekeeper. For example, if you set the <code>vbroker.se.exterior.scm.ex-iiop.manager.importBiDir</code> property to <code>true</code>, the Gatekeeper will accept bidirectional connections from the client. Setting the <code>vbroker.se.exterior.scm.ex-iiop.manager.exportBiDir</code> property to <code>true</code> will cause the Gatekeeper to request bidirectional connections with the server.</p>

Properties that support pass-through connections

This table lists the properties that support pass-through connections.

Property	Default	Description
<code>vbroker.gatekeeper.enablePassthru</code>	<code>false</code>	
<code>vbroker.gatekeeper.passthru.blockSize</code>	<code>16384</code>	This property specifies the buffer size that the channel uses for every read and write. A high value handles large messages with a single read and write, but increases the resources used by a single channel. A low value will optimize resource utilization, while degrading performance due to multiple read and writes.
<code>vbroker.gatekeeper.passthru.connectionTimeout</code>	<code>0 (never times out)</code>	This property specifies the amount of time a given channel will wait before it stops waiting for connections and shuts down the channel.

Property	Default	Description
<code>vbroker.gatekeeper.passthru.logLevel</code>	0	This property enables logging for the pass-through component
<code>vbroker.gatekeeper.passthru.streamTimeout</code>	2000	This property specifies the amount of time in milliseconds an established channel will wait for messages before it shuts itself down.

Administration properties

This table lists the administration properties. This is another HIOP listener which is intended to be used by the intranetpass-through server side. The default port is 9091.

Property	Default	Description
<code>vbroker.se.iiop_tp.type=normal</code>		
<code>vbroker.se.iiop_tp.scms=iiop_tp,hiop_ts</code>		
<code>vbroker.se.iiop_tp.scm.hiop_ts.listener.port=9091</code>		
<code>vbroker.se.iiop_tp.scm.hiop_ts.servletList=orb</code>		
<code>vbroker.se.iiop_tp.scm.hiop_ts.servlet.orb.class=com.inprise.vbroker.HIOP.servlets.ORBServlet</code>		
<code>vbroker.se.iiop_tp.scm.hiop_ts.servlet.orb.GET=true</code>		
<code>vbroker.se.iiop_tp.scm.hiop_ts.servlet.orb.PUT=true</code>		
<code>vbroker.se.iiop_tp.scm.hiop_ts.servlet.orb.load=false</code>		

Index

Symbols

... ellipsis 1-2
[] brackets 1-2
| vertical bar 1-2

A

access control 2-3, 3-13
access rules 3-15
administrative service 3-11
alwaysProxy 2-7
alwaysSecure 2-8
applets, Java 2-9
asynchronous IO 3-17
authentication 3-12

B

Bidirectional Communication 2-14

C

cache, forwarder 3-16
callback
 listeners 3-9
 ports 3-2
 service 3-6
chains, Gatekeeper 2-14
clients, configuring 2-7
client-side firewalls 2-2, 2-13
clusters 2-4
configuring
 client-centric runtime 2-7
 clients 2-7
 Gatekeeper at runtime 3-1
 Gatekeeper while not running 3-1
 POAs 2-11
 servers 2-6, 2-11
 servers at runtime 2-12
 VisiBroker runtime 2-6
connections
 managing 3-9
 server 3-17
Console
 Gatekeeper Management 3-2
 VisiBroker Console 3-2
contacting Borland 1-3
conventions
 platform 1-2
 platform icons 1-2
 typographic 1-2

CORBA documentation 1-3
creating properties files 3-19

D

developer support 1-3
Dispatcher settings 3-9
dispatchers 3-18
documentation
 additional 1-3
 CORBA 1-3
 OMG 1-3
dual-homed machines 2-2, 2-9

E

editing properties files 3-19
enablePassthru 2-8
exiting Gatekeeper Management Console 3-19
exterior
 host address 3-4
 IP addresses 3-2
 ports 3-2
 proxy host 3-4

F

fault tolerance 2-4
filtering
 port 2-2
 router 2-10
firewall policies 3-12
 setting global 2-12
 setting individual 2-11
firewalls 2-2
 client-side 2-2, 2-13
 server-side 2-2, 2-9
forward ports 3-2
forwarder cache 3-16

G

Gatekeeper
 and Smart Agent 2-18
 as IIOP proxy 2-11
 as NT service 2-20
 chains 2-14
 clusters 2-4
 command 2-19
 configuring at runtime 3-1
 configuring while not running 3-1
 master 2-4, 3-18

- performance guidelines 2-5
- performance properties 3-16
- properties
 - files 2-20
- restarting 3-1
- run as a servlet 2-3
- security features 2-18
- slave 2-4, 3-18
- starting 2-19
- starting the Gatekeeper 2-19
 - with Java applets 2-9
- Gatekeeper Management Console
 - exiting 3-19
 - starting 3-2
- GIOP Proxy server 2-1

H

- h option 2-20
- help option 2-20
- HIOP 2-3, 3-5, 3-8, 3-11
- HIOP/SSL 3-5, 3-8
- hosts, dual-homed 2-9
- HTTP 2-13
- HTTP service 3-10
- HTTP tunneling 2-7, 2-8, 2-13

I

- identity/password pairs 3-12
- IIOP 3-5, 3-6, 3-8, 3-9, 3-11
- IIOP proxy 2-11
- IIOP proxy services 3-5
- IIOP/HTTPS 2-7
- IIOP/SSL 2-7, 3-5, 3-6, 3-8, 3-9
- improving performance 3-16
- install option 2-20
- installation support 1-3
- installing
 - Gatekeeper as NT service 2-20
- interior
 - host address 3-4
 - proxy host 3-4
- IOR file 3-4
- IP addresses 3-2

J

- Java
 - properties A-1
- Java applets 2-9
- Java sandbox security 2-1, 2-2, 3-7

L

- listeners
 - callback 3-9

- request 3-8
- load balancing 2-4, 3-18
- location service 3-7

M

- managing connections 3-9
- master Gatekeepers 2-4, 3-18
- multiple Gatekeepers 2-14

N

- Naming Service
 - properties A-5
- NATs 2-3, 2-9
- Network Address Translation device 2-3, 2-9
- network interface cards 2-2
- NICs 2-2
- NT service 2-20

O

- OMG documentation 1-3
- options, Gatekeeper command 2-20
- ORB performance properties 3-17
- OS Agent 2-18, 3-7

P

- passthrough connection 2-8
- peer authentication 3-12
 - modes 3-13
- performance 2-5, 3-16
 - ORB properties 3-17
- platform designation with icons 1-2
- POAs
 - configuring 2-11
- port filtering 2-2
- Portable Object Adapter 2-11
- ports 3-2
- properties 2-4
 - files 2-4
 - creating 3-19
 - editing 3-19
 - Java A-1
 - Naming Service A-5
 - proxy 2-9
 - Server-Side Server Engine A-1, A-3, A-4
- properties file 2-20
- props option 2-20
- proxy properties 2-9
- PROXY_MODE_POLICY 2-8

Q

- Quality of Service 2-7
- quiet option 2-20
- quitting Gatekeeper Management Console 3-19

R

- remote agent 3-8
- request forwarding service 3-5
- request listeners 3-8
- request peer authentication mode 3-13
- request_and_trust peer authentication mode 3-13
- require peer authentication mode 3-13
- require_and_trust peer authentication mode 3-13
- restarting Gatekeeper 3-1
- router filtering 2-10
- runtime
 - configuring client-centric 2-7
 - configuring Gatekeeper at 3-1
 - configuring VisiBroker 2-6

S

- sandbox security, Java 2-1, 2-2
- scalability 2-5
- security 2-18, 3-12
- servers
 - configuring 2-6, 2-11
 - configuring at runtime 2-12
 - connections 3-17
- server-side firewalls 2-2
- Server-Side Server Engine
 - properties A-1, A-3, A-4
- service-loaders 3-18
- services
 - administrative 3-11
 - callback 3-6
 - HTTP 3-10
 - location 3-7
 - request forwarding 3-5
- servlets, running Gatekeeper as 2-3
- setup wizard 3-3

- shutting down Gatekeeper Management Console 3-19
- slave Gatekeepers 2-4, 3-18
- Smart Agent 2-18, 3-8
- starting
 - Gatekeeper 2-19
 - Gatekeeper Management Console 3-2
- support options 1-3

T

- TCP
 - endpoints 3-2
 - mechanism 2-9
- technical support 1-3
- tunneling 2-7, 2-8
- typographic conventions 1-2

U

- usage option 2-20

V

- vbroker.orb.alwaysProxy 2-7
- vbroker.orb.alwaysSecure 2-8

W

- web server integration 2-3
- web sites, Inprise 1-3
- What's New 1-1

Y

- Y2K issues 1-4
- year 2000 issues 1-4