IBM

WebSphere Scalability: WLM and Clustering

Using WebSphere Application Server Advanced Edition

- How to scale IBM WebSphere Application Server Advanced Edition
- Sample configurations help you to start your WebSphere site
- Examines several techniques, including WLM

Ken Ueno
Tom Alcott
Jeff Blight
Jeff Dekelver
Daniel Julin
Colin Pfannkuch
Tzueing Shieh

Redbooks

ibm.com/redbooks

SG24-6153-00

International Technical Support Organization

WebSphere Scalability: WLM and Clustering

Using WebSphere Application Server Advanced Edition

September 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix J, “Special notices” on page 565.

First Edition (September 2000)

This edition applies to:

- IBM WebSphere Application Server Advanced Edition V3.021 for AIX, Windows NT and OS/400
- IBM WebSphere Application Server Advanced Edition V3.5 for AIX and Windows
- IBM HTTP Server for OS/400 V4R4
- IBM HTTP Server V1.3.6.2 for AIX and Windows NT (applies to WebSphere V3.021)
- IBM HTTP Server V1.3.6.12 for AIX and Windows (applies to WebSphere V3.5)
- IBM Java Development Kit V1.1.7 and 1.1.8 for OS/400
- IBM Java Development Kit V1.1.8 for AIX (applies to WebSphere V3.021)
- IBM Java Development Kit V1.2.2 for AIX (applies to WebSphere V3.5)
- IBM Java Development Kit V1.1.7 for Windows (applies to WebSphere V3.021)
- IBM Java Development Kit V1.2.2 for Windows (applies to WebSphere V3.5)
- IBM DB2 UDB V6.1 FP4 for AIX and Windows
- IBM Network Dispatcher V3.0 for AIX

for use with the AIX V4.3.3, OS/400 V4R4, OS/400 V4R5, Windows NT 4.0 SP5 and Windows 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Preface

This redbook discusses various options for making your applications more accessible to more people based on WebSphere Application Server, Advanced Edition. The objective of this book is to explore how the basic configuration can be extended to provide more computing power, by better exploiting the power of each machine, and by using multiple machines.

It examines a number of scaling techniques, including:

- Cloning and pooling of multiple Java Virtual Machines
- Distributing the load from one Web server to multiple application servers, using Servlet Redirector and OSE transports
- Using IBM Network Dispatcher to distribute the load between multiple Web servers
- Using the Workload Management facility (WLM) to distribute load at the EJS (Enterprise Java Services) level in WebSphere (servlets and JSPs as well as EJBs)
- Distributing session states using the Session Clustering facility

This book shows step-by-step procedures for the UNIX, Windows NT and AS/400 platforms. Both simple and advanced configurations are covered.

This has been done using WebSphere Application Server V3.021 (OS/400, AIX and Windows NT) and V3.5 (AIX and Windows) Advanced Edition.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Kenichiro Ueno is an Advisory I/T Specialist at the International Technical Support Organization, Raleigh Center. He manages residencies and produces redbooks. His most recent publication was *WebSphere V3 Performance Tuning Guide*. Before joining the ITSO, he worked in Internet Systems, IBM Japan Systems Engineering Co., Ltd. in Japan as an I/T Specialist.

Tom Alcott is an Advisory I/T specialist in the United States. He has been a member of the World Wide WebSphere Technical Sales Support team since its inception. Before he started working with WebSphere, he was a systems engineer for IBM's Transarc Lab supporting TXSeries. His background

includes over 18 years of application design and development on both mainframe-based and distributed systems. He has written and presented extensively on a number of WebSphere runtime and security issues.

Jeff Blight is a Senior IT Specialist working as the e-business technology specialist in the AS/400 Technical Support for the Enterprise Server Group in IBM UK.

Jeff Dekelver is a Software Engineer in the WebSphere Advanced Server for the AS/400, in Rochester, Minnesota. He has five years of experience in AS/400, and four years in Java. He holds a Bachelor of Science degree in Computer Science from the University of Wisconsin-LaCrosse. His areas of expertise include Java, WebSphere including Work Load Management, and object-oriented design.

Daniel Julin is a Senior Consulting I/T Specialist in the United States. He is currently attached to the IBM WebSphere Execution Team, a small team from WebSphere Development that focuses on addressing the most difficult issues encountered in conjunction with the deployment of WebSphere to customers. He has over 15 years of experience in research and industry, in the areas of operating systems, distributed systems, transaction processing systems and networking.

Colin Pfannkuch is a Software Engineer at the AS/400 Support Center, Rochester, Minnesota. He is a member of the Support Center's Client Technology Application group. Part of this group's responsibilities is supporting WebSphere and Java on the AS/400. He joined IBM after graduating with a Bachelor of Science degree in Computer Science from Buena Vista University, Storm Lake, Iowa.

Tzueing Shieh is an Advisory I/T specialist in IBM US. He is a member of the World Wide WebSphere Technical Sales Support team. His career at IBM started in 1987 when he joined IBM Software Development lab at Research Triangle Park in North Carolina. He worked mainly on software development on IBM mainframes and workstations.

Thanks to the following people for their invaluable contributions to this project:

Tom Barlen
Gail Christensen
Shawn Walsh
Margaret Ticknor
Mike Haley
International Technical Support Organization, Raleigh Center

Thanks to the following IBM employees:

Amber Roy-Chowdhury, WebSphere Development, IBM Transarc Lab
Joe Bockhold, WebSphere Development, Rochester
Jason R McGee, WebSphere Development, Raleigh
Michael Morton, WebSphere Development, Raleigh
Michael Fraenkel, WebSphere Development, Raleigh
Jerry Cuomo, Manager, WebSphere Performance, Raleigh
Ruth Willenborg, Manager, WebSphere Performance, Raleigh
Chris Forte, WebSphere Performance, Raleigh
Charlie Bradley, WebSphere Performance, Raleigh
Graeme N. Dixon, STSM, WebSphere Development, IBM Transarc Lab
Raj Nagratnam, WebSphere Development, Raleigh
Tianyu Jiang, WebSphere Development, IBM Transarc Lab
Gabe Montero, WebSphere Development, Raleigh
Steve Roma, WebSphere Test, Raleigh
Shweta Mathur, WebSphere Test, Raleigh
Melissa Passow, WebSphere Solution Integration, Rochester
Tom Gissel, WebSphere Solution Integration, Rochester
Rohit Singh, WebSphere Solution Integration, Rochester
Keys Botzum, WebSphere Consulting Services, Pittsburgh
Ken Hygh, WebSphere Enablement, Raleigh
Sung-Ik Son, WebSphere Enablement, Raleigh
Steve King, System Administration, Raleigh

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 577 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Part 1. WebSphere scalability overview

Chapter 1. WebSphere scalability overview and key concepts

This chapter provides a conceptual overview of the goals and issues associated with scaling applications in the WebSphere environment. The following chapter will present the various techniques that can be brought to bear to implement scaling.

1.1 Objectives

As outlined by the product documentation, a minimal, typical configuration for a WebSphere Advanced Edition (AE) installation is depicted in Figure 1. There is a single Web (HTTP) Server, and a single Application Server, and both are co-resident on a single machine. Obviously, the performance characteristics of this setup are limited by the power of the machine on which it is running, and by various constraints inherent in the configuration itself and in the implementation of WebSphere AE. A discussion of the various components in the WebSphere runtime is provided in 1.2, “WebSphere run-time components and client types” on page 6.

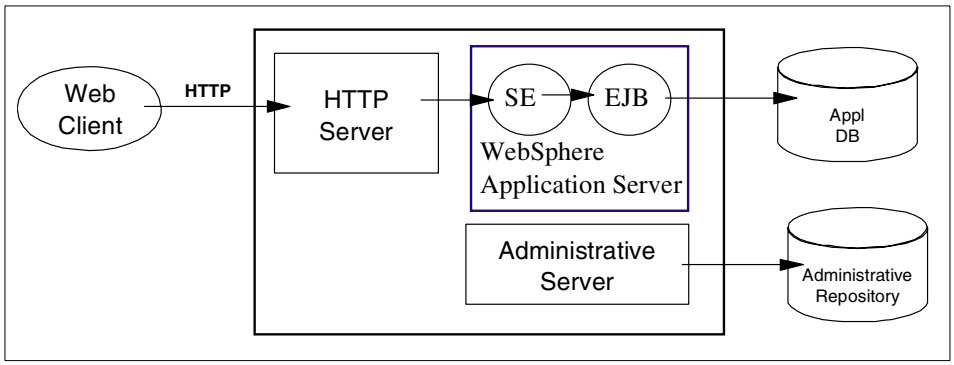

Figure 1. Basic WebSphere configuration

The objective of this book is to explore how this basic configuration can be extended to provide more computing power, by better exploiting the power of each machine, and by using multiple machines. Specifically, we are interested in defining system configurations that exhibit the following properties:

- Scalability: The proposed configurations should allow the overall system to service a higher client load than that provided by the simple basic configuration. Ideally, it should be possible to service any given load, simply by adding the appropriate number of servers or machines.

- Load-balancing: The proposed configurations should ensure that each machine or server in the configuration processes a fair share of the overall client load that is being processed by the system as a whole. In other words, it would not do for one machine to be overloaded, while another machine is mostly idle. If all machines are of roughly the same power, each should process a roughly equal share of the load. If various machines are of different power, each should process a portion of the load in proportion to its relative processing power.

  Furthermore, if the total load changes over time, the system should naturally adapt itself to maintain this load-balancing property, regardless of the actual total magnitude of this load. In other words, all machines may be used at 50% of their capacity, or all machines may be used at 100% of their capacity.

- Failover: The proposition to have multiple servers (potentially on multiple independent machines) naturally leads to the potential for the system to provide failover. That is, if any one machine or server in the system were to fail for any reason, the system should continue to operate with the remaining servers. The load-balancing property should ensure that the client load gets redistributed to the remaining N-1 servers, each of which will henceforth process a proportionately slightly higher percentage of the total load. Of course, such an arrangement pre-supposes that the system is designed with some degree of over-capacity, so that N-1 servers are indeed sufficient to process the total expected client load.

  Ideally, the failover aspect should be totally transparent to clients of the system: when a server fails, any client that is currently interacting with that server should be automatically redirected to one of the remaining servers, without any interruption of service and without requiring any special action on the part of that client. In practice, however, most failover solutions may not be completely transparent. For example, a client that is currently in the middle of an operation when a server that fails, may receive an error from that operation, and may be required to retry (at which point the client would be connected to a different, still-available server). Or the client may observe a “hiccup” or delay in processing, before the processing of his requests resumes automatically with a different server. The important point for failover, is that each client, and the set of clients as a whole, are able to eventually continue to take advantage of the system and receive useful service, even if some of the servers fail and become unavailable. Conversely, when a previously-failed server is repaired and again becomes available, the system may transparently start using that server again to process a portion of the total client load.

The failover aspect is also sometimes called *fault-tolerance*, in that it allows the system to survive a variety of failures or faults. It should be noted however that failover is only one technique in the much broader field of fault-tolerance, and that no such technique can make a system 100% safe against any possible failure. The goal is to greatly minimize the probability of system failure, but it cannot be completely eliminated.

Note that in the context of discussions on failover, the term *server* most often refers to a physical machine (which is typically the type of component that fails). But we'll see below that WebSphere also allows for the possibility of one *server process* on a given machine to fail independently, while other processes on that same machine continue to operate normally.

In addition, we should also consider a number of secondary characteristics when evaluating a configuration:

- Dynamic changes to configuration: In certain configurations, it may be possible to modify the configuration on-the-fly without interrupting the operation of the system and its service to clients. For example, it may be possible to add or remove servers to adjust to variations in the total client load. Or it may be possible to temporarily stop one server to change some operational or tuning parameters, then restart it and continue to service client requests. Such characteristics, when possible, are highly desirable, since they enhance the overall manageability and flexibility of the system.
- Mixed configuration: In some configurations, it may be possible to mix multiple versions of a server or application, so as to provide for staged deployment and a smooth upgrade of the overall system from one software or hardware version to another. Coupled with the ability to make dynamic changes to the configuration, this property may be used to effect upgrades without any interruption of service.
- Fault isolation: In the simplest application of failover, we are only concerned with clean failures of an individual server, in which a server simply stops to function completely, but this failure has no effect on the health of other servers. But there are sometimes situations where one malfunctioning server may in turn create problems for the operation of other, otherwise healthy servers. For example, one mafunctioning server may hoard system resources, or database resources, or hold critical shared objects for extended periods of time, and prevent other servers from getting their fair share of access to these resources or objects. In this context, some configurations may provide a degree of *fault isolation*, in that they reduce the potential for the failure of one server to affect other servers.

- Security: The potential to distribute the processing responsibilities between multiple servers and in particular multiple machines, also introduces a number of opportunities for meeting special security constraints in the system. Different servers or types of servers may be assigned to manipulate different classes of data or perform different types of operations. And the interactions between the various servers may be controlled --- for example through the use of firewalls --- to prevent undesired accesses to data.

  This book does not specifically focus on security and firewall-related issues, but will try to point out the important observations as they pertain to various configurations.

1.2 WebSphere run-time components and client types

1.2.1 Run-time components

The WebSphere Application Server consists of the following components:

- Administrative Server
- Application Server
- Administrative Repository
- Administrative Console
- HTTP Server and HTTP Server Plugin

As its name implies the WebSphere Administrative Server (“Adminserver”) provides administrative functions for all nodes running in a WebSphere domain. The Adminserver is responsible for maintaining the configuration of a WebSphere domain, providing security services, Work Load Management, and monitoring the run-time state. The Adminserver process runs in its own JVM on each node in a WebSphere domain (“cluster”).

The Administrative Repository stores the configuration for all WebSphere Application Server components inside a WebSphere domain. Each Adminserver communicates changes in configuration and run-time state to all other Adminservers through the Administrative Repository. The Administrative Repository consists of a series of tables that are persisted as Entity Enterprise Java Beans by WebSphere (as of V3.021 there were 34 tables, though the number is subject to change in future versions of WebSphere).

An Application Server in WebSphere is the process that is used to run your servlet and/or EJB-based applications, providing both the servlet run-time

components (Servlet Engine, Web applications) and EJB run-time (EJB container). Like the Adminserver, each WebSphere Application Server runs in it's own JVM. The capability to create multiple Application Server processes in the WebSphere runtime is known as "cloning". Cloning an of applicaton server on a single physical machine provides one means of increasing throughput via the notion of "vertical scalability". Cloning can also be employed in WebSphere to support the concept of "horizontal scalability" where multiple processes are distributed across multiple physical machines. These concepts and other means of scalability will be discussed in more detail in Chapter 2, "Techniques for WebSphere WLM and clustering" on page 15.

The WebSphere Administrative Console provides an easy to use, graphical "window" to a WebSphere domain ("cluster"). The Administrative Console connects to one of the Adminservers running in a cluster and can be used change to the configuration or run-time state on any machine in a domain.

While the HTTP Server is not strictly part of the WebSphere runtime, WebSphere communicates with your HTTP server of choice: IBM HTTP Server powered by Apache, Microsoft Internet Information Server, Apache, Netscape Enterprise Server, and Lotus Domino via a plug-in. This plug-in communicates requests from the HTTP server to the WebSphere runtime.

1.2.2 Web clients and Java clients

The basic configuration shown in Figure 1 on page 3 above refers to a particular type of client for WebSphere characterized as a *Web client.* Such a client uses a Web browser to interact with the WebSphere Application Servers, through the intermediary of a Web server, that in turn invokes a servlet within WebSphere. Although Web clients constitute the majority of users of WebSphere today, WebSphere also supports another type of client characterized as a *Java client*. Such a client is a stand-alone Java program (possibly GUI-based, and possibly not), that uses the Java RMI/IIOP facilities to make direct method invocations on various EJB objects within a WebSphere application server, without going through an intervening Web server and servlet, as shown in Figure 2.

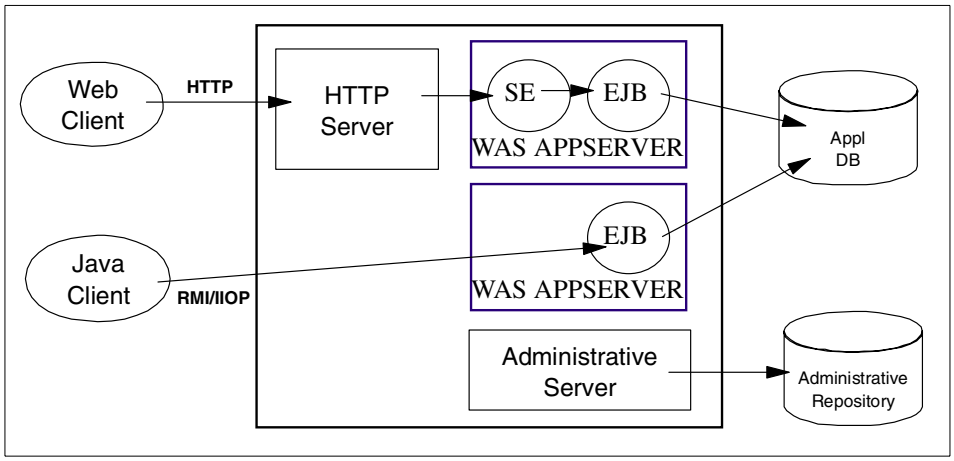

Figure 2. WebSphere configuration with Java clients

Much of the discussion of scalable configurations in this book is specific to Web clients, as they tend to focus on the interaction between Web servers and application servers, and how clients access the Web servers from the outside world.

1.3 Managing state among servers: sessions and affinities

All of the load distribution techniques discussed in this book rely, on one level or another, on using multiple copies of an application server and arranging for multiple consecutive requests from various clients to be serviced by different servers.

If each client request is completely independent of every other client request, then it does not matter if two requests are ever processed on the same server or not. However, in practice, there are many situations where all requests from an individual client are not totally independent. In many usage scenarios, a client makes one request, waits for the result, then makes one or more subsequent requests that depend upon the results received from the earlier requests.

Such a sequence of operations on behalf of one client falls into two categories:

Stateless: If the server that processes each request does so based solely on information provided with that request itself, and not based on information that it “remembers” from earlier requests. In other

words, the server does not need to maintain *state information* between requests.

Stateful: If the server that processes a request does need to access and maintain state information generated during the processing of an earlier request.

Again, in the case of stateless interactions, it does not matter if different requests are being processed by different servers. But in the case of stateful interactions, we must ensure that whatever server is processing a request, it has access to the state information necessary to service that request. This can be ensured either by arranging for the same server to process all the client requests that are associated with the same state information, or by arranging for that state information to be shared and equally accessible by all servers that may require it. And in that latter case, it is often advantageous to arrange for most accesses to the shared state to be performed, from the same server, so as to minimize the communications overheads associated with accessing the shared state from multiple servers.

This consideration for the management of stateful interactions will be a factor in many discussions of various load-distribution techniques throughout this book. It also requires the discussion of a specific facility, the *Session Management Facility*, and a new concept, *Server Affinity*.

1.3.1 HTTP sessions and the session management facility

In the case of an HTTP client interacting with a servlet, the state information associated with a series of client requests is represented as an *HTTP session*, and identified by a *session ID.* The session manager module that is part of each servlet engine, is responsible for managing HTTP sessions, providing storage for session data, allocating session IDs, and tracking the session ID associated with each client request, through the use of cookies or URL rewriting techniques.

The session manager provides for the storage of session-related information either in-memory within the application server --- in which case it cannot be shared with other application servers --- or in a back-end database, shared by all application servers.

The latter option, sometimes referred-to as *persistent sessions* or *session clustering*, is critical to the use of HTTP sessions with any load-distribution configuration. With this option, whenever an application server receives a request associated with a session ID which it currently does not have in memory, it can obtain the required session state by accessing the back-end database, and is thus able to service the request. Conversely, when this

option is not enabled, if any load-distribution mechanism happens to route an HTTP request to an application server other than the one where the session was originally created, that server would be unable to access the session, and would thus not produce correct results in response to that request.

Of course, the session manager implements sophisticated caching optimizations to minimize the actual overhead of accessing the back-end database, in particular in the case where multiple consecutive requests get routed to the same application server.

Finally, the storing of session states in a persistent database also provides a degree of fault-tolerance to the system. If an application server crashes or stops, any session state that it may have been working on would normally still be available in the back-end database, so that other application servers can take over and continue processing subsequent client requests associated with that session.

Note

This mechanism does not provide a 100% guarantee that a session state will be preserved in case of a server crash. If a server happens to crash while it is literally in the middle of modifying the state of a session, some updates may be lost and subsequent processing using that session may be affected. However, such a situation represents only a very small window of vulnerability, and a very small percentage of all occurrences that typically happen throughout the life of a system in production.

1.3.2 EJB sessions or transactions

In the case of an EJB client interacting with one or more EJB's, the management of state information associated with a series of client requests is governed by the EJB specification, implemented by the WebSphere EJS container, and depends on the types of EJB's that are the targets of these requests.

1.3.2.1 Stateless session bean

By definition, when interacting with a **stateless session bean**, there is no client-visible state associated with the bean. Every client request directed to a stateless session bean is independent of any previous request that was directed to the same bean. The container will maintain a pool of instances of stateless session beans of each type, and will provide an arbitrary instance of the appropriate bean type whenever each client request is received. It does

not matter if the same actual bean instance is used for consecutive requests, or even if two consecutive requests are serviced by bean instances in the same application server.

1.3.2.2 Stateful Session Bean

In contrast, a **stateful session bean** is used precisely to capture state information that must be shared across multiple consecutive client requests that are part of one logical sequence of operations. The client must take special care to ensure that it is always accessing the same instance of the stateful session bean, by obtaining and keeping an EJB object reference to that bean. At the present time, WebSphere supports the distribution of stateful session bean homes among multiple application servers but not the distribution of a specifc instance. Each instance of a particular stateful session bean exists in only one application server, and can only be accessed by directing requests to that particular application server (of course, different instances of the same type of stateful session bean, used by different clients, may be spread among multiple servers). We will see in 2.3, “WLM” on page 23 that the various load-distribution techniques available in WebSphere, must make special provisions to support this characteristic of stateful session beans.

1.3.2.3 Entity Bean

Finally, we must consider the case of an **entity bean**. Most external clients access WebSphere services through session beans, but it is possible for an external client to access an entity bean directly. Furthermore, a session bean inside WebSphere is itself often acting as a client to one or more entity beans also inside WebSphere; if load-distribution features are used between that session bean and its target entity bean, then the same questions arise as with plain external clients.

Strictly speaking, the information contained in an entity bean is not usually associated with a “session” or with the handling of one client request or series of client requests. But it is common for one client to make a succession of requests targeted at the same entity bean instance. Furthermore, and unlike all the previous cases, it is possible for multiple independent clients to access the same entity bean instance more-or-less concurrently. Therefore, it is important that the state contained in that entity bean be kept consistent across the multiple client requests.

For entity beans, the notion of a session is more-or-less replaced by the notion of transaction. For the duration of one client transaction to which it participates, the entity bean is instantiated in one container (normally the container where the first operation within that transaction was initiated). All

subsequent accesses to that same bean, within that same transaction, must be performed against that same instance in that same container.

In-between transactions, the handling of the entity bean is specified by the EJB specification, in the form of a number of caching options:

- With **option A caching**, WebSphere assumes that the bean will only ever be used within a single container, and it is thus the responsibility of all clients of that bean to always direct their requests to the one bean instance within that specific container. Further with option A caching, the bean has exclusive access to the underlying database, this means that one cannot create clones when using option A caching in WebSphere.
- With **option C caching**, on the other hand, the bean is always reloaded from the database at the beginning of each transaction. Therefore, it is in theory acceptable for a client to attempt to access the bean and start a new transaction on any container that has been configured to host that bean. This is effectively similar to the session clustering facility described for HTTP sessions: the shared state is maintained in a shared database, and can be accessed from any server when required.

1.3.3 Server affinity

The discussion above implies that any load-distribution facility, when it picks a server to which to direct a request, is not entirely free to pick any available server:

- In the case of stateful session beans or entity beans within the context of a transaction, there is only one valid server. Websphere WLM will always direct a client's access to a stateful session bean to the single server instance containing the bean (no possibility of choosing the wrong server here). If the request is directed to the wrong server, it will either fail, or that server itself will be forced to forward the request to the correct server, at great performance cost.
- In the case of clustered HTTP sessions or entity beans in-between transactions, the underlying shared database ensures that any server can correctly process each request. However, accesses to that underlying database may be expensive, and it may be possible to improve performance by caching the database data at the server level. In such a case, if multiple consecutive requests are directed to the same server, they may find the required data still in the cache, and thereby reduce the overhead of access to the underlying database.

The characteristics of each load-distribution facility, that take into account these constraints, are generally referred-to as server affinity: in effect, the

load-distribution facility recognizes that multiple servers may be acceptable targets for a given request, but they also recognize that each request may have a particular affinity for being directed to a particular server, on which it will be handled better or faster.

We will encounter this notion of server affinity throughout the discussion of the various load-distribution facilities in 2.2, “Web Server to application server forwarding - OSE” on page 19 and 2.3, “WLM” on page 23. In particular, we will encounter the notion of session affinity, where the load-distribution facility recognizes the existence of a session and attempt to direct all requests within that session to the same server, and of transaction affinity, in which the load-distribution facility recognizes the existence of a transaction, and behaves similarly.

Finally, we will also see that a particular server affinity mechanism can be weak or strong. In a weak affinity mechanism, the system attempts to enforce the desired affinity for the majority of requests most of the time, but may not always be able to provide a total guarantee that this affinity will be respected. In a strong affinity mechanism, the system guarantees that affinity will always be strictly respected, and generates an error when it cannot.

Chapter 2. Techniques for WebSphere WLM and clustering

WebSphere provides a number of tools and techniques that come into play when implementing configurations that provide scalability, load-balancing and failover. We will summarize each of them in turn here, then provide in-depth descriptions in subsequent chapters, and finally show how these tools and techniques are combined to create real-life system configurations.

2.1 Cloning

Cloning is a mechanism provided by the WebSphere Administration system, that allows for the creation of multiple copies of an object such as an application server. Cloning is the process of taking a server that you've set up, and creating a model based upon that setup. Once you have a model made, you can then create clones of that server. With extra clones running you can improve the performance of your server. There is a point of diminishing returns, a point where the more clones that you add will actually slow you down with the extra maintenance and traffic generated by the clones and the management of them. Since cloning is part of the core of workload management, this will be demonstrated in detail throughout this book.

In brief, the system administrator creates a *model* for an application server, and from this model may create any number of copies or *clones*. The model is a logical representation of the application server, that exists only as information managed by the WebSphere Administration system. It has the same structure and attributes as a real application server: it may contain servlet engines, EJB containers, servlets, EJB's, etc. and allows the administrator to view and modify any properties of these logical objects. But it is not associated with any node, and does not correspond to any real server process running on any node. The clones created from this model, on the other hand, represent real application server processes running on real nodes. They are identical in every way to the model from which they were created except for some clone-specific attributes which must be set on a per-clone basis. Furthermore, if the system administrator makes changes to the model, these changes will be automatically reflected to all the clones. Several clones from the same model may be instantiated on multiple nodes, and it is also possible to instantiate multiple clones on the same node.

In a typical scenario, a system administrator might create a model of an application server, populate it with all the objects necessary for the implementation of his target application, fine-tune the properties of these

objects, and finally, when he is ready to deploy the application, create and start a number of clones that will begin to execute the application.

Figure 3 shows an example of a possible configuration that includes clones. Model 1 has two clones on node A, and three clones on node B. Model 2, which is completely independent of model 1, has two clones on node B only. Finally, node A also contains a free-standing application server, that is not a clone of any model.

Note

In the example and the discussion above, we only considered the creation of models and clones for entire application servers, which in turn contain servlet engines, EJB containers, etc. In truth, the WebSphere Administration system supports the creation of models and clones for objects at any level of the containment hierarchy. For example, it is possible to model and clone an individual EJB, without cloning either the application server or the EJB container in which it has been deployed. Although this capability is very important internally for the implementation of the overall model/clone facility in WebSphere, for the purposes of scalability, we will only consider the cloning of entire application servers. Moreover in terms of a best (or recommended) practice) you should only clone entire application servers which facilitates the administration of models/clones.

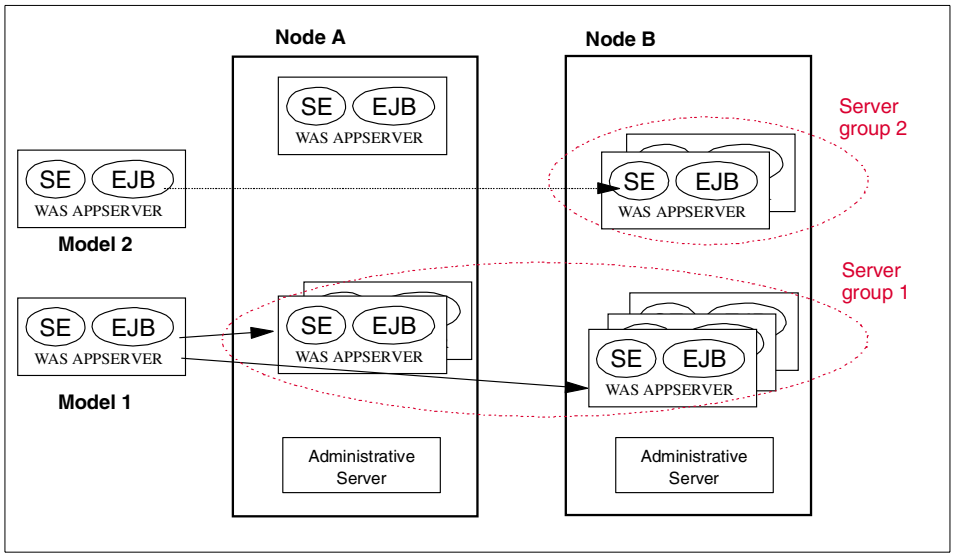

Figure 3. Models and clones

In practice, cloning provides two important benefits:

- It is a way to simplify system administration: clones can be used to quickly create and maintain identical copies of a server configuration.
- It is a way to organize workload distribution for several mechanisms (such as OSE Remote, WLM, and Servlet Redirector) provided with WebSphere: the set of all the clones of one model of an application server constitutes a logical group called a *server group* or *cluster*. The various workload distribution mechanisms use this abstraction of a server group to define the set of application servers among which they are to distribute the client requests. Since by definition all the clones are identical, the workload distribution mechanisms can safely assume that any one of the clones is equally capable of servicing any request.

In subsequent sections, we will see how this notion of clones and server groups is used to control workload distribution with the OSE transport, with the WLM tool for EJB's, and with the Servlet Redirector.

2.1.1 Vertical and horizontal cloning

In practice, we will see scalability used in two distinct contexts within the WebSphere Application Server run-time:

Vertical cloning refers to the practice of defining multiple clones of an application server on the same physical machine. Experience has shown that a single application server, which is implemented by a single JVM process, cannot always fully utilize the CPU power of a large machine and drive the load up to 100%. This is particularly true on large multiprocessor machines, because of inherent concurrency limitations within a single JVM. Vertical cloning provides a straightforward mechanism to create multiple JVM processes, that together can fully utilize all the processing power available.

Horizontal cloning refers to the more traditional practice of defining clones of an application server on multiple physical machines, thereby allowing a single WebSphere application to span several machines while presenting a single system image. Horizontal cloning can provide both increased throughput and failover.

2.1.2 Secure cloned resources

WebSphere V3 can be applied with security to cloned resources. WebSphere V3 has different steps for protecting cloned enterprise beans and cloned servlets.

2.1.2.1 Protecting cloned enterprise beans

Enterprise beans and their clones have separate identities. Therefore, you must explicitly protect each and every bean by configuring resource security for the bean and including it in a secured enterprise application.

For example, the “BeanThere” and “BeanThereClone” are considered to be two different enterprise beans, although they might be treated as clones in a WLM environment. To protect the beans, the administrator must configure resource security for each bean. Each bean must be explicitly included in a secured enterprise application, such as “BeanThereApplication.”

2.1.2.2 Protecting cloned servlets

In contrast to its approach to enterprise bean clones, WebSphere security does not treat a servlet and its clones as separate resources -- if the original servlet is protected, its clones are too, with no additional steps required by the administrator.

To secure a servlet, add its “Web resource” configuration (URI) to a secured enterprise application.

2.2 Web Server to application server forwarding - OSE

In any WebSphere configuration supporting Web clients, the Web server is responsible for receiving the requests from all the clients, filtering those that need to be serviced by WebSphere, and forwarding these to the Servlet Engine in an application server for processing. This actual forwarding of requests from the Web server to the application server is accomplished through a transport mechanism called *OSE*. OSE is a proprietary protocol internal to WebSphere, that can use a variety of IPC mechanisms provided by the underlying operating system to effect the actual transport of data: pipes, UNIX-domain sockets, and TCP/IP sockets.

The behavior of OSE is controlled by a trio of properties files that are generated by the WebSphere administration system, and read by the plug-in loaded inside the Web server. These files effectively list all the URLs that are to be serviced by WebSphere and, for each URL, specify one or more OSE queues associated with one or more application servers that are designated to serve that particular URL.

The OSE configuration lends itself to two types of load distribution in WebSphere:

- The configuration can easily be setup so that different URLs are always forwarded to different application servers. Although this does not correspond to a true load-balancing or failover solution in the strict sense of the term, it is clear that this approach can be used to arrange for the distribution of the overall load to be serviced by a WebSphere system, between multiple servers.

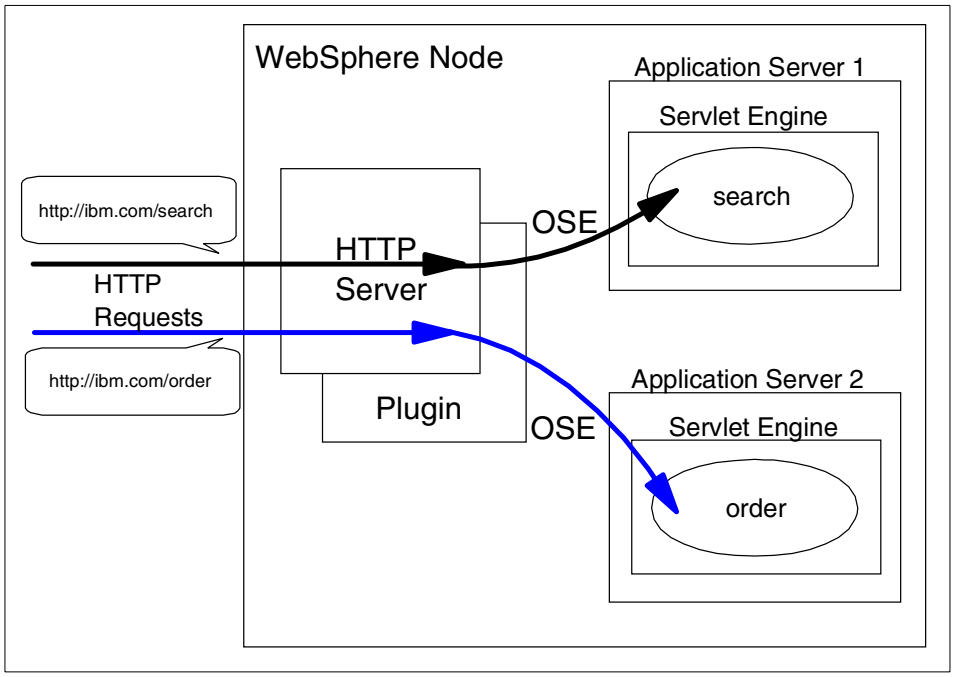

Figure 4. Different URLs are forwarded to different application servers

- OSE is fully integrated with the cloning feature of WebSphere. In the simplest configuration, a single Web server uses OSE to forward Web requests to a single application server, and there is obviously no question of load distribution. But if there are multiple clones defined to respond to a single particular URL, the OSE transport will automatically distribute all requests among all available clones.

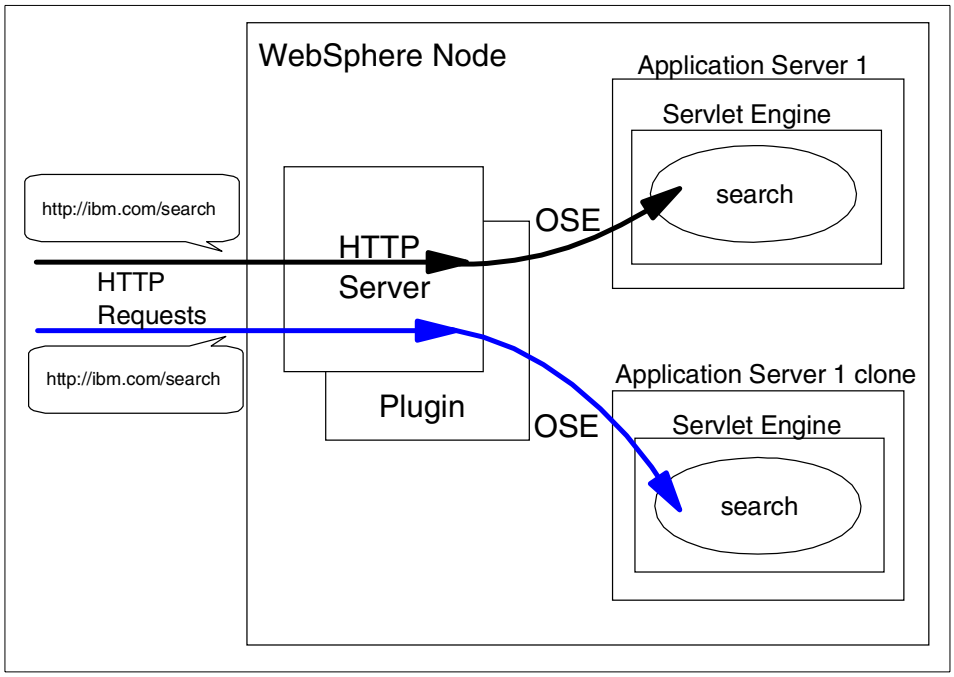

Figure 5. The OSE transport distributes all requests among all available clones

In the latter case when using multiple clones, the way in the which the Web server selects the application server clone to receive each request is a combination of round-robin with session affinity:

- If session persistence is not enabled (the default), the Web requests are distributed among all available clones in a strict round-robin fashion that is each clone gets the next request in turn, except as noted below with session affinity and with a clone restart.
- If session clustering and session affinity are enabled:
  - For all Web requests that are not associated with a session, or for the first request of each session (for example, before a cookie and/or session ID have been established), the same round-robin distribution policy is used.
  - For all Web requests that are associated with a session, WebSphere attempts to consistently route all requests within the same session to the same clone. Different sessions will be assigned semi-randomly to different clones.

Note however that there is no guarantee that the same clone will absolutely always be used for all requests within one session. The affinity is established and maintained via a hash function on the available servers and as such can only provide a “best-effort” at maintaining session affinity, subject to occasional switches imposed by the implementation (for example when the number of available clones changes during the life of the session). The system relies on the session clustering facilities to ensure that session state will not be lost in the event of such a switch. While OSE session affinity in conjunction with the session caching provided WebSphere Session Manager can greatly improve performance, applications that require session information to be available across multiple client invocations MUST persist session to a database.

The OSE transport will also automatically handle failover and changes in the set of available clones. If a clone is stopped or crashes, all subsequent Web requests will be distributed among the remaining available clones, and the unavailable clone will be skipped. Conversely, if a clone is added or restarted, the system will automatically start to distribute a fair share of the load to it. In fact, upon start/restart of a clone the next several requests may be dispatched to that clone, as opposed to a strict round-robin fashion. OSE will then again start to direct requests in a round-robin manner to all running clones.

Note

WebSphere 3.0.2.1 does not support session affinity in the OSE transport. Look for this feature to be introduced in WebSphere 3.0.2.2 and 3.5.

2.2.1 Local OSE

The initial specification for the OSE transport called for it to be used only locally, for example, between a Web server and one or more application server clones that are executing on the same physical machine as the Web server.

2.2.2 Remote OSE

However, starting with WebSphere Version 3.0.2.1, the OSE transport specification has been extended to also support remote operation, for example, communication between a Web server on one machine, and multiple application server clones that may be executing on one or more other machines that are distinct from the machine hosting the Web server. In this

case, the underlying transport mechanism used for OSE communication is TCP/IP, but all other characteristics of the OSE mechanisms remain the same as in the local case, including routing different URLs to potentially different application servers, load distribution between all clones, session affinity, failover, etc.

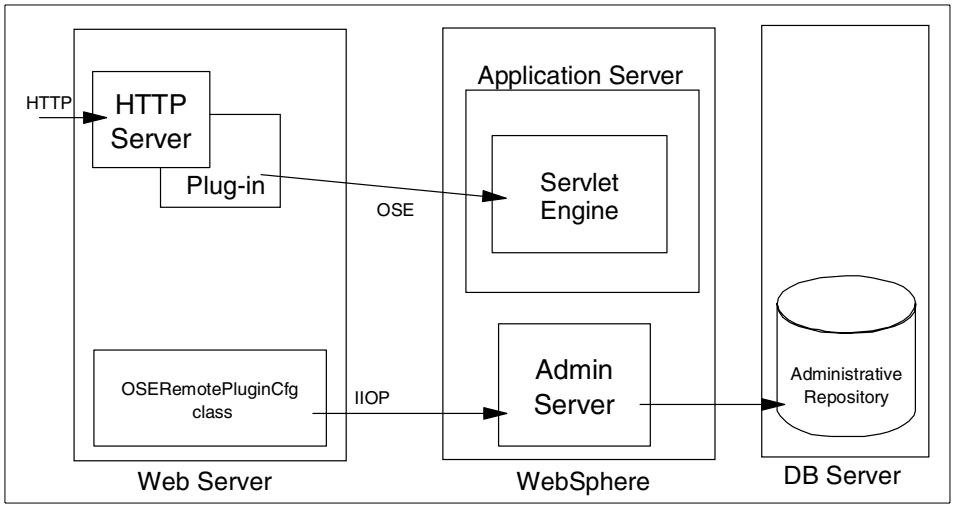

Figure 6. OSE Remote

This **OSE Remote** option provides additional flexibility for load distribution, and also offers some distinct advantages in secure configurations where the Web server may be separated from the rest of the WebSphere system by a firewall. See Appendix A, “Firewall considerations” on page 465 for more details.

2.3 WLM

WLM or Workload Management is the primary mechanism for load-distribution of requests directed at EJBs. In a simple, non-load-balanced EJB in WebSphere, each client of the bean holds a stub that contains a CORBA reference to the corresponding bean on the server. Whenever the client invokes an operation on that bean, the request is simply forwarded to the server object associated with that CORBA reference. Note that the client can be a stand-alone Java program using RMI/IIOP, a servlet operating within a WebSphere servlet engine, or another EJB.

When using the WLM facility, the simple client stub above is augmented by a smart stub, that contains a collection of CORBA references to multiple instances of the same bean in different servers. Whenever the client invokes an operation on the bean, the smart stub automatically and transparently forwards the request to any one of the available server objects, thereby achieving both load-balancing and failover when appropriate. The smart stubs transparently communicate with the WebSphere EJS runtime to keep track of which servers and EJB instances are available at any given time.

The WLM facility is enabled through one of three alternatives:

1. EJBs developed and deployed from VisualAge for Java are automatically WLM'd.
2. EJBs developed and deployed outside of Visual Age for Java, must first be deployed inside the WebSphere run-time, then the WLMjar processor (a .bat file on NT, a .sh file on UNIX) must be executed against the deployed EJB with WebSphere V3.02x.
3. With WebSphere V3.5, when you deploy an Bean inside the WebSphere run-time, you can specify that the bean is to WLM enabled.

All three options generate the smart stubs and other classes required for WLM. These smart stubs behave as outlined above and look exactly the same as simple stubs as far as the client is concerned; there is no need for any change in the client code to take advantage of the WLM facility.

As with other load-distribution facilities in WebSphere, the set of EJB instances that are available for load-distribution through WLM, is defined by the set of available clones of any given object. Currently, the WLM facility provides load-distribution among:

- All clones of the home of a session (stateful or stateless) or entity bean (thereby allowing bean instances to be created in different servers)
- All the clones of an instance of a given stateless session bean or the instance of a given entity bean

In fact, the only type of EJB reference not subject to load-distribution through WLM are the instances of a given stateful session bean. This is because, as discussed in 1.3.2, “EJB sessions or transactions” on page 10, stateful session beans cannot be replicated or are not shared between multiple servers.

2.3.1 WLM runtime

The preceding discussion has provided an outline of the WebSphere WLM runtime. The following will provide a more detailed explanation of how the WLM runtime actually works.

The first two steps are depicted in Figure 7.

- During the initial call from an EJB client the underlying WLM client runtime (the “smart stubs” discussed previously) contacts the WebSphere Administrative Server to obtain information about the server group. Note that since the clones that comprise a server group can be distributed across multiple nodes, some of this information may be obtained from the Administrative Repository for all nodes in the cluster.
- In turn the Administrative Server returns the requested information to the EJB client.

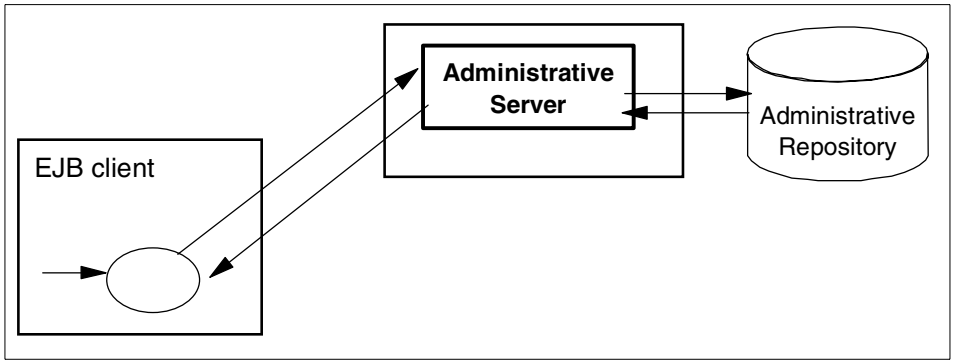

Figure 7. WLM client initial access: steps 1 and 2

The information returned regarding the server group and the currently active clones is in turn used to populate an array of proxy stubs, each one representing an active clone. In the simple case depicted in Figure 8 on page 26 below, there are two active clones, so two corresponding proxy stubs (labeled Proxy1 and Proxy2) are created.

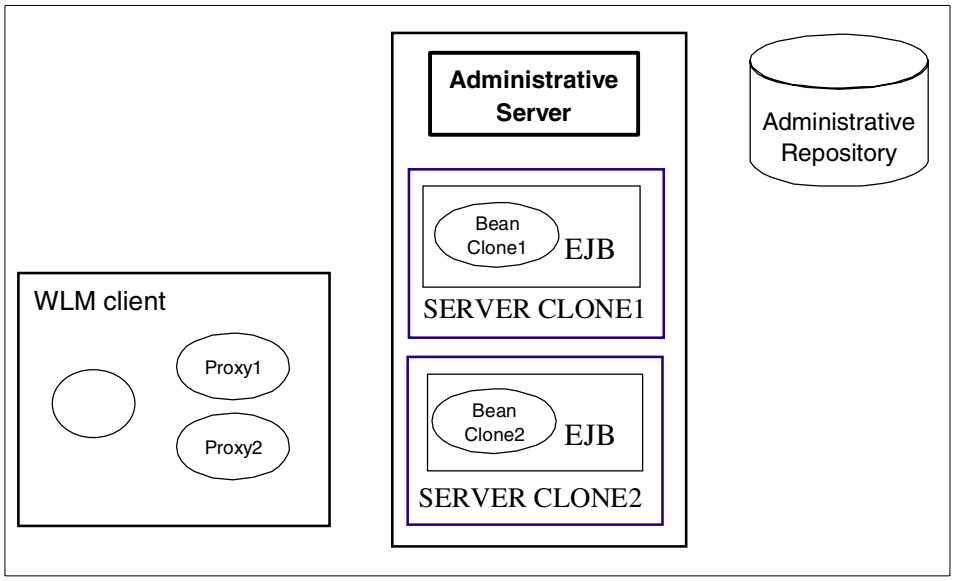

Figure 8. WLM client initial access: step 3

Lastly as depicted in Figure 9, the WLM runtime proxy manager directs the request to a clone based on the WLM policy in effect at the time.

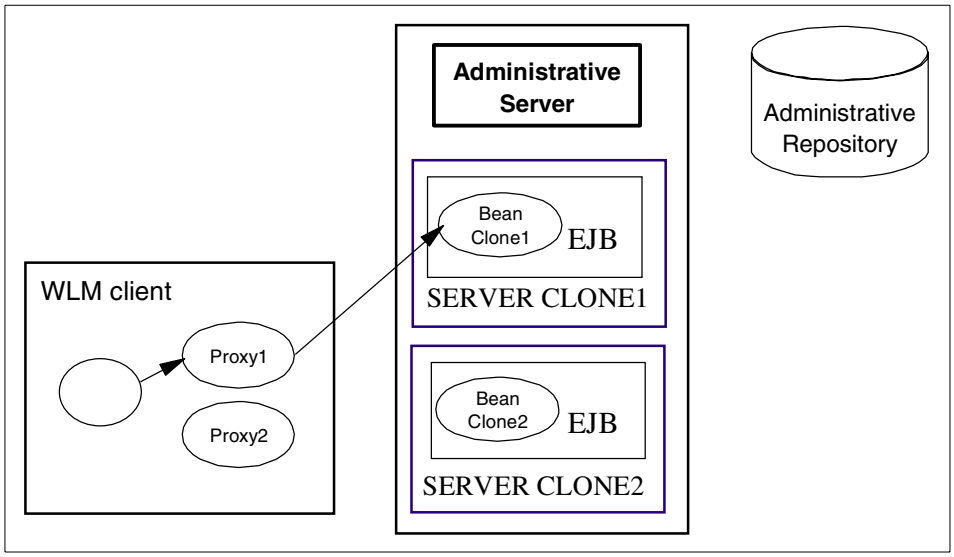

Figure 9. WLM client initial access: step 4

Unless a refresh of the WLM runtime client cache is required, subsequent requests utilize the existing proxies for each active server (we will discuss the refresh mechanisms in detail below). Assuming that there has not been a change in the state of the model server group, the second request is either dispatched to the same clone, if the call is in context of a transaction, or to another clone in the server group as depicted below in Figure 10.

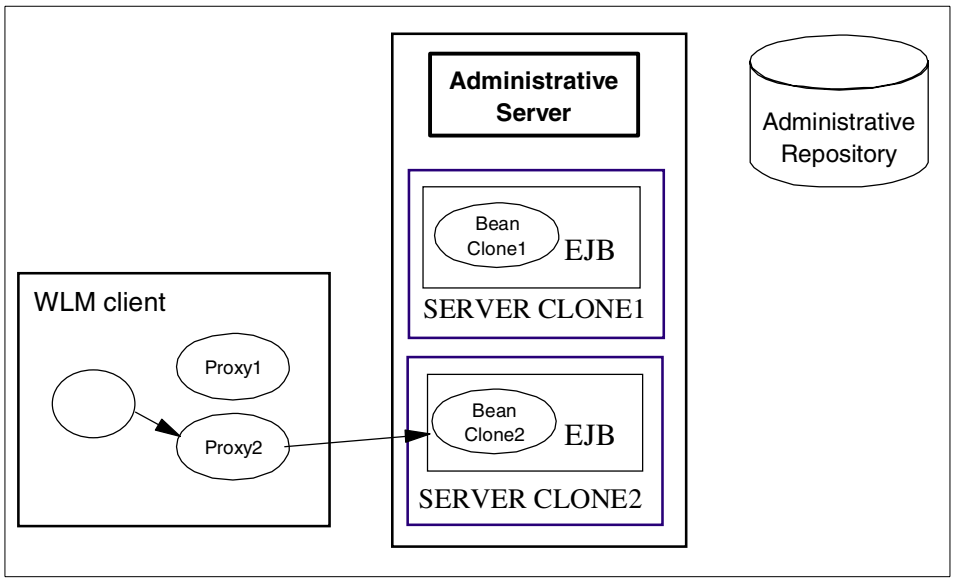

Figure 10. WLM client, subsequent requests

Of course the above would be fine if changes never occurred in the state of the EJB servers. As we all know it is necessary to stop and start servers for maintenance. The runtime also has to be capable of reacting to hardware and software failures.

Let's assume that instead of succeeding that the second client request failed. In this case the WLM client runtime will attempt to determine if more recent information exists for the server model group. If more recent information is not available and it is safe to retry (as discussed below in 2.3.3, "WLM runtime exception handling" on page 36), then a retry is attempted to the same clone, if the clone is still unavailable then the clone is marked as unusable as depicted below in Figure 11 on page 28 and it will remain in that state for a prescribed time interval (UnusableInterval) after which the WLM client runtime will attempt to use the clone again. In the interim the WLM client runtime will direct requests to surviving servers in the server model group as depicted in Figure 11 on page 28.

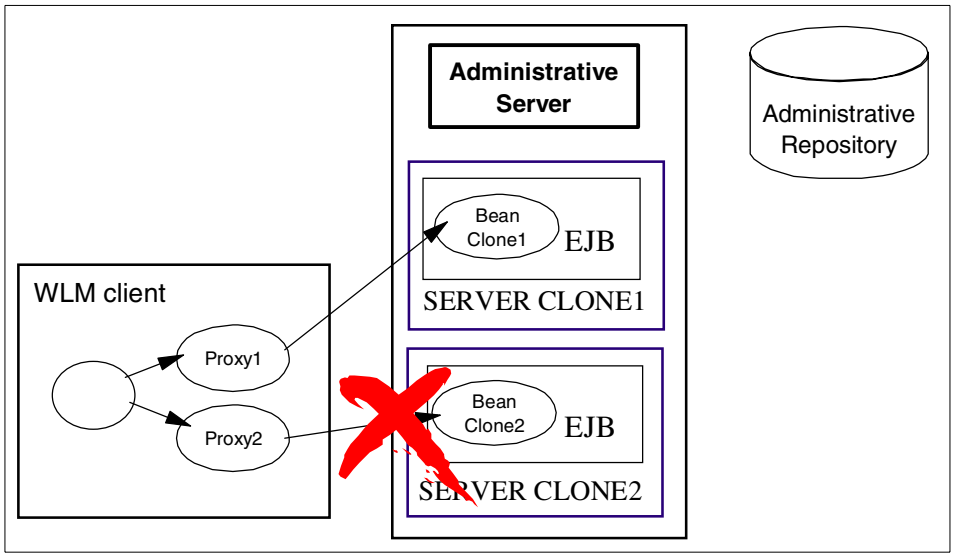

Figure 11. WLM client request to a failed server

If requests to subsequent servers in the model server group are unsuccessful, these exceptions will be handled as described above for all the servers in model server group until a request succeeds or all servers are marked as unusable. In this case the WLM client runtime will contact the Administrative Server for new model server group information as depicted in Figure 12.

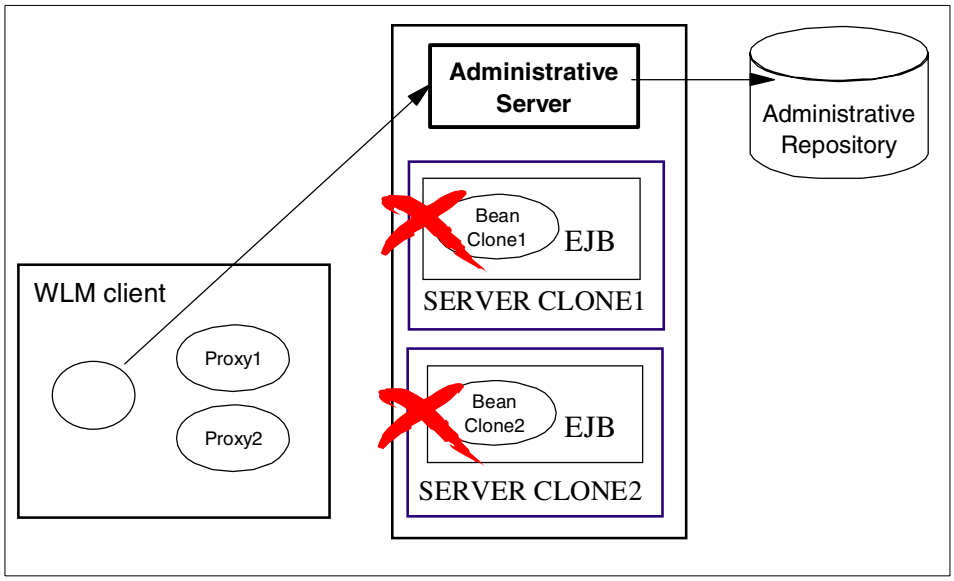

Figure 12. WLM client request failed to all servers

2.3.1.1 WLM runtime refresh mechanisms

As noted above, it is necessary to propagate changes in WLM runtime state to other nodes within a WebSphere domain as well as to clients that may be making requests of servers running on those nodes. There are two components to this; one for the servers and one for the clients.

First let's consider the WLM server refresh that occurs when a change is made to a model server group. As an example consider the case where a new clone is added to a model. In our example a third clone (Server Clone 3) has been added to the model, and a start for the third clone has been issued from the Administrative Console as depicted in Figure 13 on page 30 (note that alternatively the server start could be issued from the command line with XMLConfig or wscp). After the start has been issued from the Administrative Console, the change in runtime state is sent to the Administrative Repository from the Administrative Server.

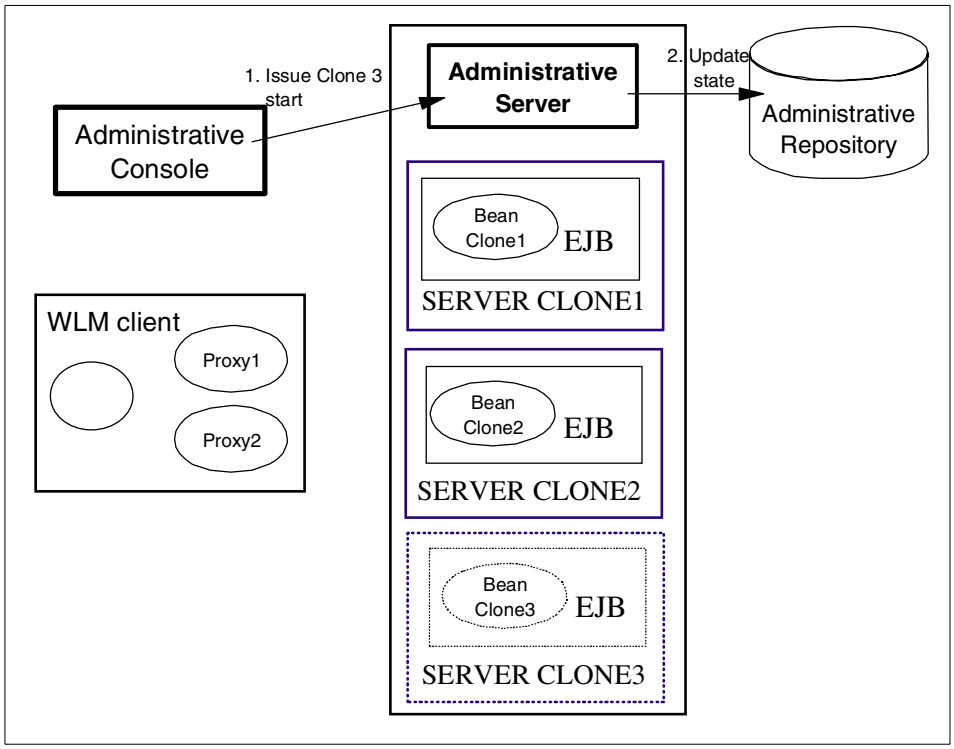

Figure 13. WLM server refresh, repository update

Once the change in runtime state has been issued to the Administrative Repository, the actual start command is issued to the server clone (Server Clone 3 in our example) as depicted in Figure 14.

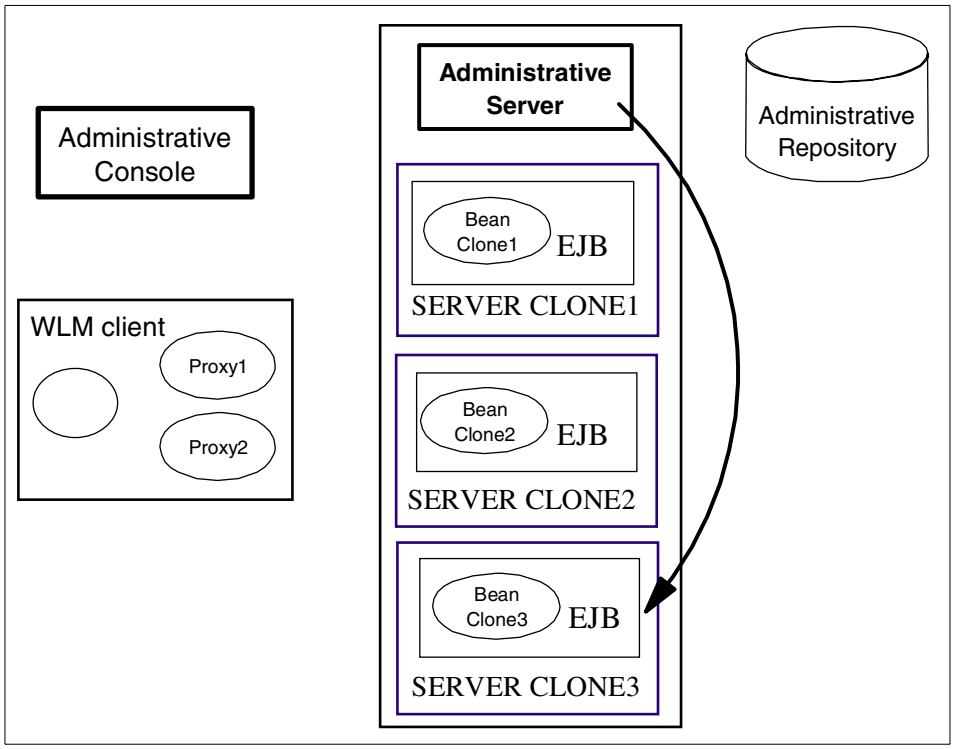

Figure 14. WLM server refresh, server start

The final steps in the process are to propagate the change in the state of the model server group to other clones running on the same machine and then to the Administrative Servers for other nodes in the WebSphere domain. This is depicted in Figure 15 on page 32. The remote Administrative Servers will in turn propagate the change in state to any clones for the model server group running on remote nodes.

Note

The propagation of runtime state changes to clones by the Administrative Server is not instantaneous; it can take up to one minute. Changes in state from the server down are in turn propagated as prescribed by the refresh interval property (-Dcom.ibm.ejs.wlm.RefreshInterval=xxx) which by default is 600 seconds.

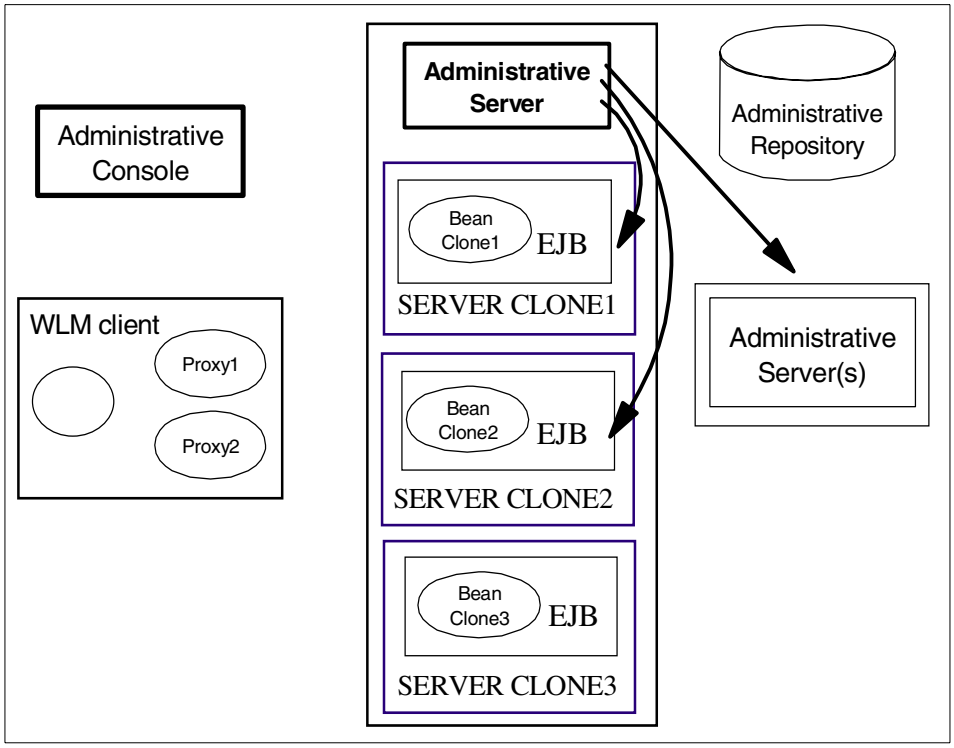

Figure 15. WLM server refresh, servers and nodes

As far as changes propagating to the WLM client, this occurs as part of normal request processing. As depicted in Figure 16 on page 33 a WLM client makes a request, and the request is directed to a known clone based on affinity or WLM selection policy (as appropriate). In this example the request is directed to Server Clone 2.

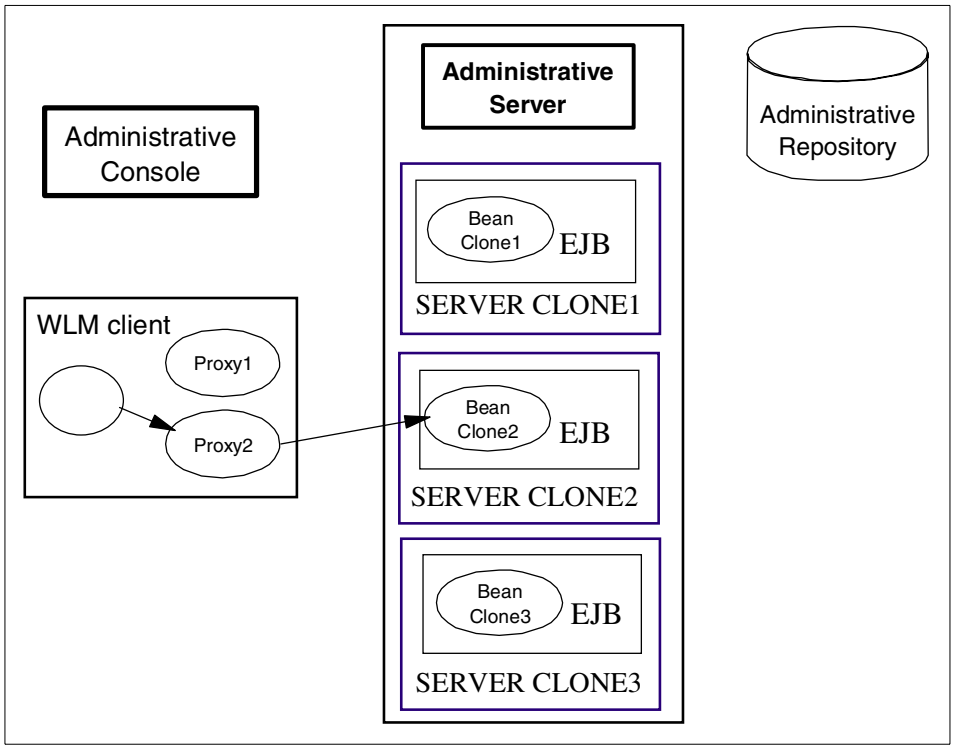

Figure 16. WLM client refresh, client request

Upon receiving the request, the WLM server runtime makes use of an epoch number that is changed each time there is a change to the state of the model server group. A comparison is made between the epoch number received with the client request with that on the server. In this example, the server detects that the client cache is stale (old) since the epoch numbers do not match. As part of the request reply packet to the client, the new WLM state is sent as depicted in Figure 17.

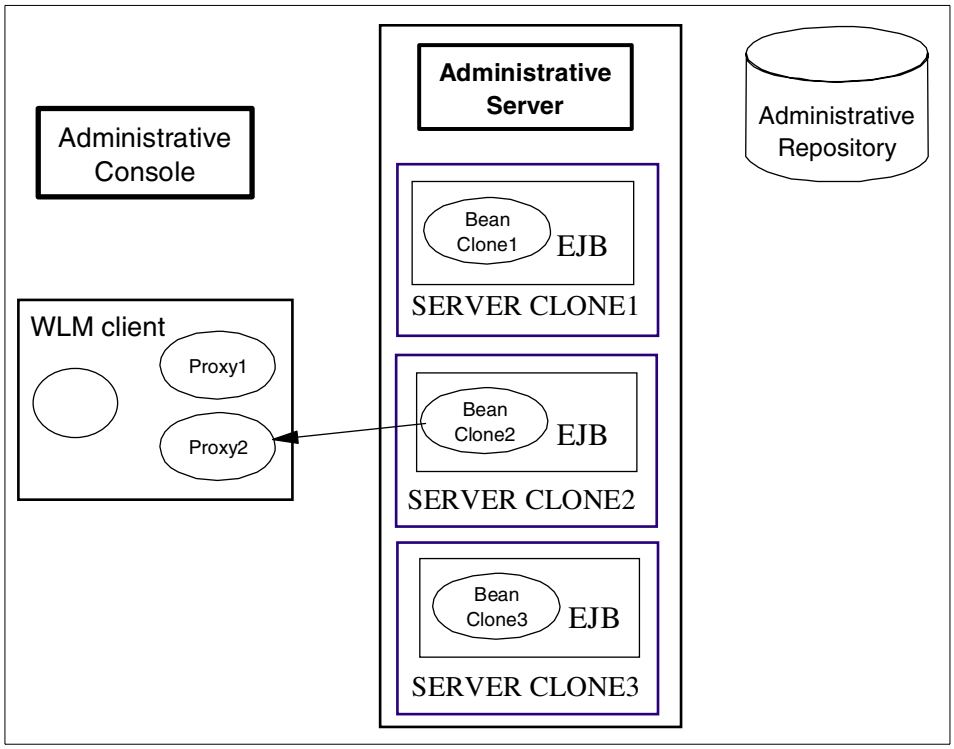

Figure 17. WLM client refresh, client reply

The last step in the process is depicted in Figure 18 on page 35. Once the client receives the request reply, the WLM runtime client is updated to reflect the changes in model server group. In this case a third proxy is added, corresponding to Server Clone 3.

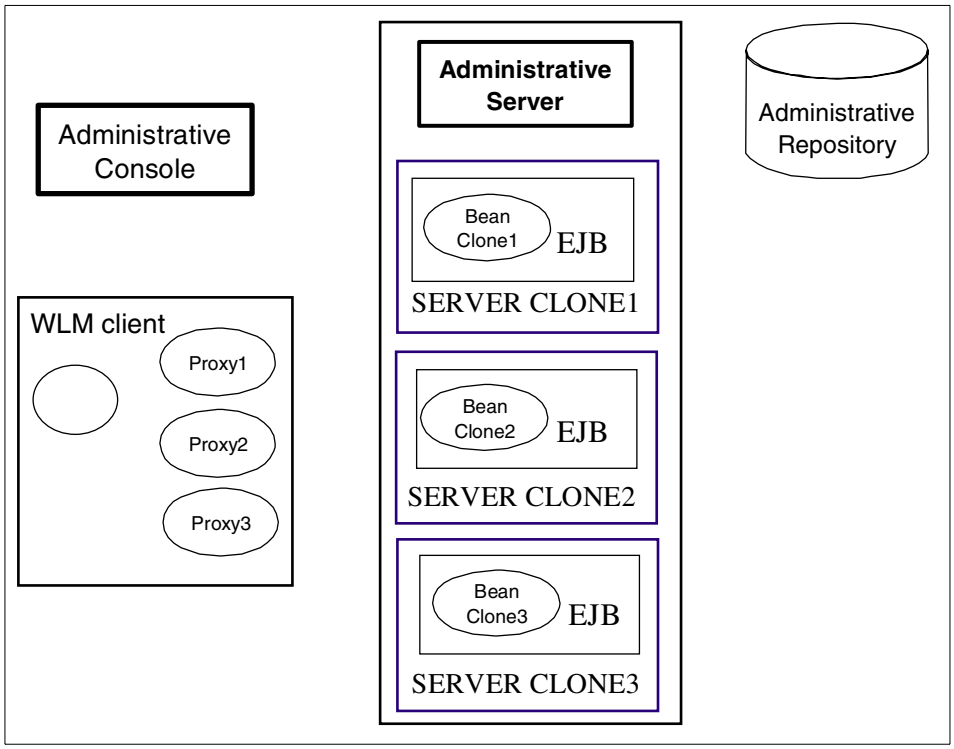

Figure 18. WLM client refresh, updated client runtime

2.3.2 WLM load balancing options

The actual selection of a particular object instance among all available clones of that object, is controlled by a load-balancing policy (Workload management selection policy) attribute associated with the application server group that contains this object, according to the following rules:

1. If the client that is issuing a request is located within the same JVM (typically an application server) that also contains an available instance of the target object, the WLM facility is bypassed, and the request is always dispatched to that local object instance.
2. If the request is associated with a transaction, and a previous request within the same transaction has already been dispatched to a given instance of a given object, then the new request is also dispatched to that same instance. This implements the transaction affinity property described earlier.

3. If the server group's load-balancing policy (Workload management selection policy) is set to random-prefer-local, and there exists one or more available instances of the target object inside an application server that is executing on the same machine as the client (but not in the same application server, else rule 1 would apply), the request is dispatched to one of those application servers on the same machine, selected at random. If there are no available instances on the same machine, the request is dispatched to any available instance at random, regardless of which machine the application server is running on.
4. If the server group's load-balancing policy (Workload management selection policy) is set to round-robin-prefer-local, and there exists one or more available instances of the target object inside an application server that is executing on the same machine as the client (but not in the same application server, else rule 1 would apply), the request is dispatched to one of those application servers on the same machine, selected in round-robin fashion. If there are no available instances on the same machine, the request is dispatched to any available instance in round-robin fashion, regardless of which machine the application server is running on.
5. If the server group's load-balancing policy (Workload management selection policy) is set to random, the request is dispatched to any available instance at random, regardless of which machine the application server is running on.
6. If the server group's load-balancing policy (Workload management selection policy) is set to round-robin, the request is dispatched to any available instance in round-robin fashion, regardless of which machine the application server is running on.

2.3.3 WLM runtime exception handling

In general, if an object instance is found to be unavailable due to a crash or for other reasons, the request will be retried with another available instance, thereby providing automatic failover. Automatic failover does **not** take place when a CORBA COMM_FAILURE exception or a CORBA NO_RESPONSE exception is thrown with "MAYBE" COMPLETION_STATUS, which maps to a java.rmi.RemoteException. This is because WLM cannot know if the operation partially completed or not. In these cases the application has to expect possible failures and initiate a retry, if appropriate. WLM will then try to find a surviving server and dispatch the request. WLM also does not retry in the case of an application exception, again because WLM cannot know if it is appropriate to do so. Automatic failover will take place when a

"COMPLETED_NO" COMPLETION_STATUS is received, since it is safe to do so. Of course no retry is required with COMPLETED_YES.

A simple example of the application code is depicted below in Figure 19 for instances of a “MAYBE” COMPLETION_STATUS minor code. In practice additional code would be required to determine the outcome of the attempted operation. As a matter of course this type of failure is extremely rare, usually corresponding to network outages (most typically the result of pulling a network cable during a test of WebSphere WLM).

```
success = false;
noRetries = 2; // or some small positive number.
for (i = 0; i < noRetries; i++) {
// We may get back exceptions from the WLM runtime.
// Catch these and retry.
   try {
   serverName = tester.invoke(); //tester represents the remote object
   success = true;
   } catch (RemoteException e) {
   Tr.warning(tc, "tester.invoke() threw Exception", e);
   //Or, do whatever cleanup you want
   }
   if (success) break;//no need to retry.
}
```

Figure 19. Example application retry code

Figure 20 shows an example of a WLM-enabled EJB client dispatching requests to two cloned application servers.

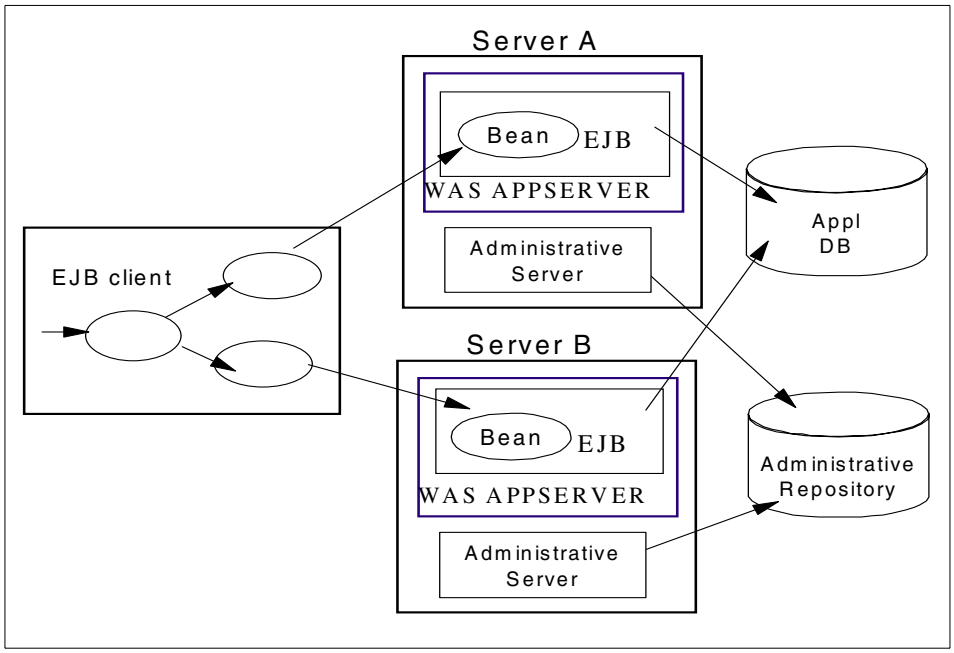

Figure 20. WLM-enabled client

It should be noted that the round-robin selection policy implemented by the smart stubs, is in effect independent for each client, and each corresponding instance of a smart stub. So, strictly speaking, one should not expect this mechanism to provide a strict round robin distribution of all incoming requests from all clients, as perceived from the point of view of the servers. However, the main goal, to provide a statistically fair distribution of the load across all servers, is achieved by this implementation. With a large number of independent clients, this approach further approximates a random distribution, without resorting to an explicit randomizing function.

It should also be noted that, for both the random and round-robin policies, each server instance is considered equal to all other instances, and there is no way to cause the system to favor some servers over others (for example servers on different machines, or subject to different background loads). Future versions of WebSphere may provide extensions such a weights associated with each server, or various schemes to use dynamic measures of the load on each server to influence the selection policy.

With the current available selection policies, one way to ensure that one machine receives a larger proportion of the requests than another machine, is

to use vertical cloning to define a greater number of application servers on the larger machine.

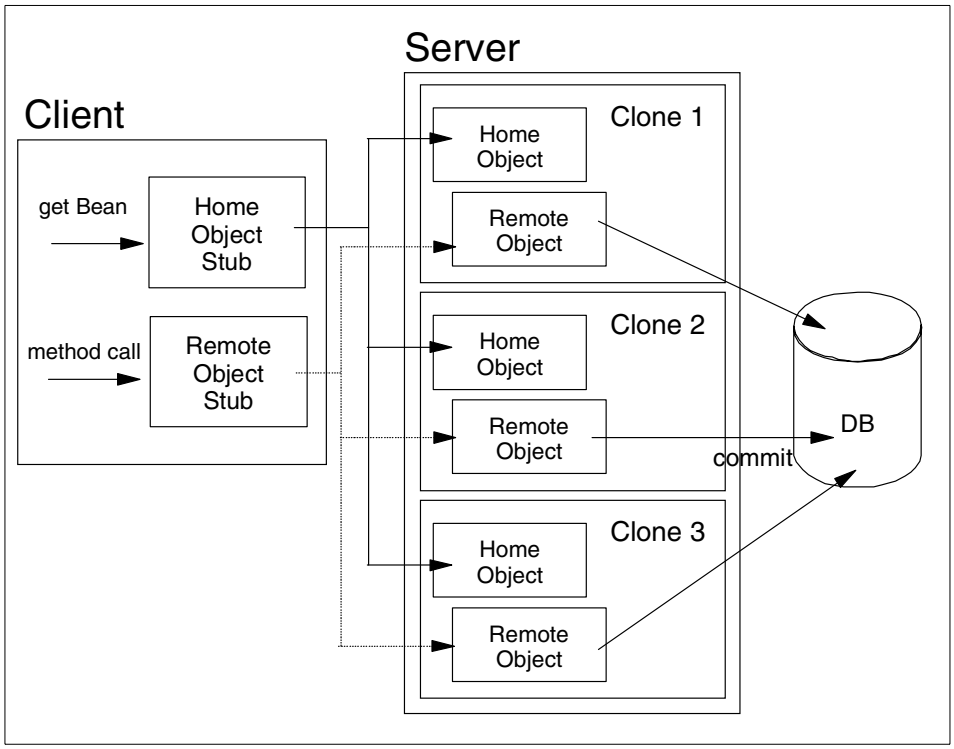

Figure 21. WLM for EJBs

WLM for Administrative Servers with V3.5 PTF2

Administrative Servers can participate in workload management. Currently, the primary benefit is that it provides failover capability for Administrative Servers, improving the availability of administrative and naming services (since the Administrative Server process also provides naming services).

Workload management must be enabled (or disabled) for all Administrative Servers in a domain.

When an Administrative Server participates in workload management, an exception is thrown if the server fails during an administrative task. Subsequent requests are redirected to the other servers in the domain, minimizing the disruption to administrative operations.

For example, a command issued through the WebSphere Administrative Console can fail if a server becomes unavailable while the command is being executed. However, if workload management is enabled, any subsequent attempts to execute the command are redirected to another Administrative Server. This allows the command to be successfully reissued, with only a slight delay for the initial redirection. (The original Administrative Server will pick up its share of administrative requests when it comes back on-line.)

To enabling workload management, start all Administrative Servers in the domain with workload management enabled. WebSphere Application Server provides two ways to enable workload management for Administration Servers:

- By setting the following property in the admin.config file:

  ```
  com.ibm.ejs.sm.AdminServer.wlm
  ```

  This enables workload management for all Administrative Servers that are started using this configuration file.

- By specifying the -wlm argument when starting an Administrative Server from the command line. For instance:

  ```
  java com.ibm.ejs.sm.server.AdminServer -wlm ...
  ```

  where . . . represents any other arguments that are specified when starting the server.

Enabling workload management through the admin.config file is recommended because it is easier to administer than enabling it through the command line.

2.4 Servlet Redirector

The Servlet Redirector, like the Remote OSE facility described in 2.2.1, “Local OSE” on page 22, is a mechanism that allows HTTP requests received by a Web server to be forwarded to one or several application servers.

Servlet Redirector can be thought of as a special case application server, that runs on the same machine as the Web server, and that receives HTTP requests through a local OSE channel, just like a regular application server and servlet engine would. But instead of processing the incoming HTTP requests in a servlet locally, Servlet Redirector forwards these requests to the “real” application servers.

The forwarding mechanism uses the EJB facility. Each servlet engine in each application server that accepts requests forwarded from Servlet Redirector, contains a special stateless session EJB, the RemoteSRP bean, that exports a simple method which is invoked for each incoming HTTP request, and passes that request on for execution by the servlet engine. The Servlet Redirector itself acts as an EJB client: whenever it receives an HTTP request from the Web server, it looks up the appropriate forwarding EJB in the target application servers, and invokes the forwarding method through RMI/IIOP, like any ordinary EJB client.

Since the communication between Servlet Redirector and the target application servers is through EJB client invocations, it can naturally be load-distributed through the WLM facility, thereby giving Servlet Redirector the ability to reach multiple cloned application servers to service each HTTP request, with the same load-balancing policies available to any WLM client.

Figure 22 shows an example of a Servlet Redirector forwarding requests to two cloned application servers.

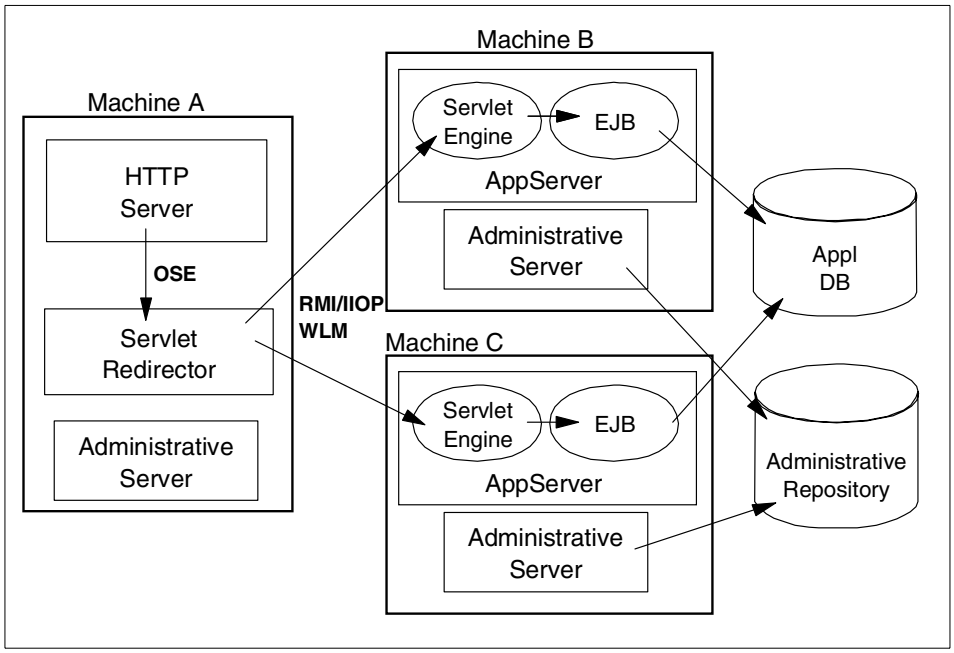

Figure 22. Servlet Redirector

2.4.1 Servlet Redirector configurations

Since Servlet Redirector is essentially a stripped-down application server, it normally runs under the control of a WebSphere Administrative Server, just like any other application server. However, it is often desirable to run Servlet Redirector more-or-less alone on the machine hosting the Web server, and to leave most of the WebSphere administrative machinery only on the machines hosting the actual application servers. This gives rise to a few variants for the configuration of Servlet Redirector:

- In the thick Redirector configuration, the node running Servlet Redirector is configured as a full-fledged WebSphere node, with an Administrative Server and all the attending processes and attributes. In this mode, Servlet Redirector is administered just like any other WebSphere process.
- In the thin Redirector configuration, Servlet Redirector is executing stand-alone on a node, without the normal support provided by an Administrative Server running on the same node. Special scripts are provided to start Servlet Redirector manually, and to generate the few configuration files that it needs to operate.

- Finally, in the Admin-agent Redirector configuration, Servlet Redirector executes on a node along with a stripped-down version of the Administrative Server, called an admin-agent. Like a full-fledged Administrative Server, an admin-agent can perform all the necessary functions to manage the Servlet Redirector. But unlike a full-fledged Administrative Server, an Admin-agent is slaved to a “real” Administrative Server and does not access the Administrative Repository directly. The "master" Administrative Server is responsible for all the administrative operations, and simply sends orders to the admin-agent to control the Servlet Redirector remotely.

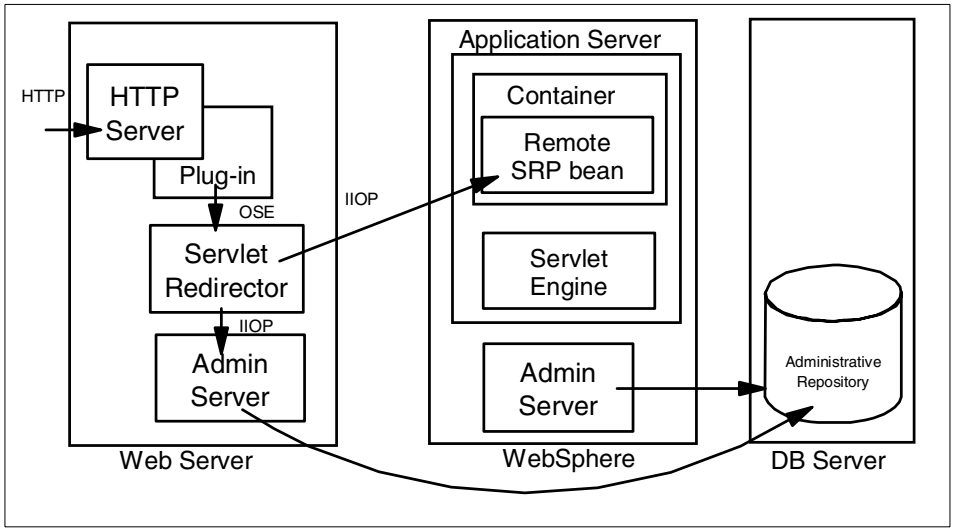

Figure 23. Thick Servlet Redirector

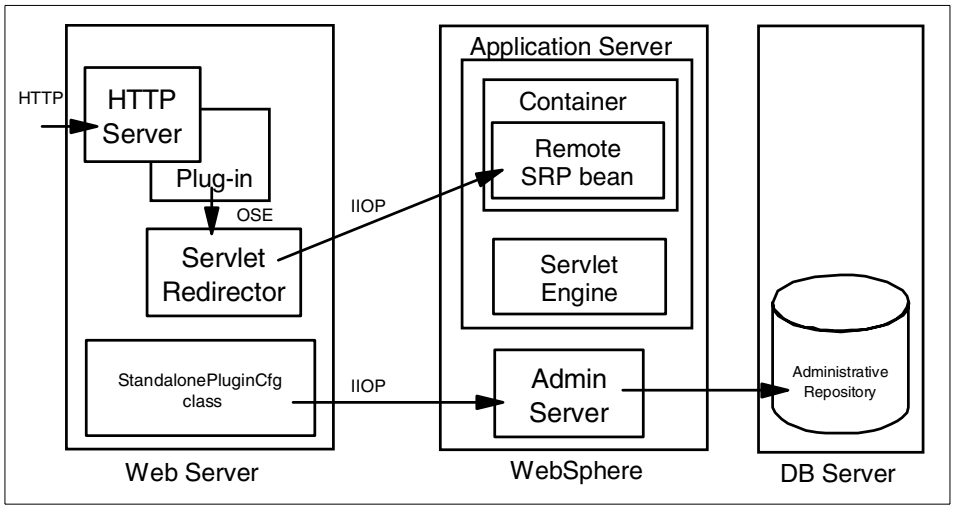

Figure 24. Thin Servlet Redirector

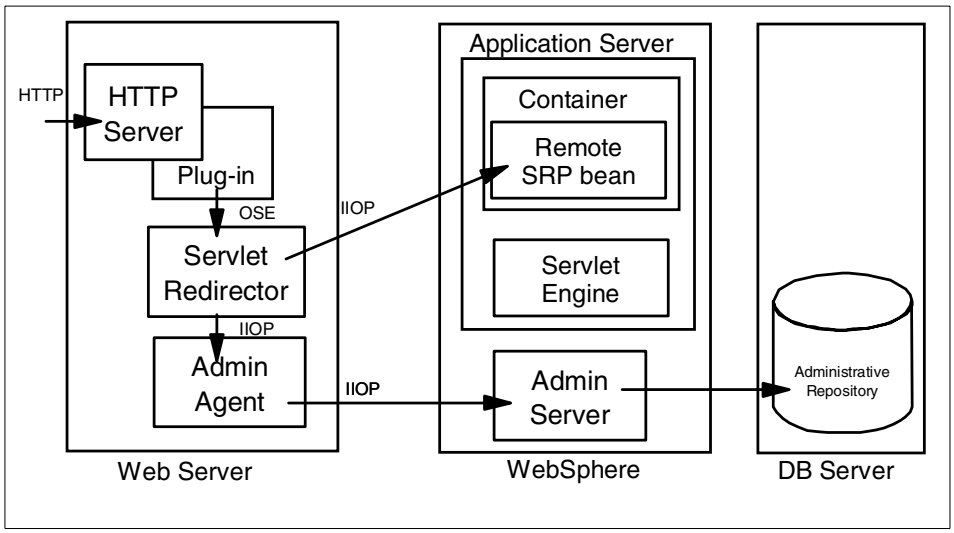

Figure 25. Thick Servlet Redirector with Administrative Server agent

See Chapter 3, “Introduction to topologies” on page 47 and Appendix A, “Firewall considerations” on page 465 for details of the pros and cons of various Redirector configurations.

2.5 Network Dispatcher

Network Dispatcher is a standard component of WebSphere Edge Server. It is a high-performance *IP Sprayer:* it accepts incoming TCP/UDP request packets such as HTTP or HTTPS from many Web clients, and transparently redirects them to any one of a number of Web servers that are part of a *Network Dispatcher Cluster.* Each client is fooled into thinking that it is communicating directly with a given Web server, when in fact Network Dispatcher is in the middle, intercepting all requests and multiplexing them among all the available Web servers.

Network Dispatcher provides scalability and load-balancing among all the Web servers in the cluster, by distributing the requests from clients so that each server receives only a fraction of all the incoming requests. It also provides failover, by keeping track of which servers are currently available or failed, and distributing requests only to the currently available servers.

Network Dispatcher is implemented as a special software program, that normally runs on a dedicated machine on the network. It also supports failover, in that two dedicated machines can cooperate to run the Network Dispatcher software. If one of the machines fails, the other automatically takes over, and the dispatching/multiplexing of client requests can continue unaffected.

By default, Network Dispatcher distributes the client requests among all available servers in a *round-robin* fashion: each server in the cluster gets the next request in turn. But it also comes with very extensive customization options that allow the system administrator to control how it decides the distribution of requests among the servers in the cluster. These include the use of fixed *weights* to favor some servers over others and give them a specific percentage of the total load; dynamic load measurements to adjust the distribution based on the current load on each server; custom *rules* that govern how multiplexing decisions are made based on the type of request, time of day, etc.; and custom *advisors* that execute a user-specified program to make multiplexing decisions. Finally, Network Dispatcher can also implement a particular form of *affinity*, in which requests from a particular client or group of clients are consistently routed to a particular server, so as to minimize the need for multiple servers to coordinate their actions if they are all processing requests on behalf of the same client.

Figure 26 shows a simple configuration using IBM Network Dispatcher.

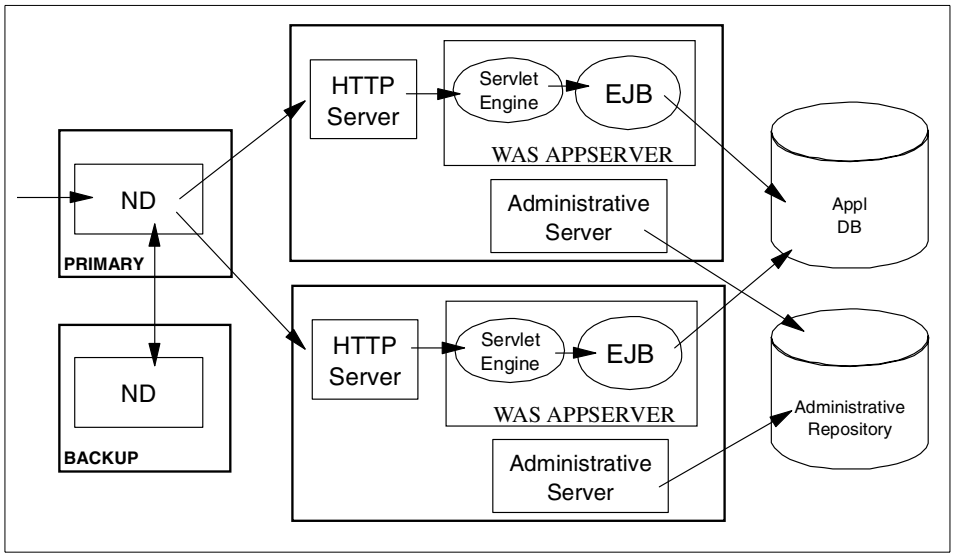

Figure 26. Using IBM Network Dispatcher with WebSphere

Chapter 3. Introduction to topologies

While a variety of factors come into play when considering the appropriate topology for a WebSphere deployment, the primary factors to plan for typically include:

- Security
- Performance
- Throughput
- Availability
- Maintainability
- Session state

This chapter will discuss the factors mentioned above and will describe various topologies that address each of these items. In addition options regarding the use of multiple clones, multiple WebSphere domains, vertical and horizontal scaling as well as multiple tiers (separating the servlet and EJB servers) will be covered as well.

3.1 Topology selection criteria

3.1.1 Security

Security concerns usually require physical separation of the HTTP (Web) server from the application server processes, typically across one or more firewalls.

EJB security reminder

Recall that as mentioned in 2.1.2, “Secure cloned resources” on page 18 enterprise beans and their clones have separate identities. Therefore, you must explicitly protect each and every bean by configuring resource security for the bean and including it in a secured enterprise application.

3.1.2 Performance

Performance involves minimizing the response time for a given transaction load. While a number of factors relating to the application design can affect this, adding additional resources in the following two manners, or a combination of both, can be used to good effect:

- Vertical scaling, which involves creating additional application server processes on a single physical machine in order to provide multiple thread pools, each corresponding to the JVM associated with each application server process.
- Horizontal scaling, which involves creating additional application server processes across multiple physical machines.

3.1.3 Throughput

Throughput while related to performance, more precisely involves the creation of some number of application server instances (clones) in order to increase the number of concurrent transactions that can be accommodated. As with performance the application server instances can be added through vertical and/or horizontal scaling.

3.1.4 Availability

Availability requires that the topology provide some degree of process redundancy in order to eliminate single points of failure. While vertical scalability can provide this by creating multiple processes, the physical machine then becomes a single point of failure. For this reason a high availability topology typically involves horizontal scaling across multiple machines.

Hardware HA

By providing both vertical and horizontal scalability the WebSphere Application Server runtime architecture eliminates a given application server process as a single point of failure. In Version 3.5 of WebSphere the capability to WLM the adminserver process further reduces the potential that a single process failure can disrupt processing on a given node. In fact the only single point of failure in a WebSphere domain/cluster is the database server where the WebSphere administrative repository resides. It is on the database server that any hardware-based High Availability (HA) solutions such as HACMP, Sun Cluster, or MC/ServiceGuard should be configured. There is very little to be gained from trying to configure WebSphere Advanced to work in conjunction with a hardware-based HA product; moreover it is not a supported configuration as of this writing. The only case where a hardware-based HA solution would provide value is where WebSphere Advanced was serving as the coordinator of a distributed (two phase commit) transaction. If a WebSphere node were to go down, than any in-doubt transaction (after prepare, before commit) could not be resolved automatically until the node was restored to service. Again, this is not a supported configuration at this time.

3.1.5 Maintainability

While maintainability is somewhat related to availability, there are a specific set of issues that need to be considered when deploying a topology that is maintainable. In fact some maintainability factors are at cross purposes to availability. For instance, ease of maintainability would dictate that one minimize the number of application server instances in order to facilitate online software upgrades. Taken to the extreme, this would result in an single application server instance, which of course would not provide a high availability solution. In many cases it is also possible that a single application server instance would not provide the required throughput or performance. Deciding on the degree of vertical and horizontal scaling that one needs to incorporate in a topology should also consider the matter of hardware upgrades (for example, adding CPU's, memory, or upgrading to faster CPU's). As we will see below, one alternative topology for maintainability involves creating more than one WebSphere domain.

3.1.6 Session state

Unless you have only a single application server or your application is completely stateless, then maintaining session state between HTTP client

requests will also play a factor in determining your topology. In WebSphere V3.x the only way to share a session between multiple application server processes (clones) is to persist the session to a database. Additionally, the configuration of an HTTP sprayer such as the Network Dispatcher component of WebSphere Edge Server needs to be considered when session state is important.

3.1.7 Topology selection summary

The following table is a summary of topology selection.

Table 1. Topology selection summary

	Security	Performance	Throughput	Maintainability	Availability	Session
Vertical Clones		Improved throughput on large SMP servers	Limited to resources on a single machine	Easiest to maintain	Process isolation	Required
Horizontal Clones		Best in general	Best in general	Code migration to multiple nodes	Process and hardware redundancy	Required
HTTP Separate	Allow for firewalls/ DMZs	Usually better than local	Usually better than local			
Three Tiers	Most options for firewalls	Typically slower than single JVM	Additional clones may improve throughput			
One Domain				Ease of maintenance		
Multiple Domains				Harder to maintain than single domain	Process, hardware & software redundancy	

3.2 Vertical scaling with WebSphere workload management

In the simplest case, one can configure many application server clones on a single machine, and this single machine also runs the HTTP server process. This configuration is depicted below in Figure 27.

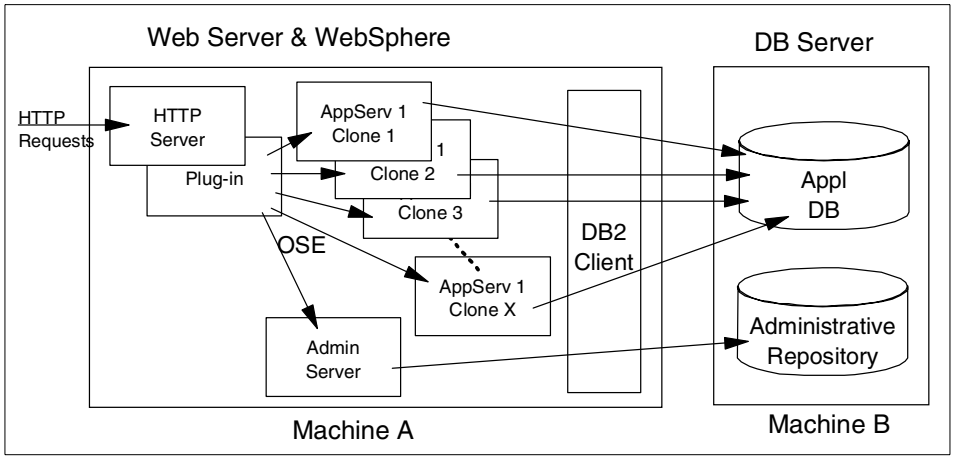

Figure 27. Vertical scaling with clones

While at first glance this would appear to be the simplest to configure, as we'll see later on, separating the HTTP server(s) from the application server processes on separate physical machines is not significantly harder with WebSphere than the simple case depicted here. Though one is limited to the resources available on a single machine as well as the availability risk when using a single machine, this configuration does provide for process isolation and improved throughput over a configuration where only a single application server process is running. This assumes of course that sufficient CPU and memory are available on the machine.

Database location

You will notice that even in the simple “single machine” configuration depicted above, the WebSphere configuration repository resides on a remote database server. There are several reasons why this represents a good practice.

First, most enterprises have already invested in a high availability solution for their database server, and the configuration repository represents a single point of failure in WebSphere, so it pays to make this highly available.

Second, the database that houses the configuration repository should be backed up on a regular basis, just as application data is. Housing the repository on the same server as the application data, usually simplifies this task since appropriate DBA procedures such as database backup processes are already defined for this machine.

Additionally, the database server is typically sized and tuned for database performance, which may differ from the optimal configuration for the application server (in fact on many UNIX servers, installing the database involves modification of the OS kernel).

Lastly, if one places both the database and application server on the same machine, then under load you have two processes: the application server and the database server, competing for increasingly scarce resources (CPU and memory), so in general one can expect significantly better performance by separating the application server from the database server.

How many clones?

In order to take advantage of WebSphere WLM and provide for application failover and throughput one needs to decide how many clones to create. The answer is dependent on the JDK, the application and the hardware environment.

JDK 1.2.x provides for much better parallelism than JDK 1.1.x, so with V3.5 of WebSphere a given number of clones should provide better throughput than will be realized for a given application and hardware environment running V3.02.

Obviously the larger the application (in terms of components), the larger the memory requirement for the JVM heap and thus the fewer number of clones that can be run on a given set of hardware. One should always monitor memory use when determining the number of clones (via `vmstat` on UNIX or the Windows NT Performance Monitor), and should not exceed the available physical memory on a machine.

In general one should tune a single instance of an application server for throughput and performance, then incrementally add clones testing performance and throughput as each clone is added. By proceeding in this manner one can determine what number of clones provide the optimal throughput and performance for their environment. In general once CPU utilization reaches 75% little, if any, improvement in throughput will be realized by adding additional clones.

The WebSphere Performance Lab has observed that in general there is a small benefit, if any, to be gained by creating multiple clones on a server with less than 4 CPUs. On machines with more than 4 CPUs, one should expect that the number of clones will be somewhat less than the number of CPUs. Precisely “how much less?” is highly dependent on your application, hardware environment and the JDK. Given the correlation between the application, hardware and JDK, the only way to determine the correct number for one’s environment is to add clones monitoring throughput and system resources as described above.

3.3 HTTP server separation from the application server

WebSphere provides four different alternatives for physically separating the HTTP server from the application server:

- OSE Remote
- thick Servlet Redirector
- thick Servlet Redirector Administrative-agent
- thin Servlet Redirector
- Reverse proxy / IP forwarding

When compared to a configuration where the application server and the HTTP server are located on a single physical server, each of these alternatives can be utilized to provide varying degrees of improvement in:

- Performance
- Process isolation
- Security

3.3.1 OSE Remote

As described in 2.2.2, “Remote OSE” on page 22, OSE Remote extends the OSE protocol “off box” to allow for physical separation of the HTTP server and the servlet engine(s). OSE supports clustering and workload management of application servers. This means that the HTTP server can send requests that require intensive processing to multiple application server machines, freeing up the HTTP server machine to process more requests. This option provides for both vertical (as depicted in Figure 28) and horizontal scaling of the WebSphere environment.

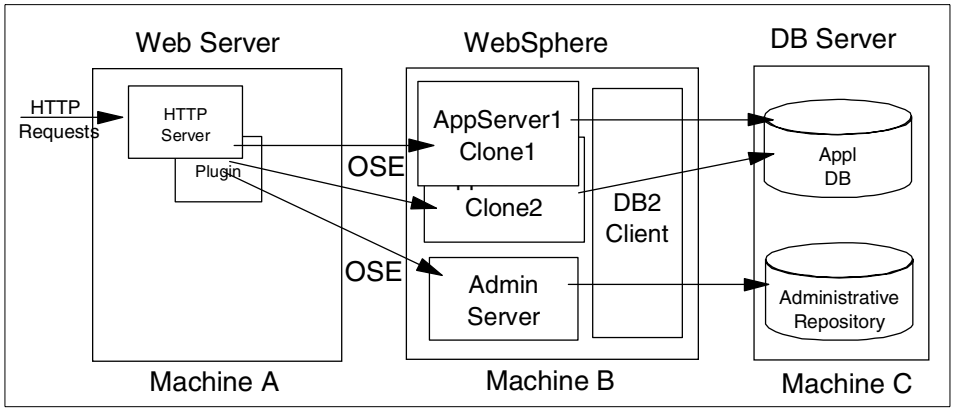

Figure 28. OSE Remote

The OSE link does not however support data encryption between the HTTP server and the application server, though it does not preclude use of HTTPS between the browser and the HTTP server. Thus where all data traffic, even that in a DMZ, must be encrypted, OSE Remote is not sufficient, even when WebSphere security is enabled. Recall that all WebSphere Application Server internal IIOP traffic implicitly uses SSL when security enabled. This is not true for OSE. In environments that require that all network communication be encrypted, the Servlet Redirector should be used instead of OSE Remote.

OSE Remote also provides better performance than the Servlet Redirector. Throughput with the Servlet Redirector is typically 15-30% slower than OSE running local. OSE Remote on the other hand performs nearly as well (within a few percent) as OSE local and in some configurations may perform better than OSE local by virtue of separation of the HTTP and application server processes.

In summary the benefits of OSE Remote are:

- Better performance than the Redirector.
- Supports WebSphere security.
- Does not need database access through a firewall.
- Support for NAT firewalls. See Appendix A, “Firewall considerations” on page 465 for more detail.

While the disadvantages to OSE Remote are:

- Manual configuration and administration of the HTTP server plug-in files when changes are made to the application server.

- OSE communication is not SSL encrypted, though one might wish to use a Virtual Private Network solution such as IPSec (IP Security) for communication security.

3.3.2 Thick Servlet Redirector

As outlined in 2.4, “Servlet Redirector” on page 41, the Servlet Redirector runs as a process on the same server as the HTTP server, intercepts the OSE protocol messages and forwards each servlet request over IIOP (or IIOP/SSL) to an appropriate servlet engine. The first of the Servlet Redirector configuration options is the thick configuration. In this configuration a WebSphere Administrative Server runs on the same box as the Redirector and takes care of process configuration management. In addition, this topology requires configuration of the appropriate database server client software on the HTTP server machine, which may not be prudent in a secure environment as discussed below.

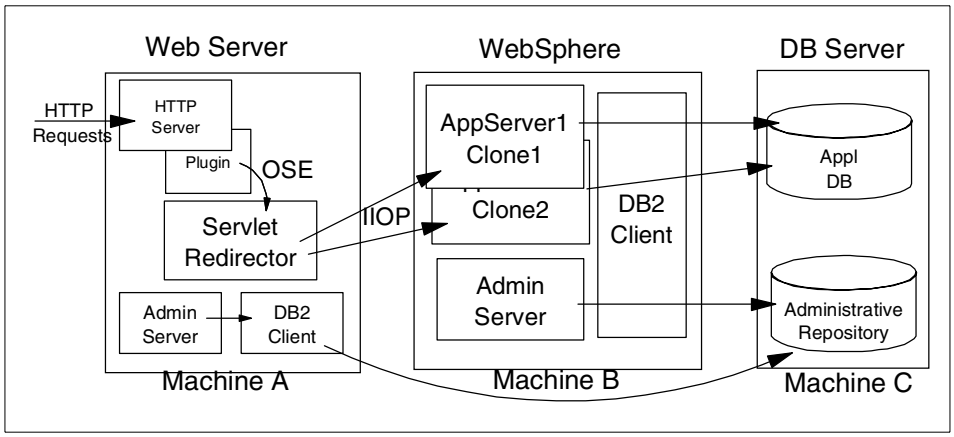

Figure 29. Thick Servlet Redirector

With the thick Servlet Redirector each WebSphere Administrative Server connects directly to the WebSphere Administrative Repository. This requires that either the database be installed locally or that the appropriate database drivers be installed and configured on the machine. More importantly from a security standpoint a database user ID and password must be stored on the machine for use by the database processes. And also the Administrative Server on the HTTP server machine requires a TCP connection to the remote database. Therefore you must open a port in the firewall to pass DB traffic. In topologies where security is a concern, such as a server running outside a firewall, this may not be acceptable.

In summary the benefits of a thick Servlet Redirector are:

- Ease of configuration and administration.
  - Automated update of HTTP server plug-in files when changes are made to the application server.
  - Ability to start and stop the Servlet Redirector from the Administrative Console.
- Supports WebSphere security for servlets/EJBs.
- Encryption of the communications protocol between the Servlet Redirector and the application server.

While the disadvantages to the thick Redirector are:

- Requires access to database through firewall.
- Requirement for database principal identify and password to exist outside the firewall.
- Does not support NAT firewalls.
- When used in conjunction with a firewall, the firewall must support IIOP traffic.

3.3.3 Thick Servlet Redirector Administrative agent

An alternative to configuring the database client software on the same machine as the HTTP server is the Administrative agent. In this alternative the administration server on the HTTP server is configured to run as an agent to an Administrative Server running on another server. As such the Administrative Server agent is the recipient of configuration and administration information from the Administrative Server running on the other node. The Administrative Server on the other node is responsible for communication with the WebSphere configuration repository.

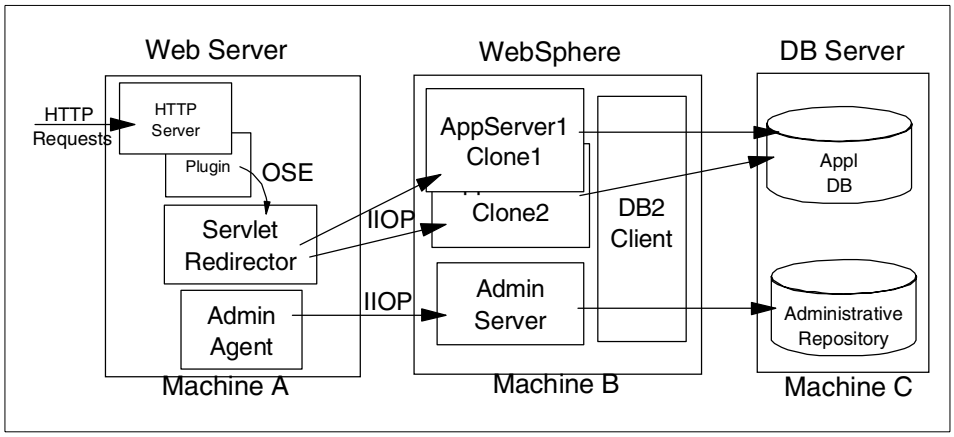

Figure 30. Thick Servlet Redirector Administrative agent

An alternative to the normal Administrative Server is to configure an Administrative Server to run in agent mode. In agent mode the Administrative Server processes on one machine attach to the Administrative Server process on another machine, and use the remote Administrative Server to connect to the WebSphere administrative repository. Doing so obviates the need to configure the required database components on a machine and also reduces the number of ports that must be opened through a firewall by eliminating the need to open a port or ports for database connections. This also reduces the number of processes running on a machine (for example, DB2 UDB server or a DB2 client) and helps to improve performance.

In summary the benefits of a thick Redirector in conjunction with an Administrative Server agent are:

- Supports WebSphere security.
- Encryption of the communications protocol between the Servlet Redirector and the application server.
- Automated update of HTTP server plug-in files when changes are made to the application server.
- Does not need database access through a firewall.
- Database clients and passwords need not be present outside a firewall.
- Ability to start and stop the Servlet Redirector from the Administrative Console.

While the disadvantages of the thick Redirector Administrative agent are:

- In V3.02.x of WebSphere the Administrative Server that the Administrative agent is connected to represents a single point of failure.

  Note: In V3.5, Administrative Servers can be WLM enabled, which provides failover for the Administrative Server and removes it as a single point of failure on a node.

- Does not support NAT firewalls.
- Requires fix for routing to clones in WebSphere 3.021.
- When used in conjunction with a firewall, the firewall must support IIOP traffic.

3.3.4 Thin Servlet Redirector

The thin Servlet Redirector provides an alternative to running a WebSphere Administrative Server on the same machine as the HTTP server. Since an Administrative Server is not running on the HTTP server, one must run a script to configure the HTTP server plug-in files as well as a script to start and stop the Redirector. These functions are normally provided by the Administrative Server.

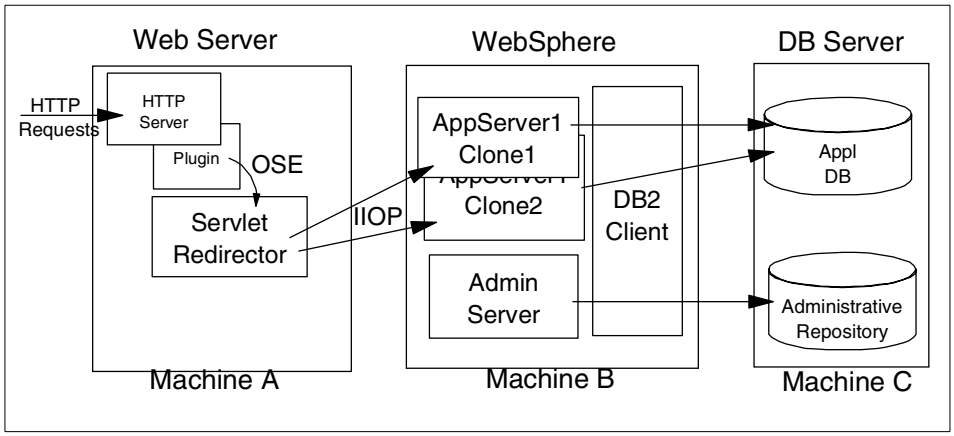

Figure 31. Thin Servlet Redirector

The thin Servlet Redirector provides SSL encryption of the IIOP protocol, but it requires that the HTTP server plug-in files be manually generated and that the Servlet Redirector be started and stopped via a script and not from the Administrative Console. Additionally by virtue of not running an administration server process on the HTTP server, more of the processing power available on the box may be used to service HTTP requests.

In summary the benefits of a thin Servlet Redirector are:

- Encryption of the communications protocol between the Servlet Redirector and the application server.
- Minimum process overhead on the HTTP server by virtue of eliminating the need for an adminserver process.
- Does not need database access through a firewall.

While the disadvantages to the thin Redirector are:

- Manual configuration and administration.
  - Manual update of HTTP server plug-in files when changes are made to the application server.
  - Start and stop of the Servlet Redirector by scripts.
- Remote Administrative Server becomes single point of failure.

  Note: In V3.5, Administrative Servers can be WLM enabled, which provides failover for the Administrative Server and removes it as a single point of failure on a node.
- Does not support NAT firewalls.
- When used in conjunction with a firewall, the firewall must support IIOP traffic.

3.3.5 Reverse proxy / IP forwarding

No additional WebSphere administration is required when setting up a reverse proxy configuration; the implementation specifics are determined by the reverse proxy server being used. In this configuration a reverse proxy that resides in the DMZ listens for incoming HTTP(s) requests and then forwards those requests to an HTTP server that resides on the same machine as WebSphere. The requests are then fulfilled and passed back through the reverse proxy to the client, hiding the originating Web server.

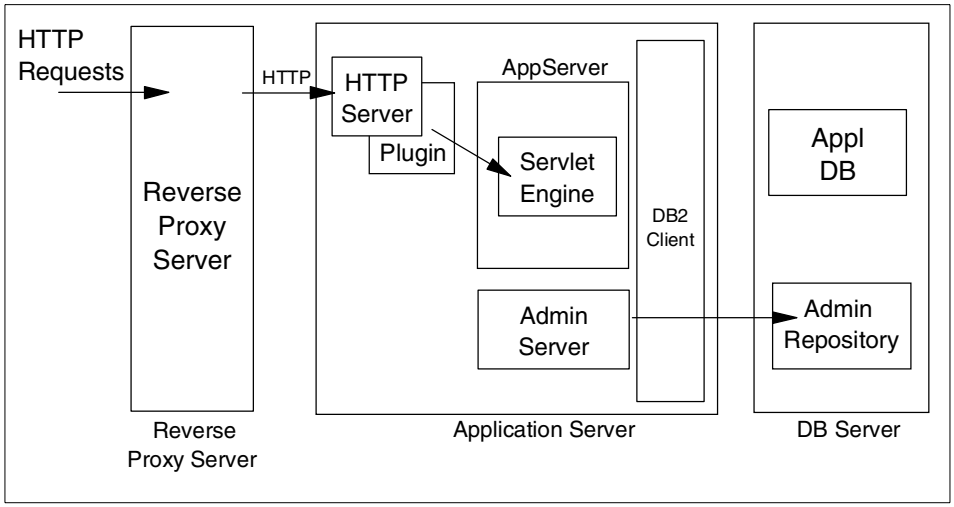

Figure 32. Reverse proxy/IP forwarding

In summary the benefits of the reverse proxy are:

- Does not require database access through firewall.
- Supports WebSphere security.
- Works with NAT.
- The basic reverse proxy configuration is well known and tested in the industry, resulting in less customer confusion.
- Uses HTTP.
- Well known, dependable format.
- Uses only one HTTP port for requests and responses.

  Note: This is also a disadvantage in some environments where policies prohibit the same port or protocol being used for inbound and outbound traffic across a firewall.

While the disadvantages to the reverse proxy are:

- No WebSphere WLM “awareness”.
- Requires more hardware and software.
- Customers may not want a reverse proxy in the DMZ.

3.3.6 HTTP server separation selection criteria

The following table is a summary of HTTP server separation selection criteria.

Table 2. HTTP server separation selection criteria

Alternative	SSL	DB Password Rq'd	WLM	NAT	Perfor-mance	Admin-istration
OSE Remote	No	No	Yes	Yes	High	Manual
Thick Servlet Redirector	Yes	Yes	Yes	No	Medium	Automated
Thick Servlet Redirector - Admin Agent	Yes	No	Yes	No	Medium	Automated
Thin Servlet Redirector	Yes	No	Yes	No	Medium	Manual
Reverse Proxy	Yes	No	No	Yes	High	Manual

One possible secure configuration utilizing multiple firewalls is shown in Figure 33.

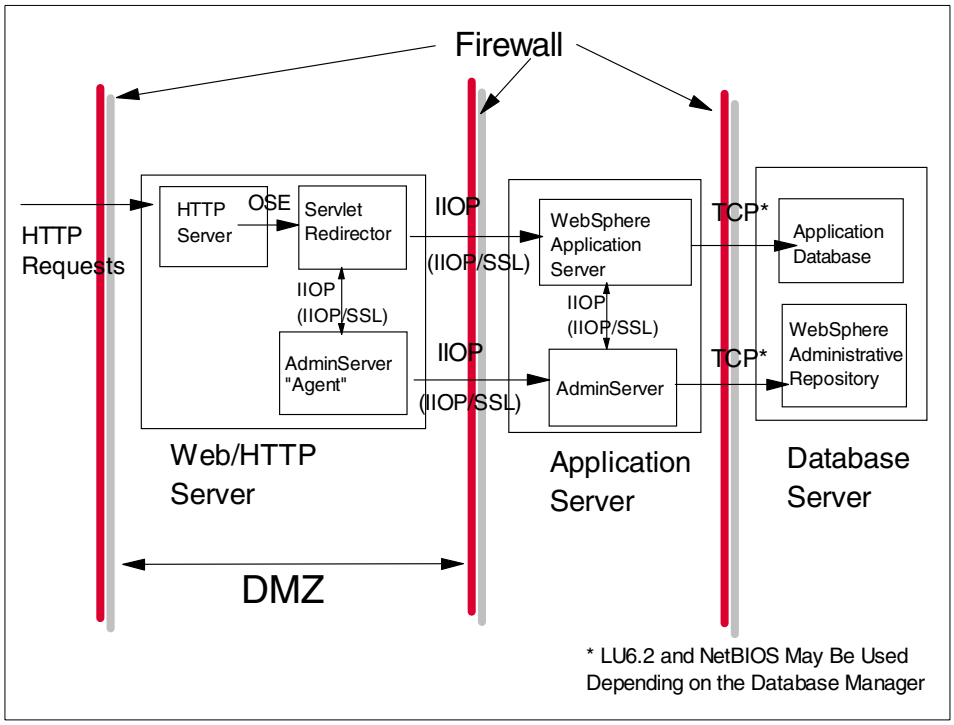

Figure 33. Sample firewall configuration with thick Servlet Redirector administrative agent

3.4 Scaling WebSphere in a three-tier environment

Partitoning your application server processes into servlet application servers and EJB application servers as depicted below can provide some advantages from a security perspective as well as some possible advantages from a performance perspective.

In this topology the EJB layer is closer to the application data, which an entity EJB provides a representation for. When running in an environment where two firewalls are employed, this allows one to provide the same level of security for entity EJBs as is provided for application data.

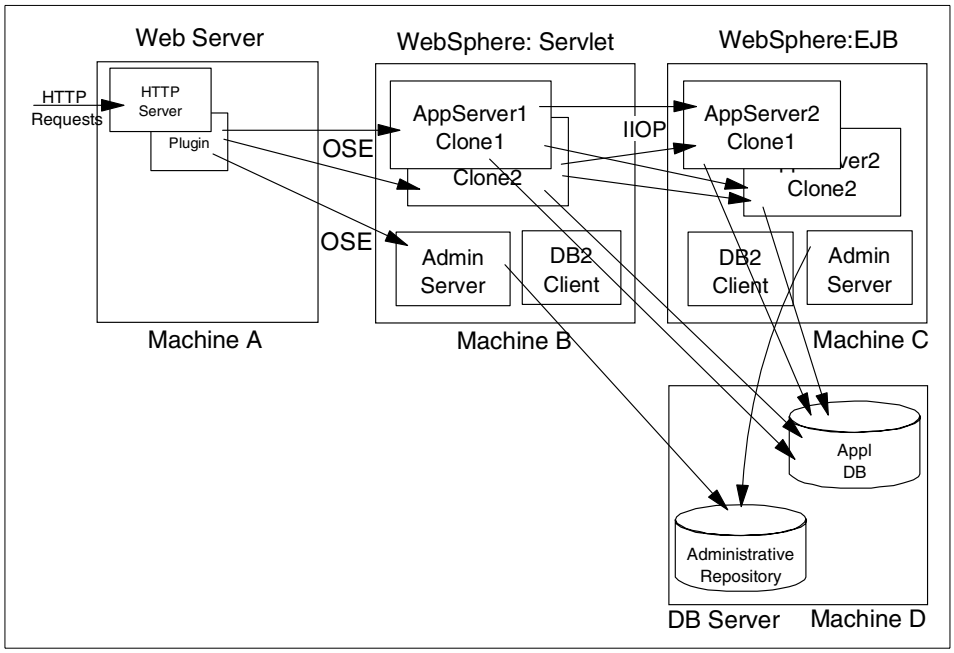

Figure 34. 3-tier environment

In terms of performance this topology allows one to replicate different numbers of application servers (for example, two servlet engines and five EJB servers or vice-versa), which may provide better performance. In general though elimination of the local JVM optimizations that occur when both the servlet (client) and the EJB (server) are resident in the same application server (JVM) as well as the network latency introduced with the topology will tend to negate any possible performance improvement brought about by virtue of the separate thread pools in each JVM and the additional processing power afforded by a separate physical server.

While providing more redundancy for application server processes, this topology also introduces more possible points of failure. In addition to more application server processes, this means that you'll have more to manage as well.

A 3-tier example

When running a performance test as part of writing this redbook where we partitioned two application servers, each running servlets and EJBs, into a single application server running servlets and a single application server running EJBs we noted a ~ 20% reduction in throughput. This is not unexpected since requests were no longer serviced from within a JVM, but instead incurred some network latency.

The available hardware (a 2-CPU workstation with 512 MB of RAM) at our disposal precluded us from creating additional application server clones, due to limited physical memory on the servers (recall from the previous discussion entitled “How many clones?” on page 53, that creating multiple clones on a machine with less than four CPUs was of limited value). With more powerful hardware, it might have been possible to provide the same or additional throughput in this topology. In any event, you will want to carefully consider the security and possible performance trade-offs in this topology for your specific hardware environment.

3.5 Horizontal scaling HTTP servers within a WebSphere environment

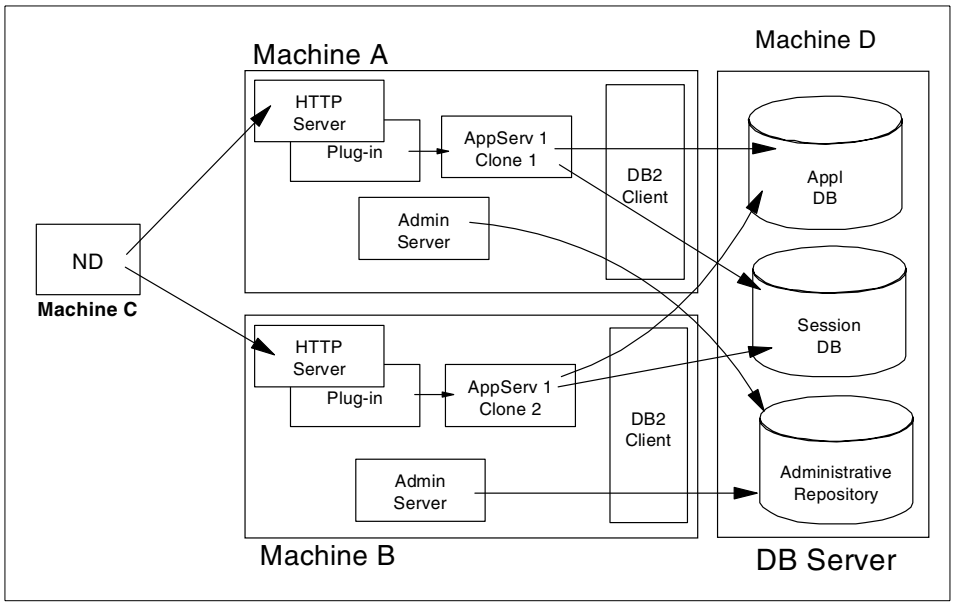

Figure 35. WebSphere with Network Dispatcher

Adding a mechanism for distributing HTTP requests, such as the Network Dispatcher component of WebSphere Edge Server as depicted above provides the following advantages:

- Network Dispatcher allows for increased number of connected users.
- Network Dispatcher eliminates the HTTP Server as a single point of failure and can be used in combination with WebSphere WLM to eliminate the application server as a single point of failure.
- Increased throughput by virtue of adding multiple servers and CPUs to servicing the workload.

Session or Sticky Bit?

Recall from the discussion in 3.1.6, “Session state” on page 49 that if maintaining session state across requests is a requirement for your application then you will need to persist session. Consider the case depicted below in Figure 36 on page 69 where multiple application server processes are running on a single machine. The first request is dispatched to machine A and App Server 1, a getSession() call is made to establish session in the JVM specific to the application server instance, and a cookie is created for the browser (or URL rewriting is performed). On the next invocation from the browser, the IP affinity (“sticky bit”) gets you back to the same physical server, but not necessarily to the same application server instance on the machine. If you don't go back to the same server instance, then there is no session object established for you, it's in memory associated with the other application server process/JVM. Also consider the case where you have two machines in a cluster, both running the same application server process (two clones in WebSphere server group). On the first request you go to machine A and establish your session object. A few seconds pass, the next request comes in from that customer and machine A has suffered a hardware failure so ND is now sending all requests to machine B. Once again, you're without a session object so your customer has to start all over again.

Some other issues to consider in using sticky bit with ND are:

Sticky bit has a time-to-live timer (configurable, usually 300 seconds), when the connection terminates it starts counting down. If it goes to zero the next connection request will go through load balancing and may be sent to a different server.

Load balancing may not work properly if relying on a “sticky bit” since ND only sees the client source IP address. All clients connecting through the same client-side proxy will all have the same source IP address, so will be directed to the same Web server. That thwarts load balancing, and can cause hotspots or overloaded Web servers, depending on the overall distribution of client IP addresses. With intranets, you can also have trouble with firewalls that do network address translation (so all packets appear with the same source IP address).

The use of a “sticky bit” does have the advantage of being simple to configure, requires no coding, and does not incur any application server performance overhead, since WebSphere doesn't have to check its state and possibly retrieve session data from a central store.

3.6 One WebSphere domain vs many

While there are no “hard” limits on the number of nodes that can be clustered in a WebSphere domain, one may want to consider creating multiple WebSphere domains for a variety of reasons:

- Two (or more) domains can be employed to provide not only hardware failure isolation, but software isolation as well. This can come into play in a variety of situations:
  - When deploying a new version of WebSphere. Note that nodes running WebSphere V3.02.x and V3.5 in the same domain is not supported.
  - When applying an e-fix or patch.
  - When rolling out a new application or revision of an existing application.
- In cases where an unforeseen problem occurs with the new software, multiple domains prevent a catastrophic outage to an entire site. A roll-back to the previous software version can also be accomplished more quickly. Of course, multiple domains imply the software has to be deployed more than once, which would not be the case with a single domain.
- Multiple smaller domains may provide better performance than a single large domain, since there will be less interprocess communication in a smaller domain.

Of course multiple domains will require more effort for day-to-day operations, since administration must be performed on each domain, though this can be mitigated through the use of scripts employing wscp and XMLConfig. Multiple domains also mean multiple administration repositories (databases), which means multiple backups here as well.

In order to distribute requests across multiple domains as depicted in Figure 37 on page 69, you’ll need a mechanism, such as Network Dispatcher for spraying your HTTP requests across the HTTP servers associated with each domain.

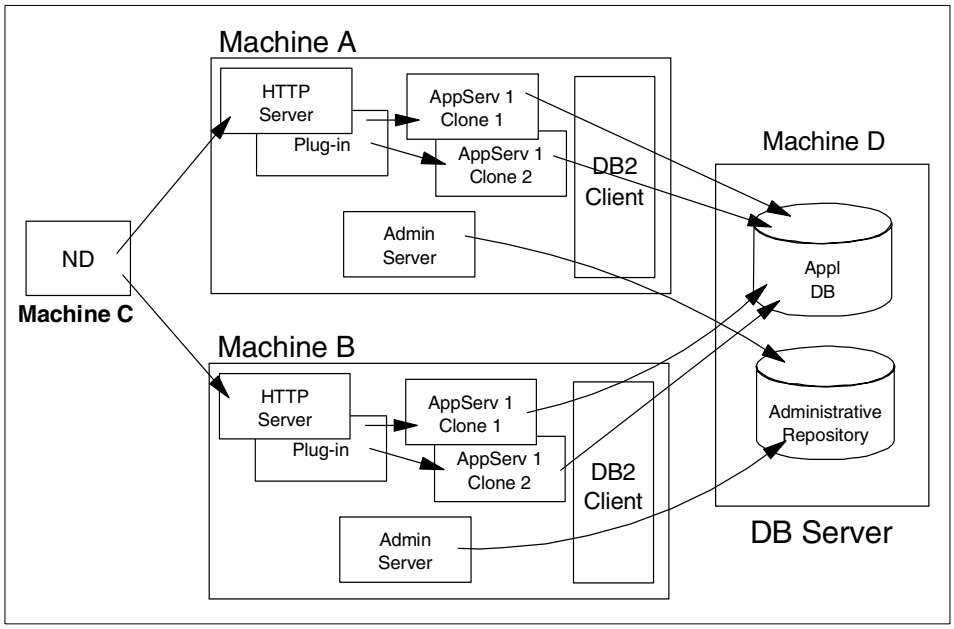

Figure 36. One WebSphere domain sharing both Appl DB and Administrative Repository

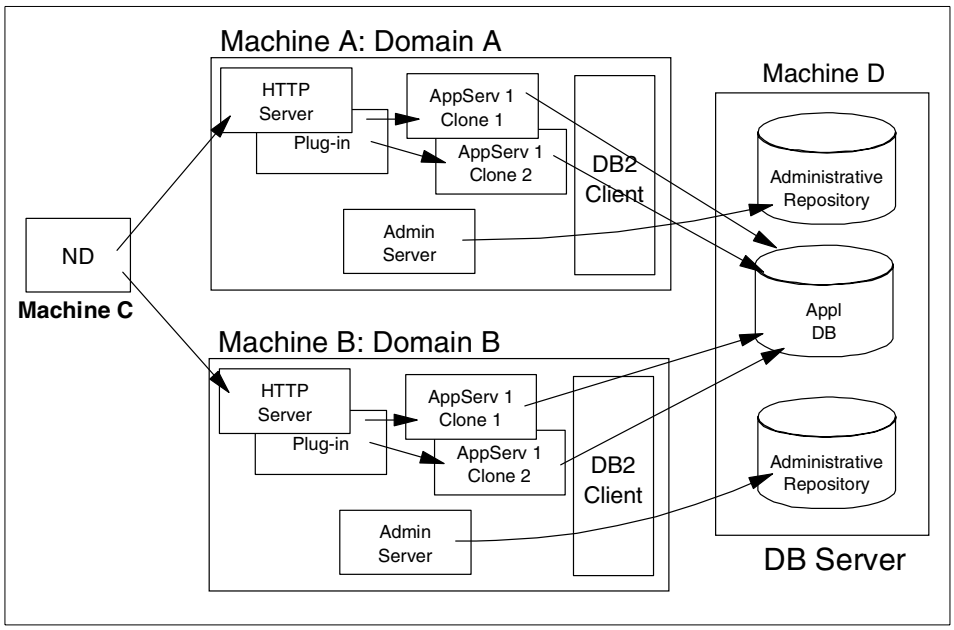

Figure 37. Multiple WebSphere domains with common Appl DB, but different Admin Repository

3.7 Multiple applications within one node vs one application per node

When deciding how to deploy your application one decision point is whether to deploy clones of an application server across all nodes in a cluster as depicted in Figure 38 on page 71, or to place all clones of a given application server on a single node as depicted in Figure 39 on page 72.

As discussed previously, horizontal cloning provides, as depicted in Figure 38 on page 71 provides:

- Process isolation
- Application software failover
- Hardware failover

As with horizontal cloning, vertical cloning as depicted in Figure 39 on page 72 provides for process isolation and application software failover, but obviously does not provide for any sort of hardware high availability. The primary advantage to vertical cloning is that the executables for a given application only have to be distributed to a single machine. Horizontal cloning on the other hand requires that your application executables be distributed across multiple machines in a cluster. Use of a file system, such as NFS or AFS, that provides a common file mount point for all nodes can ease the distribution of code across multiple nodes.

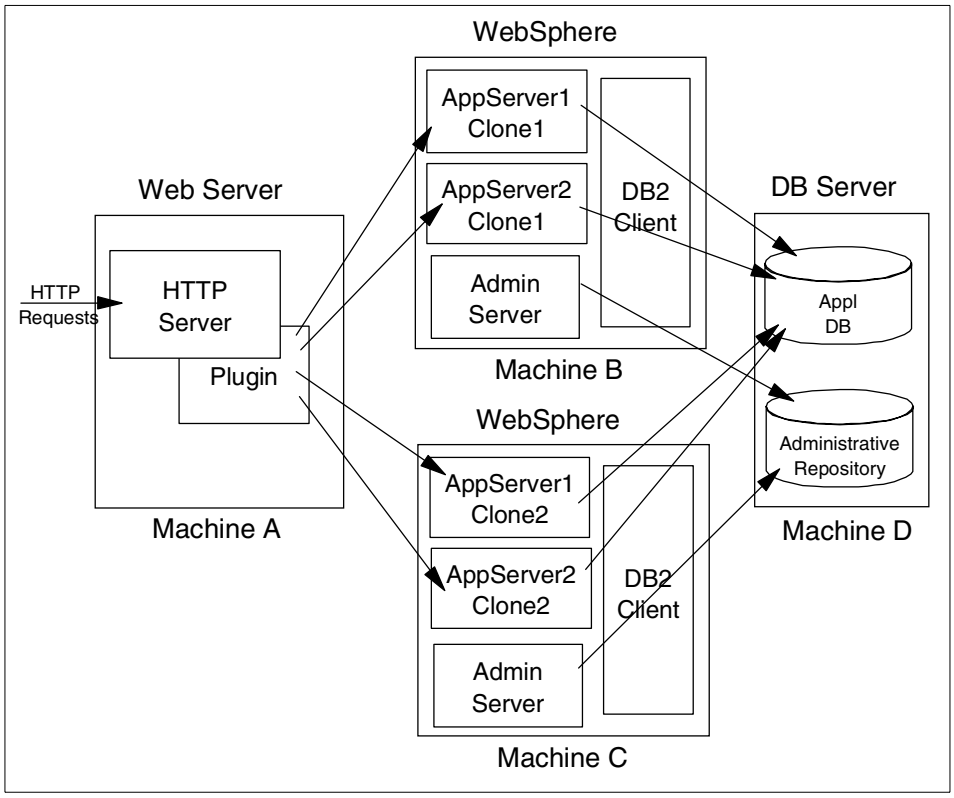

Figure 38. Multiple AppServers within one server

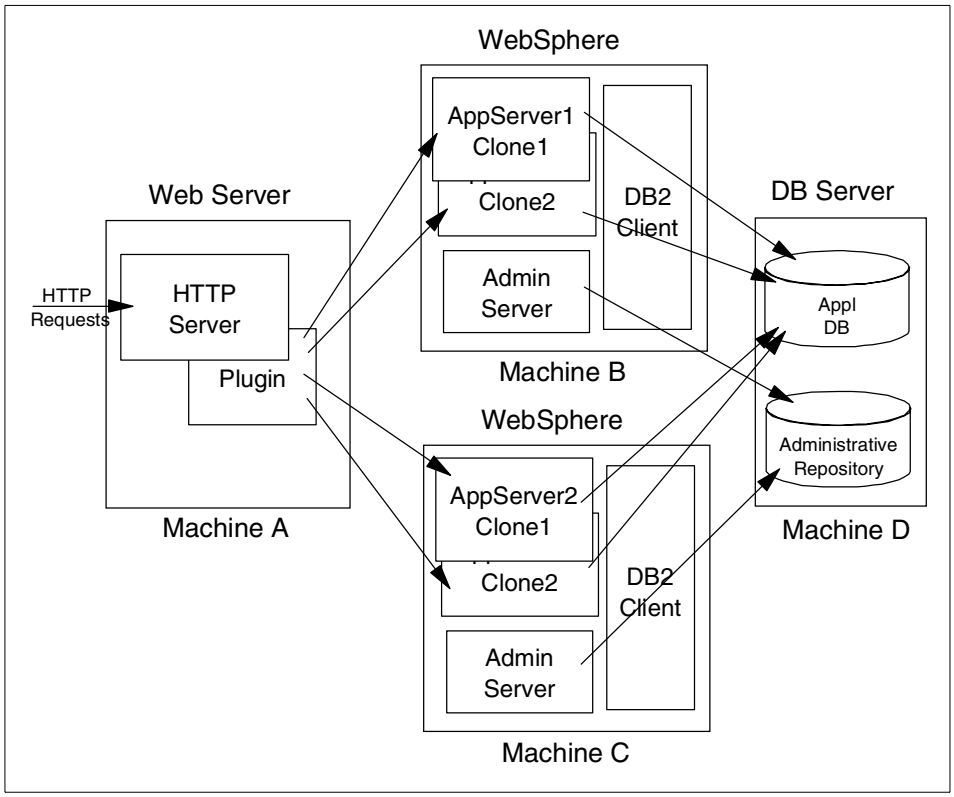

Figure 39. One AppServer per node

3.8 Closing thoughts on topologies

Whatever topology you decide on, a best practice is to partition your production acceptance environment exactly the same as your production environment. This avoids “surprises” when deploying your application into production.

Another consideration, when practical for your application architecture, is to create a number of smaller application servers, rather than a single large one.

This has at least two advantages:

- The plug-in config files can be smaller (less complexity of URIs), which leads to better startup performance and possibly better execution performance.

- At least during the development phase, it takes less time to cycle a smaller appserver to pickup various configuration changes.

Of course, as noted in 3.4, “Scaling WebSphere in a three-tier environment” on page 63, creation of multiple application servers in this manner must be carefully balanced against the increased complexity of doing so, and the potential increase in response time due to interprocess RMI/IIOP calls and network latency.

If after looking at all the topologies you’re still wondering “what would be the best possible topology” one possibility is depicted below in Figure 40.

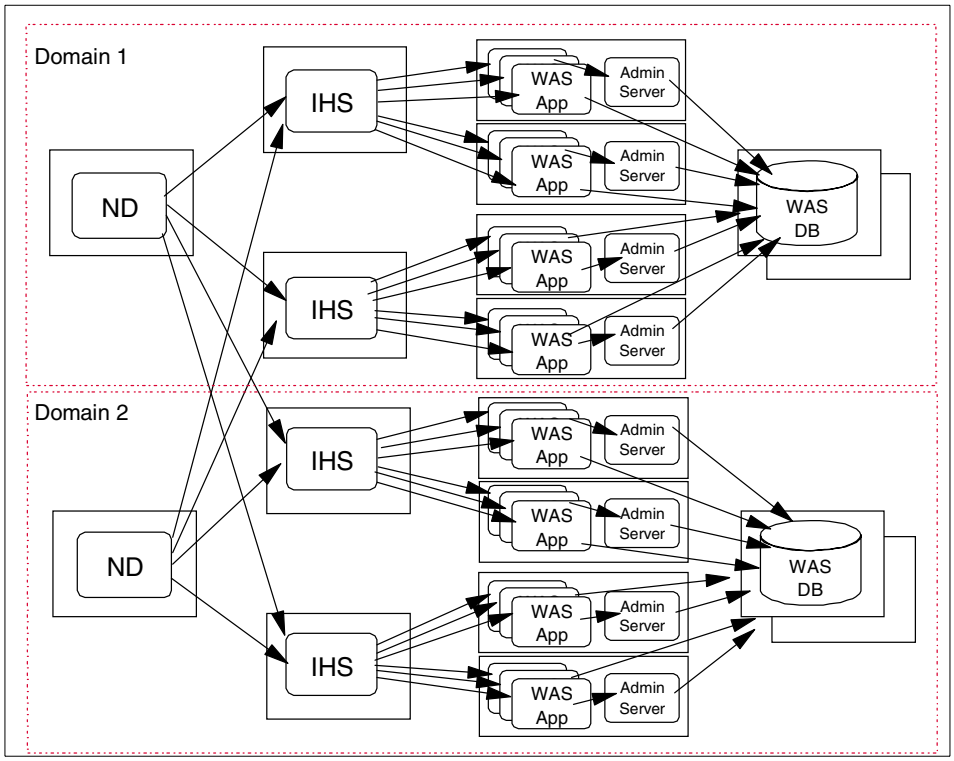

Figure 40. The best possible topology

This topology includes two of everything:

- Domains
- Database servers (two for each domain)
- Application servers (two for each domain)
- HTTP servers (two for each domain)
- HTTP dispatchers (IBM Network Dispatcher or a similar product)

Only the loss of an entire domain will be normally noticed by users. If this were to occur, the active HTTP sessions would be lost for 1/2 of the users, requiring that they log in again. Of course performance would most likely be degraded in this case as well, but it would ensure continuous availability. In most cases, hardware or software upgrades would be handled on a domain by domain basis during non-peak hours. If customer satisfaction (performance, throughput, availability) is important for your Web site then this would be the appropriate topology.

Part 2. Setting up your topology for UNIX and Windows

Chapter 4. Application server clones

WebSphere cloning provides the mechanism for scaling of WebSphere Application Servers. Servlets, EJBs, and Web resource executables can be shared by the clones, but each clone uses its own Java Virtual Machine (JVM) to run the application code. This provides identical, yet independent processes for the application to run in. For more information see 2.1, “Cloning” on page 15.

This chapter provides instructions for the administrative tasks required to configure a single machine to serve as the HTTP server and application server (an example of vertical scaling). The application server will have clones based on a single model.

Our purpose here is to clone the Default Server and all of its contents, such as servlet engine and EJB container. WebSphere allows you to create clones from a model, which is a template. So, we need to create a model first. We specify all the properties we want in the model. Clones assume properties from the model. When you change the properties of a model, the change is not automatically propagated to all of its clones. One has to restart the model. In V3.5, a "ripple restart" capability has been added to the model and this can be used for propagating changes without taking the model offline. Ripple restart runs through the set of clones, stopping and restarting them one at a time so that some clones are available all the time. True dynamic refresh may be implemented in a future release. You may create a model from scratch, or you may use an existing instance as a basis for the new model. In our example, we take the latter approach. That is, we are using the Default Server, which was created automatically by WebSphere when you chose the option, Configure administrative domain with default application server and a default application, in the WebSphere installation process.

We will discuss the following steps for setting up this configuration.

1. Product installation
2. Create the model
3. Configure the model for WLM
4. Create the clones
5. Start the servers
6. Test the servers
7. Related topologies

4.1 Overview of the configuration

A single machine will run the HTTP server and application server clones. A Database Server will reside on a separate machine.

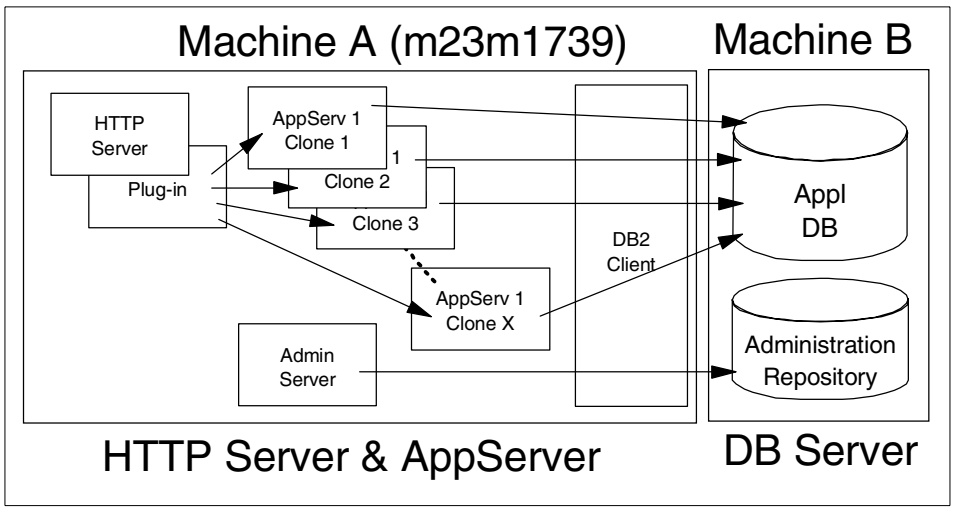

Figure 41. Clones of an application server

4.2 Installation summary

Table 3 is a summary of the software installation steps. We will not discuss the software installation. Please refer to the *Getting Started* documentation which is part of the WebSphere product.

Table 3. Software installation steps

Step No.	Machine A (HTTP/WAS)	Machine B (DB)
1		Install UDB Apply FixPack
2		Create DB
3	Install DB2 client Apply FixPack	
4	Catalog node Catalog db	
5	Install JDK (V3.02x)	
6	Install Web server	

Step No.	Machine A (HTTP/WAS)	Machine B (DB)
7	Install WebSphere (Production Application Server; -Plugins, -Core Server)	

4.3 Start the Administrative Console

Before you start configuring the models and clones, you need to start all the software that you need:

1. Start a DB2 instance for the Administrative Repository on Machine B if it is not started.
2. Start Administrative Server on Machine A.

   In our example, we specified “install.initial.config=true” in the admin.config file to install the Default Server.
3. Start Administrative Console on Machine A.

   Note: You can use any machine for your Administrative Console.
4. When the console finishes initializing, make sure you see Machine A in the Topology page as shown in Figure 42.

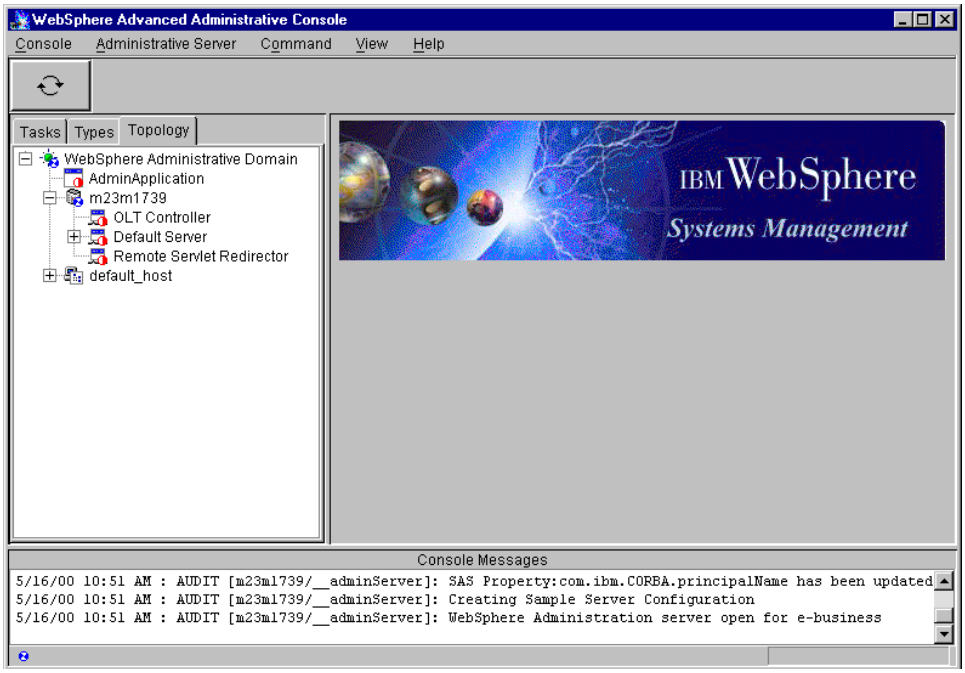

Figure 42. WebSphere Advanced Administrative Console

4.4 Create the model

A model is a template for creating additional, nearly identical copies of a WebSphere item, such as an application server or servlet engine. The copies are called clones.

In the Administrative Console, select the **Topology** tag. Expand the node (m23m1739). Locate the application server in the list. The default server that is created is called Default Server. For our sample configuration, we used Default Server as our application server. In your production environment, you probably will not want to use Default Server.

To create the model, right click on the Application Server to be cloned, and select **Create ->Model** as shown in Figure 43.

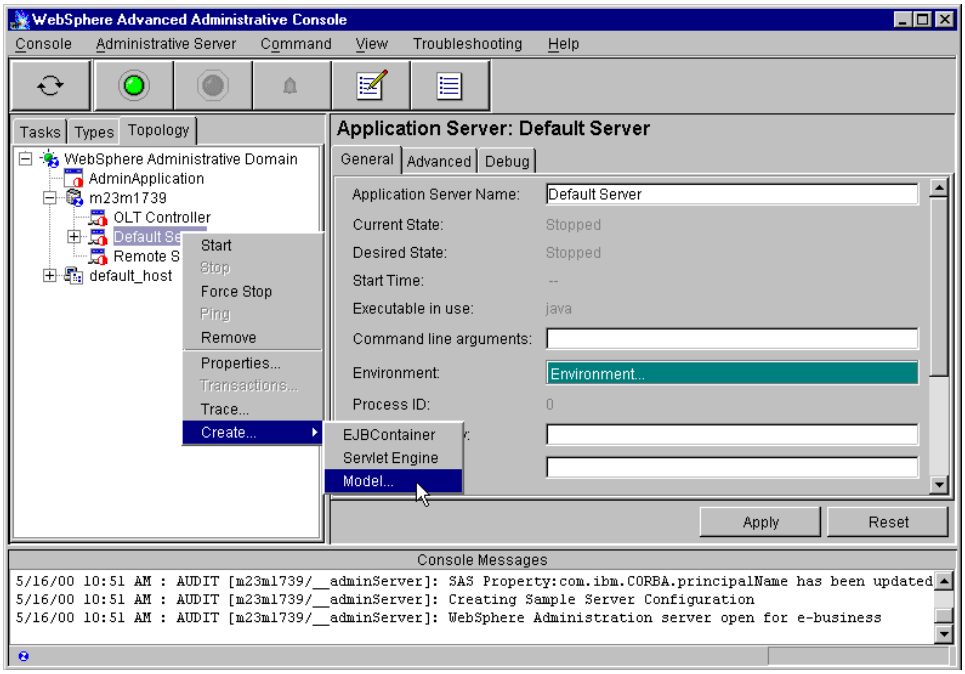

Figure 43. Create model

The Clone Properties window will appear. Enter a model name and check the box to **Recursively Model all Instances under the Server** as shown in Figure 44.

Click the **Create** button.

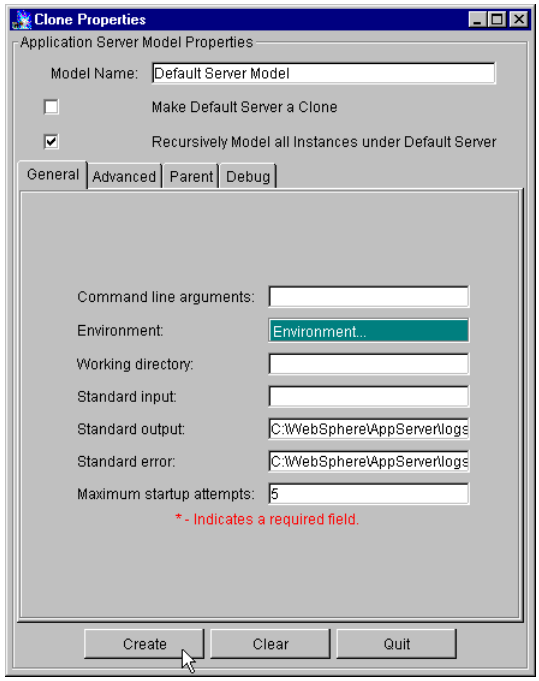

Figure 44. Clone properties

Note

You might not want to convert your initial Web application into a clone; however if you plan to enhance this topology to utilize the Servlet Redirector with V3.021, then you should click the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model's clones to fail unless this server is made into a clone.

If you are using V3.5, you should tick the option to make this server a clone. If you don't, you cannot create a clone.

When the model has been created you should see a message as shown in Figure 45.

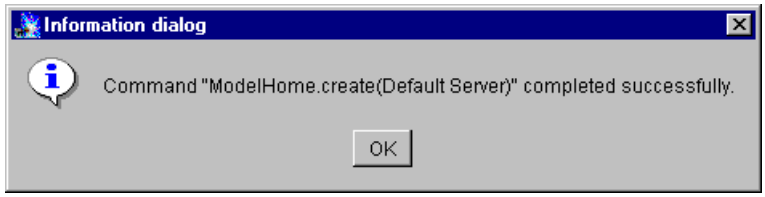

Figure 45. Model created message

4.5 Configure the model for WLM

Select the model under the Topology tab. Choose the **Advanced** tab. The default for Workload management selection policy is Round Robin Prefer Local. Since all the clones will reside on the local node, we will not change this value. For more information, see 2.3, “WLM” on page 23.

When **Round Robin Prefer Local** is selected, all requests that can be served locally will be served locally. If we were to create two clone nodes, we would need to select **Round Robin** in order for all the clones to be used evenly.

Select the policy the model is to use as shown in Figure 46.

Click the **Apply** button if you updated.

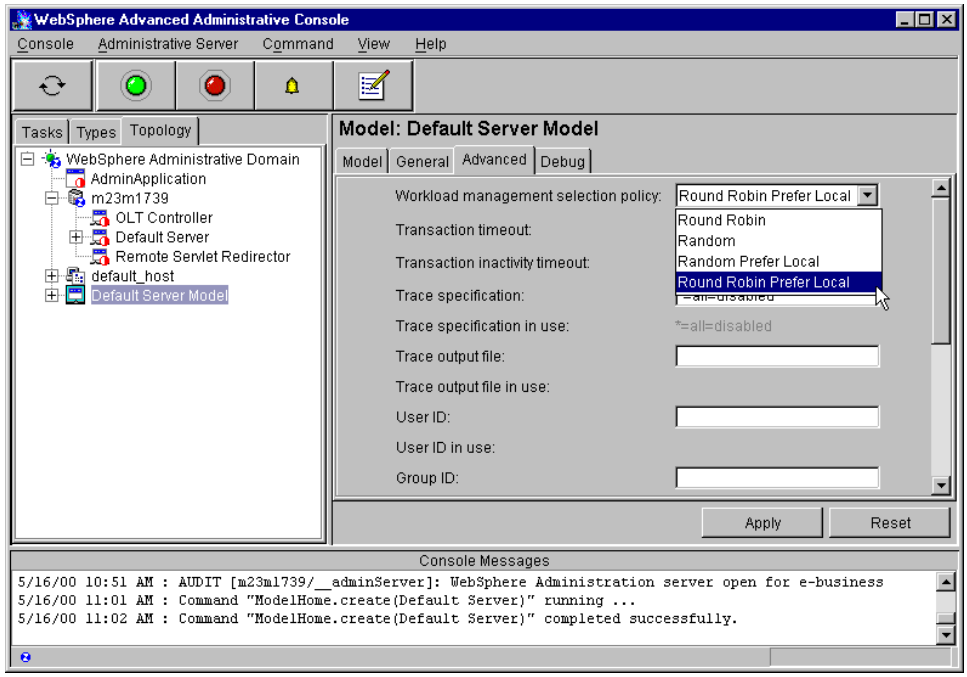

Figure 46. Model WLM

4.6 Create the clone

Using the model you created, you can now create a clone. To do this, right click on the model and select **Create->Clone** as shown in Figure 47 on page 85.

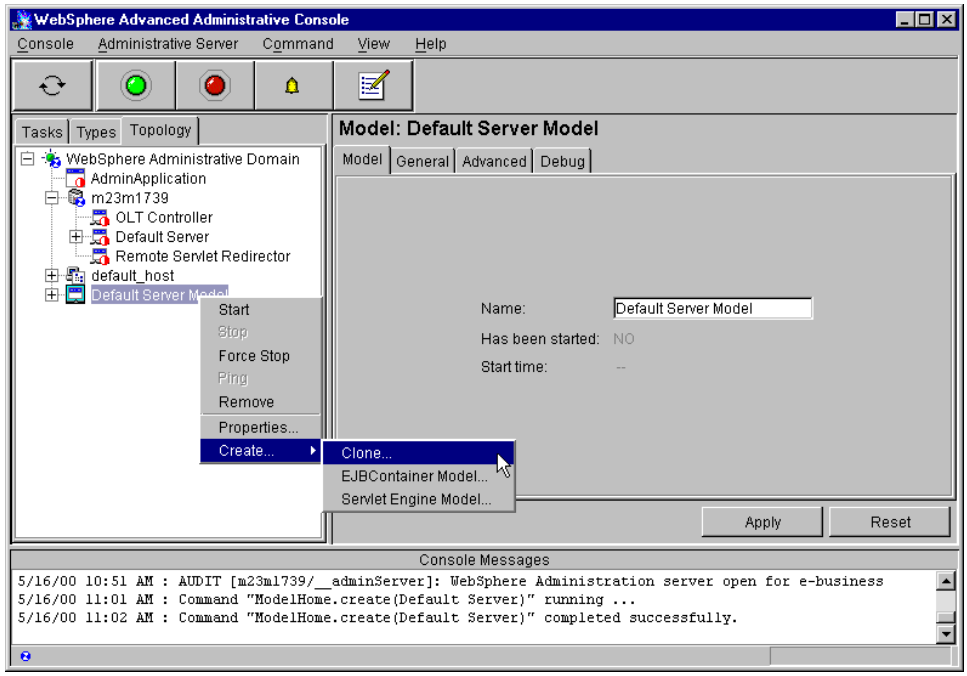

Figure 47. Create clone

The Clone Parent window will appear. Enter a clone name and select the node for the clone as shown in Figure 48.

Click the **Create** button.

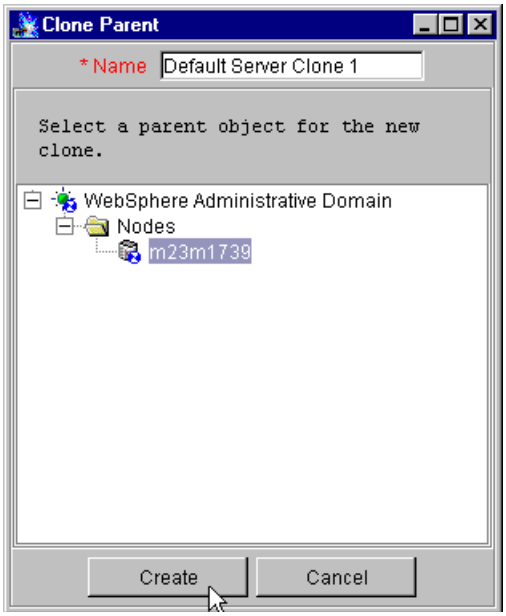

Figure 48. Clone parent

Note

If you selected the option to make the server a clone when you generated the model, you will already have one clone associated with the model (the server you cloned).

When the clone has been created, you will see a message as shown in Figure 49.

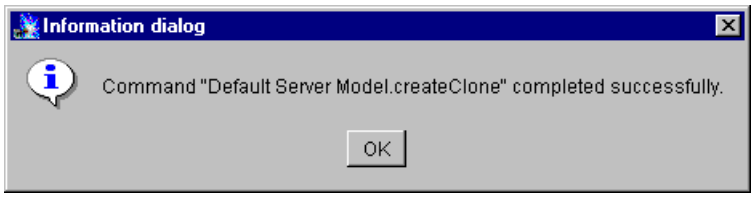

Figure 49. Clone created message

Additional Clones

Repeat this process for each clone you wish to create.

The clones will appear under the **Topology** tab as shown in Figure 50.

In our case, the clones are called `Default Server Clone 1` and `Default Server Clone 2`. The model is called `Default Server Model`. The model was created from the `Default Server`. The `Default Server` is still in the **Topology** tab, but since we did not select the option to make it a clone, it is not associated with the model.

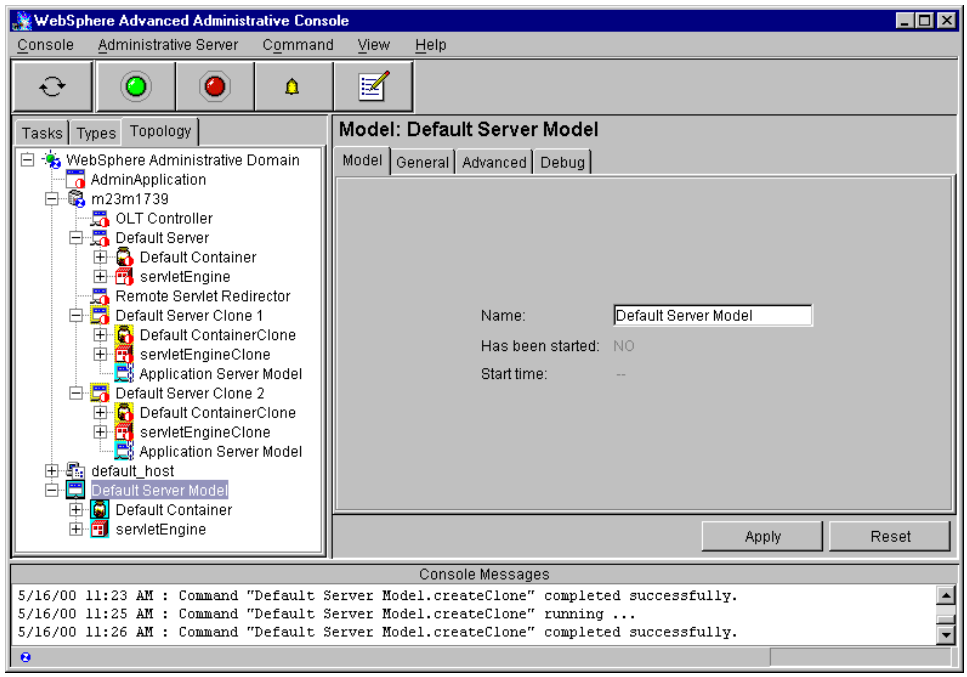

Figure 50. Clones in the Topology tab

4.7 Start the servers

We need to start the WebSphere clones and HTTP server.

4.7.1 Start the clones

The clones can be started two ways:

1. Together: using the model
2. Individually: using the clone

4.7.1.1 Starting the clones using the model

Starting the model will start all the clones associated with the model.

Start the model by right clicking on the model name in the Topology tab and select **Start** as shown in Figure 51.

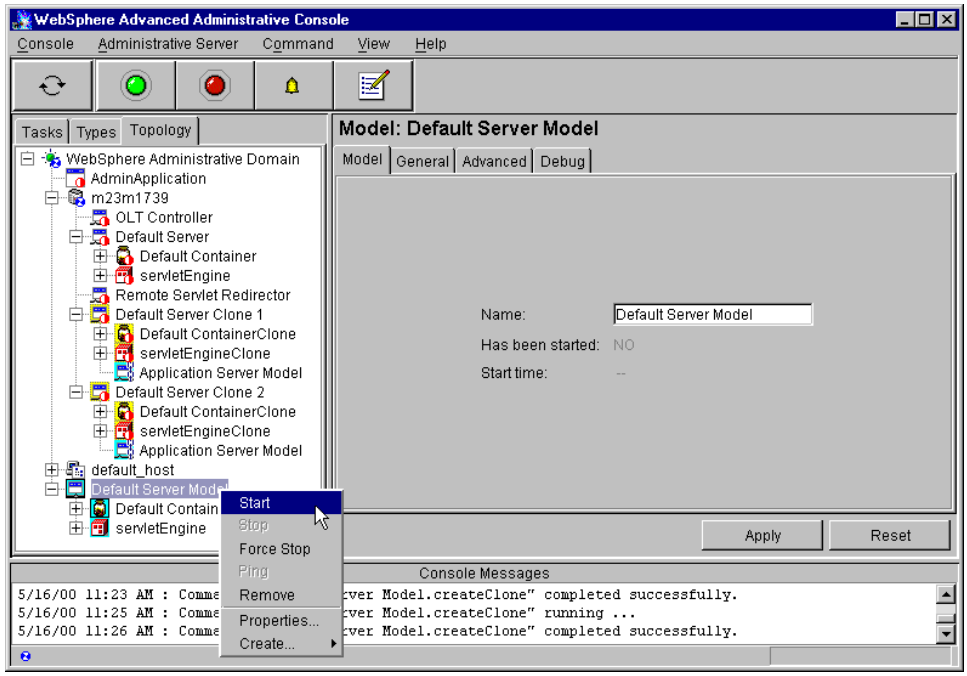

Figure 51. Start the model

When the model has started, you will see a message as shown in Figure 52.

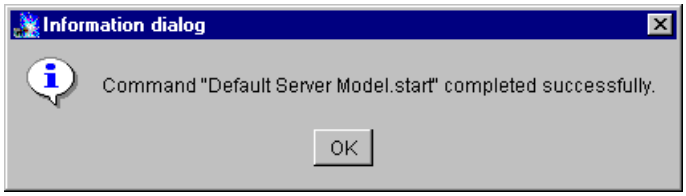

Figure 52. Model started message

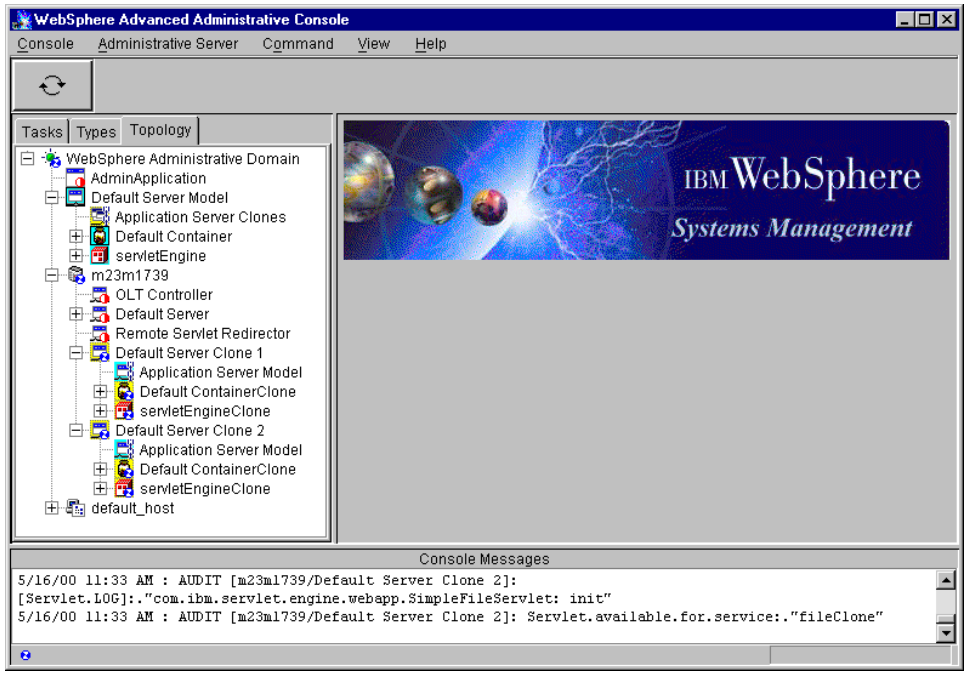

Figure 53. All clones are started

Note

At the time of writing, an issue with the Administrative Console, caused clone icons not to always display when started. We have found that refreshing the node subtree will only occasionally correct the icons. Stop and restart the Administrative Console to display the correct information.

4.7.1.2 Starting the clones individually

Any clone can be started individually. This is done by selecting the clone from the Topology tab, right clicking on it and selecting **Start**.

4.7.2 Start the Web server

Start the Web server on machine A.

4.8 Test the server

Now that the HTTP server and the WebSphere clones are running, we need to test the servers to make sure that everything is configured properly. Using a Web browser, access the following URI. In our example, m23m1739 is the HTTP Server. You should specify your URL instead of the following URL:

```
http://m23m1739/webapp/examples/showCfg
```

Figure 54 shows the results of the request.

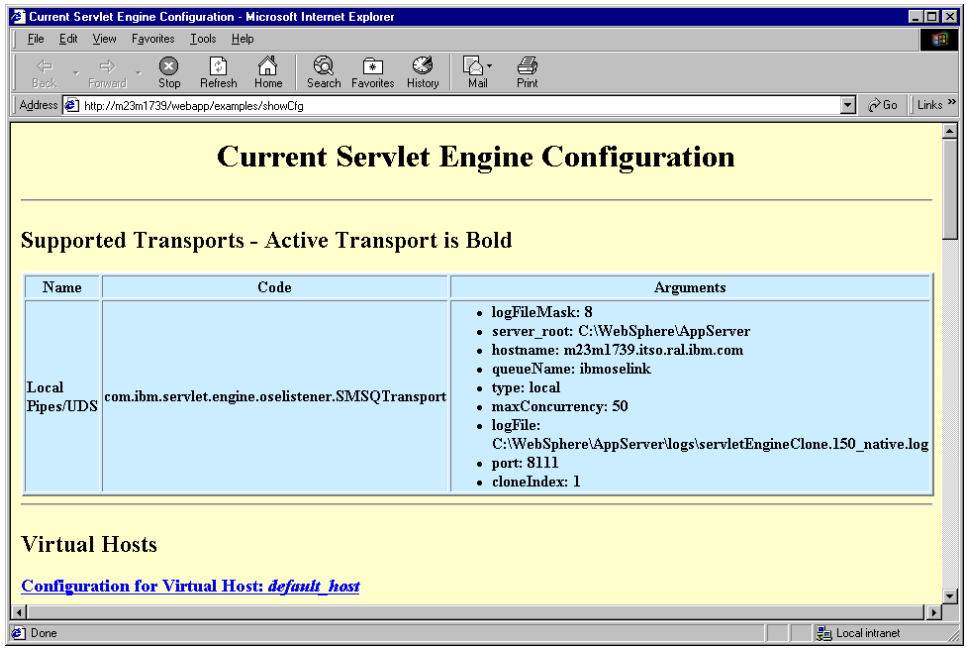

Figure 54. Results in browser 1

Notice that the request was served by Clone1 (cloneIndex=1).

A second call to the same URI will verify that we are using Round Robin Prefer Local for WLM:

Figure 55 shows the results of the request.

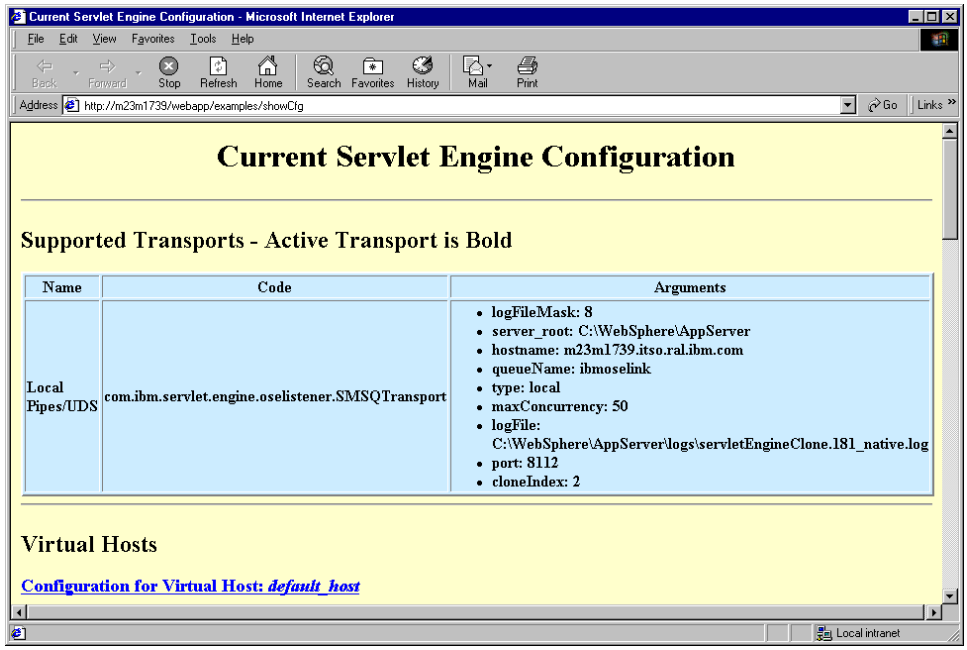

Figure 55. Results in browser 2

Notice that the request was served by Clone2 (cloneIndex=2).

4.9 Related topology

Cloning models of application servers provides a way for vertical scaling of the WebSphere environment.

4.9.1 Clones of distinct application servers

A single machine will run the HTTP server and application server clones. The Database Server resides on the separate machine. The system will contain two distinct application servers where each has a distinct Web application which has been cloned.

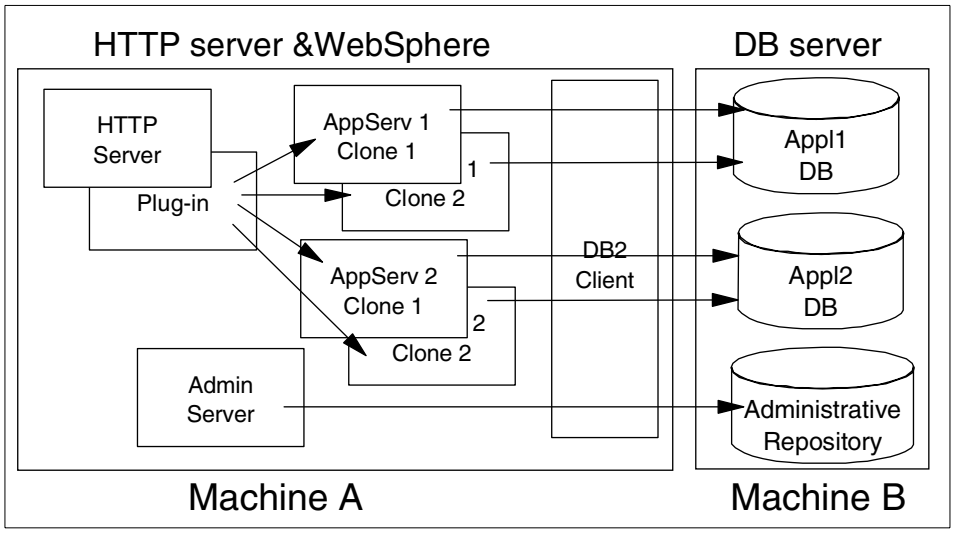

Figure 56. Cloning of multiple Web applications with a single node

This configuration can be achieved by configuring two distinct application servers as described in D.1, “Definition and configuration of an application server” on page 499 and following the cloning process discussed in this chapter for each application server.

4.10 Maintenance/troubleshooting

There are some considerations when doing maintenance on the application server.

4.10.1 Terminating an application server

When an application server aborts suddenly (we used the `kill -9 PID` command to simulate this situation), WebSphere automatically restarts the server. No action is required.

4.10.2 Stopping an application server

When an application server is stopped from the Administrative Console, WebSphere will automatically route new requests to alternative servers.

Of course, if the application server has no clones, no client requests will succeed. No additional actions are required.

4.10.3 Restarting a clone

When a clone is restarted, the routing of the requests will automatically be served by it. No additional actions are required.

Chapter 5. OSE Remote

The WebSphere Application Server uses a proprietary protocol called OSE to communicate between the plug-in and the servlet engine. When running both the HTTP server and WebSphere on the same machine, OSE is usually run over local pipes. If the HTTP server and WebSphere are running on different machines, we can use OSE Remote over TCP/IP.

OSE Remote allows the HTTP server to run on a separate machine and send requests to application server(s) running on remote machines. OSE supports clustering and workload management of application servers. This means that the HTTP server can send requests that require intensive processing to multiple application server machines, freeing up the HTTP server machine to process more requests. This provides both vertical and horizontal scaling of the WebSphere environment.

If you want to place a firewall between the HTTP Server and the application server nodes, you will need to open one port for each application server taking requests from this web server. If WebSphere's security is being used to secure resources which reside on the HTTPserver node, you will also need to open a port to allow the plugin to talk to the Administrative Server. It makes the configuration of the firewall simpler than using Servlet Redirector. In addition, OSE Remote will work with Network Address Translation (NAT) while Servlet Redirector will not. However, OSE Remote does not support encryption of the data stream between the HTTP server and WebSphere. Servlet Redirector supports SSL encryption of the data stream.

This chapter provides instructions for the administrative tasks required to configure the plug-in for OSE Remote and remote application server machines.

We will discuss the following steps for setting up this configuration:

1. Configuration overview
2. Installation summary
3. Configure the application server
    a. Configure the host aliases
    b. Configure the transport type
4. Configure the HTTP plug-in for OSE Remote
5. Start the servers
6. Test the servers

5.1 Overview of the configuration

A Web server, an application server and a database server are installed in 3 physically separate machines, Machine A, Machine B, and Machine C, respectively. All the requests from Web browsers received by the Web server in Machine A are redirected by the Web server plug-in, which is a component of WebSphere, to the application server in Machine B for processing. Machine B requires an Administrative Server to manage its resources. The Administrative Server uses a repository located on Machine C.

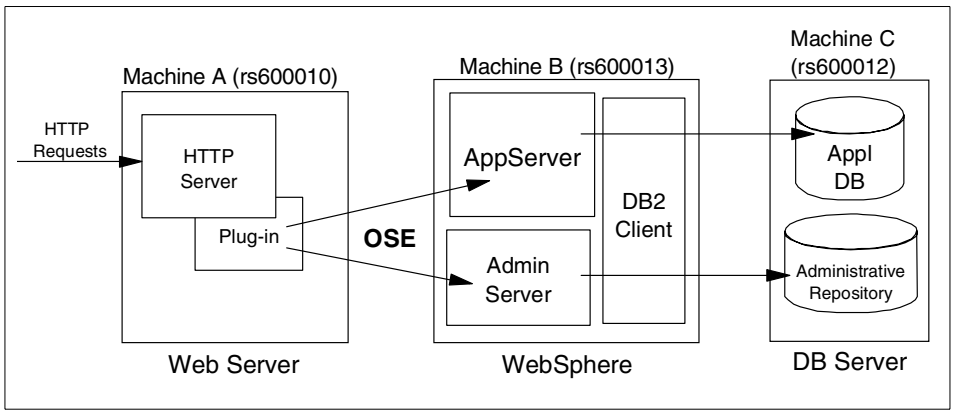

Figure 57. OSE Remote

5.2 Installation summary

Here are the instructions for setting up OSE Remote in a single WAS environment.

Table 4 is a summary of the software installation steps. We will discuss only steps 7 and 8 in this chapter. For steps 1 to 6, please refer to the *Getting Started* documentation which is part of the WebSphere product.

Table 4. Software installation steps

Step No.	Machine A (HTTP)	Machine B (WAS)	Machine C (DB)
1			Install DB2 UDB Apply appropriate FixPack

Step No.	Machine A (HTTP)	Machine B (WAS)	Machine C (DB)
2			Create DB
3		Install DB2 client Apply appropriate FixPack	
4		Catalog node Catalog db	
5	Install JDK (WebSphere 3.02x)	Install JDK (WebSphere 3.02x)	
6	Install Web server		
7	Install WebSphere (Production Application Server; -Plugins, -Core Server)		
8		Install WebSphere (Production Application Server)	

5.2.1 Install WebSphere on the Web server machine

On Machine A, you must install the Web server (installation Step 6). Make sure it is not running before installing WebSphere.

Then, you need to install the WebSphere Application Server and the Web server plug-in (installation Step 7).

To install, select **Custom Installation** and then select **Production Application Server** (**Core Server** and appropriate plug-in for your Web server). If you need to install another component, select it. For the production environment, you should not need to select anything other than Production Application Server and plug-in.

5.2.2 Install WebSphere on the application server machine

On Machine B, where the application server resides, a DB2 client is required (installation Step 3). See the instructions for configuring this in Appendix I, “Accessing remote DB2 UDB databases” on page 553 and the product documentation.

Then, you need to install the WebSphere Administrative Server and WebSphere Production Application Server (installation Step 8).

To install, select **Custom Installation** and then select **Production Application Server** with **Core Server**. We selected several components other than Production Application Server. But for the production environment, you do not have to select anything other than **Production Application Server** (**Core Server**).

5.3 Configure the application server

Before you start configuring the plug-in for OSE Remote, you need to start all software which you need:

1. Start a DB2 instance for the Administrative Repository on Machine C (in our case, rs600012) if it is not started.
2. Start Administrative Server on Machine B (in our case, rs600013).

   In our example, we specified “install.initial.config=true” in the admin.config file to install the Default Server.
3. Start Administrative Console on Machine B (in our case, rs600013).

   Note: You can use any machine for your Administrative Console.
4. When the console finishes initializing, ensure Machine B (in our case, rs600013) appears in the Topology page as shown in Figure 58 on page 99.

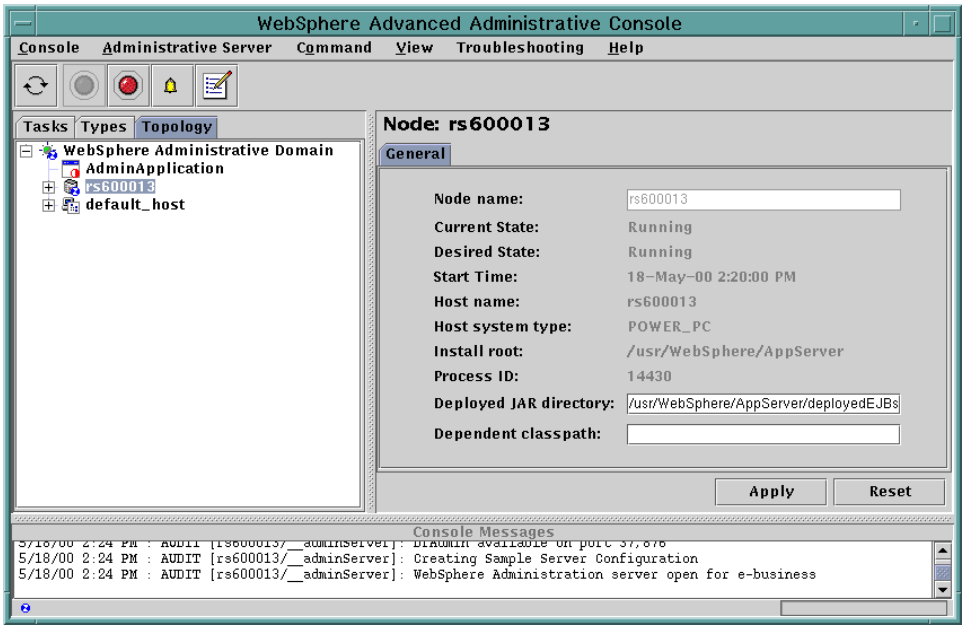

Figure 58. WebSphere Advanced Administrative Console

5.3.1 Add host alias

Add aliases for Machine A. These aliases are needed so that the virtual host will accept requests redirected from Machine A with Machine A's URI instead of Machine B's URI. You may add Machine A's hostname (rs600010), IP address, and fully qualified name, for example, "rs600010","9.24.104.119" and "rs600010.itso.ral.ibm.com".

To add the host aliases from the Administrative Console:

1. Click the **Topology** tab.
2. Click the **default_host** in the Topology tree.
3. Click the **Advanced** tab. Add aliases under **Host Aliases** and click the **Apply** button as shown in Figure 59 on page 100.

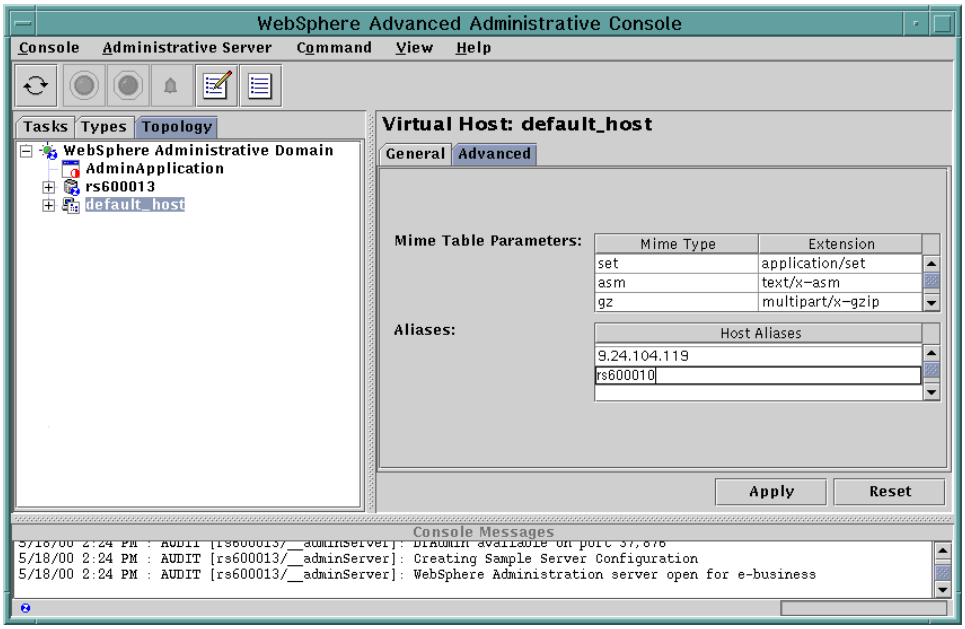

Figure 59. Add aliases

When the apply completes successfully, the host aliases will have been updated. You will need to restart all application servers using this virtual host to pick up your changes if they have already been started.

5.3.2 Configure the transport type

The servlet engine needs to be configured to use sockets instead of local pipes. You need to specify INET sockets for the transport type of OSE Queue. This is the default on Solaris.

1. From the **Topology** tab, select the servlet engine (servletEngine in our example) of the application server (Default Server in our example) under Machine B (rs600013) as shown in Figure 60 on page 101.

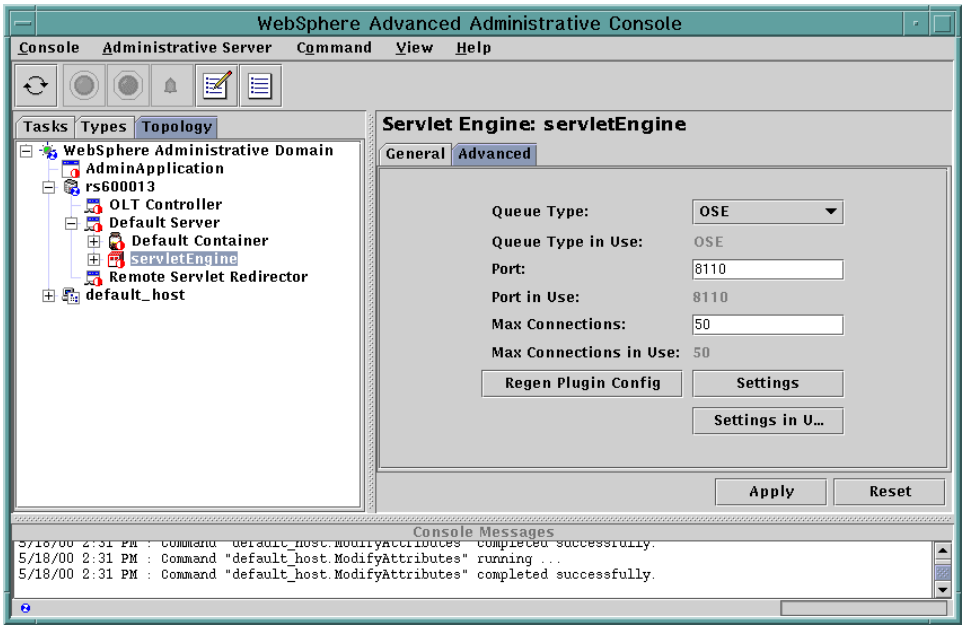

Figure 60. Transport type settings

2. Select the **Advanced** tab inside the Servlet Engine panel, choose **OSE** for the Queue Type and click **Settings**.
3. You will get the Edit Servlet Engine Transport window. Then choose `INET Sockets` for **Transport Type** as shown in Figure 61.

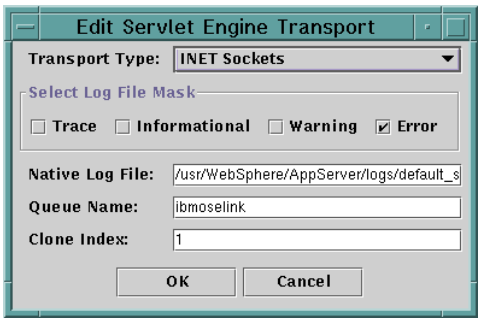

Figure 61. Transport Type: INET Sockets

4. Then click the **OK** button. You will be returned to the Administrative Console (Figure 60).
5. Click the **Apply** button. The servlet engine is now configured.

Note

Any changes to the application server such as adding a Web application, servlet, or EJB, should be made now. If you plan to create clones, do so now. See Chapter 4, “Application server clones” on page 77 for more information. Any changes you make after you generate the plug-in files require you to re-create those files by repeating the steps in 5.5, “Configure the plug-in for OSE Remote” on page 103.

After the plug-in files have been regenerated, the HTTP server must be restarted.

5.4 Start the application server

Start the application server on Machine B (rs600013). The WebSphere Application Server can be started by clicking **Default Server** under the Topology tab and selecting **Start.**

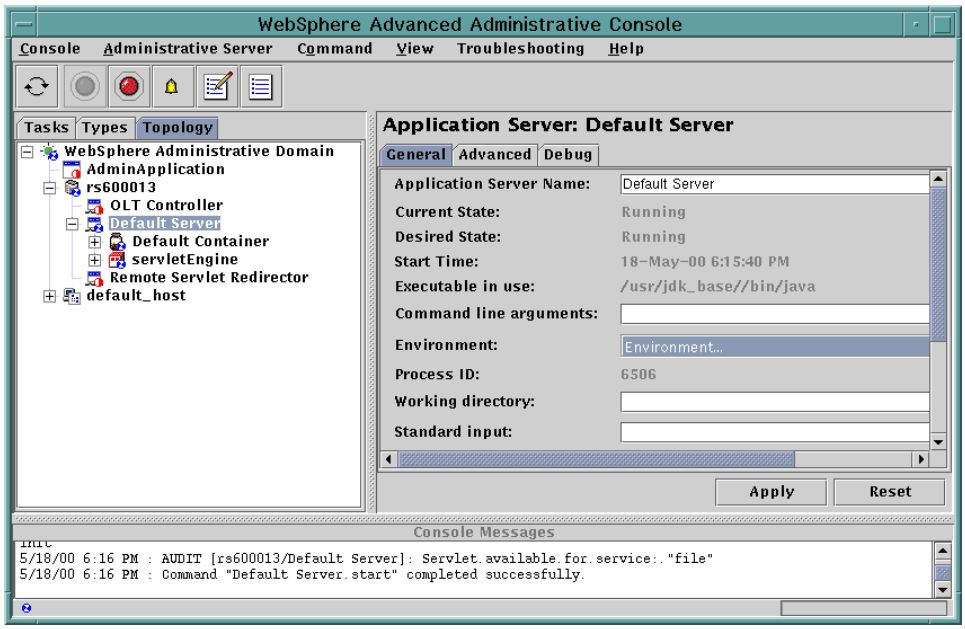

Figure 62. Application server started

If you have clones, make sure you start them from the model.

You must wait for the product to refresh the automatically generated or updated plug-in configuration files which reside in the <was_dir>/temp directory. This could take up to a few minutes.

Note that you can specify another directory in which to store the properties files. See 10.11.4, “Variation 4: OSE Remote and local” on page 305 for more information.

5.5 Configure the plug-in for OSE Remote

The WebSphere plug-in for the HTTP server uses three properties files generated by WebSphere to define its configuration:

1. queues.properties
2. rules.properties
3. vhosts.properties

These files need to be generated on the machine that will be used as the HTTP server.

The plug-in can be configured two ways:

1. Configure the plug-in manually.

   Manual configuration requires that the files be moved (copied, FTPed and so on) from the WebSphere machine to the HTTP server machine.

   Some changes to the copied files are required.

2. Configure the plug-in using a script.

   The script uses IIOP to connect to the Administrative Server to get the configuration information. This requires that IIOP traffic be allowed through any firewall that is between the HTTP server and WebSphere machines.

   Some changes to the generated files are required when using the script.

5.5.1 Configure the plug-in manually

1. Copy the following files from <was_dir>/temp on Machine B (WebSphere) to the same directory on Machine A (HTTP server):

   a. vhosts.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu May 18 18:17:16 EDT 2000
9.24.104.109=default_host
rs600010.itso.ral.ibm.com=default_host
rs600010=default_host
localhost=default_host
127.0.0.1=default_host
9.24.104.119=default_host
rs600013.itso.ral.ibm.com=default_host
rs600013=default_host
```

Figure 63. <was_dir>/temp/vhosts.properties

b. rules.properties

```
#IBM WebSphere Plugin URL Mapping Rules
#Thu May 18 18:17:16 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 64. <was_dir>/temp/rules.properties

c. queues.properties

```
#IBM WebSphere Plugin Communication Queues
#Thu May 18 18:17:16 EDT 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 65. <was_dir>/temp/queues.properties

2. After copying the queues.properties file from Machine B to Machine A, edit the queues.properties file on Machine A and add an entry for the host where the application server (clone) resides. This will be the name of the machine that is running the application server.

   The form of the entry is:

   ```
   ose.srvgrp.<queue>.<clone>.host=system
   ```

   In our example, the `queue` is `ibmoselink`, the `clone` is `clone1` and the `system` is `rs600013` (the IP address of rs600013 could have been used instead). The resulting line is shown in Figure 66.

```
#IBM WebSphere Plugin Communication Queues
#Thu May 18 18:17:16 EDT 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.host=rs600013
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 66. <was_dir>/temp/queues.properties for OSE Remote

The files are now ready for use.

5.5.2 Configure the plug-in using a script

We provide a sample shell script (OSERemoteConfig.sh) for UNIX. See Figure 67 on page 108. It sets all needed environment variables and then runs the OSE Remote Plugin Config program. Place the file in the <was_dir>/bin directory on the HTTP server machine. It uses the

Administrative Server on Machine B (WebSphere machine) to build the temporary files needed by the plug-in.

You need to edit OSERemoteConfig.sh and change the following:

1. Change the `-adminNodeName` parameter to point to the WebSphere node on Machine B that will be used as the Administrative Server (in our example, the Administrative Server is on the node named rs600013).
2. Change the `-nameServiceNodeName` parameter to point to the hostname of Machine B (in our example, rs600013 is the hostname of the machine where the WebSphere node rs600013 is located).

After you create and update the shell script (or batch file for Windows NT), you need to run it to generate the plug-in files.

For example:

```
/usr/WebSphere/AppServer/bin/OSERemoteConfig.sh
```

When the script has finished running, the temporary files should have been created for the plug-in. These files will reside in the subdirectory <was_dir>/temp on the HTTP server (Machine A). The files are queues.properties, rules.properties, and vhosts.properties. The script requires the `com.ibm.servlet.engine.oselistener.systemsmgmt.OSERemotePluginCfg` class. This is available in WebSphere Version 3.021.

CORBA Listener Port for the Administrative Server

If you have a firewall between the HTTP server and WebSphere node, you might want to configure the CORBA Listener Port for the Administrative Server which is running on the WebSphere node. To to so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the line `com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs` in the <was_dir>/bin/admin.config file.

For example:

```
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
-Dcom.ibm.CORBA.ListenerPort=33000
```

The line has been divided with line breaks for easier reading. In your actual admin.config file, ensure that the line is on a single line. In other words, combine all of the lines from “com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs” to “ListenerPort=33000” on one line, each separated by a space. You have to do this first and restart the Administrative Server before configuring OSE remote with the shell script. Manual configuration does not require this.

```
#!/bin/ksh
. $PWD/setupCmdLine.sh
# setup the classpath
WAS_CP=$WAS_CP:$WAS_HOME/lib/ibmwebas.jar
WAS_CP=$WAS_CP:$WAS_HOME/properties
WAS_CP=$WAS_CP:$WAS_HOME/lib/servlet.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/webtlsrn.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/lotusxsl.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/ns.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/ejs.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/ujc.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/repository.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/admin.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/swingall.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/console.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/tasks.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/xml4j.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/x509v1.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/vaprt.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/iioprt.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/iioptools.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/dertrjrt.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/sslight.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/ibmjndi.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/deployTool.jar
WAS_CP=$WAS_CP:$WAS_HOME/lib/databeans.jar
WAS_CP=$WAS_CP:$WAS_HOME/classes
WAS_CP=$WAS_CP:$JAVA_HOME/lib/classes.zip
export CLASSPATH=$WAS_CP

$JAVA_HOME/bin/java
com.ibm.servlet.engine.oselistener.systemsmgmt.OSERemotePluginCfg
-traceString com.ibm.servlet.engine.*=all=enabled -traceFile
$WAS_HOME/logs/configRegen.log -serverRoot $WAS_HOME -adminNodeName
rs600013 -nameServiceNodeName rs600013 -nameServicePort 900
```

Figure 67. OSERemoteConfig.sh

Note

WebSphere 3.5 provides a similar shell script (OSERemoteConfig.sh) for UNIX and a batch file (OSERemoteConfig.bat) for Windows. Therefore, you should use these instead of creating them yourself.

5.6 Configure bootstrap.properties for security

You need to modify the bootstrap.properties file on both Machines A and B, if you use WebSphere security for the resources on the Web Server.

If you use WebSphere security for only the resources on the WebSphere node, you can skip this step.

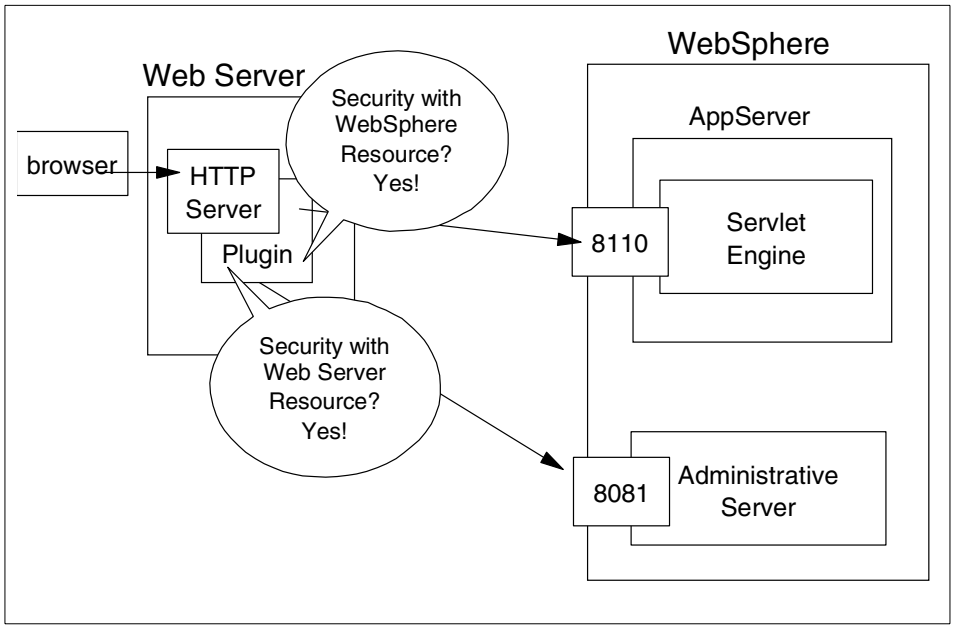

Figure 68. OSE Remote with WebSphere security

5.6.1 Editing bootstrap.properties on the application machine

Edit the <was_dir>/properties/bootstrap.properties file on the application server machine (rs600013). Set the property:

```
ose.srvgrp.ibmappserve.clone1.type=remote
```

Based on your requirements, change the clone port property as needed:

```
ose.srvgrp.ibmappserve.clone1.port=unused port number
```

If you specify or change the port, ensure that the port is the same in the bootstrap.properties file on the HTTP server machine.

```
##
# Set the specific native web-server plugins security setting
# values of ose.security.enabled may be a combination of
# true|false
ose.security.enabled=true
################################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=local
ose.srvgrp.ibmappserve.clone1.host=localhost
ose.mode=out
```

Figure 69. <was_dir>/properties/bootstrap.properties (original) on Machine B (WebSphere)

```
##
# Set the specific native web-server plugins security setting
# values of ose.security.enabled may be a combination of
# true|false
ose.security.enabled=true
################################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=remote
ose.srvgrp.ibmappserve.clone1.host=localhost
ose.mode=out
```

Figure 70. <was_dir>/properties/bootstrap.properties (after modification) on Machine B

5.6.2 Editing bootstrap.properties on the HTTP server machine

Edit the <was_dir>/properties/bootstrap.properties file on the HTTP server machine (rs600010). Set the following properties:

```
ose.srvgrp.ibmappserve.clone1.port=8081
```

Make sure this is the same as the entry in bootstrap.properties on the WebSphere machine.

```
ose.srvgrp.ibmappserve.clone1.type=remote

ose.srvgrp.ibmappserve.clone1.host=<admin server hostname>
```

```
##
# Set the specific native web-server plugins security setting
# values of ose.security.enabled may be a combination of
# true|false
ose.security.enabled=true
#############################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=local
ose.srvgrp.ibmappserve.clone1.host=localhost
ose.mode=out
```

Figure 71. <was_dir>/bootstrap.properties (original) on Machine A (HTTP server)

```
##
# Set the specific native web-server plugins security setting
# values of ose.security.enabled may be a combination of
# true|false
ose.security.enabled=true
################################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=remote
ose.srvgrp.ibmappserve.clone1.host=rs600013
ose.mode=out
```

Figure 72. <was_dir>/bootstrap.properties (after modification) on Machine A

5.6.3 Verify the rules.properties

If you use WebSphere security for the Web server resources, you should verify the <was_dir>/temp/rules.properties on the HTTP server node. After you configure WebSphere security for your Web server, an “A” is added to the URI in the properties file.

In our example, we configured WebSphere security for both the /servlet/snoop servlet and /index.html HTML file as shown in Figure 73 on page 113. As you can see, there is no transport queue specified and an “A” is added to the URI of index.html. Therefore, when you access the HTTP server with URI /index.html, the plug-in contacts the Administrative Server (port 8081 by default) on the WebSphere node to verify its security information as shown in Figure 74 on page 114.

But, there is a transport queue specified and no “A” added to the URI /servlet/snoop. Because of this, when you access the HTTP server with URI /servlet/snoop, the plug-in accesses the servlet engine instead of Administrative Server as shown in Figure 75 on page 114.

Note that this information is applicable to WebSphere V3.021 with security patches.

```
#IBM WebSphere Plugin URL Mapping Rules
#Wed May 24 14:01:30 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/index.html=,A
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 73. <was_dir>/temp/rules.properties with WebSphere security on the HTTP server

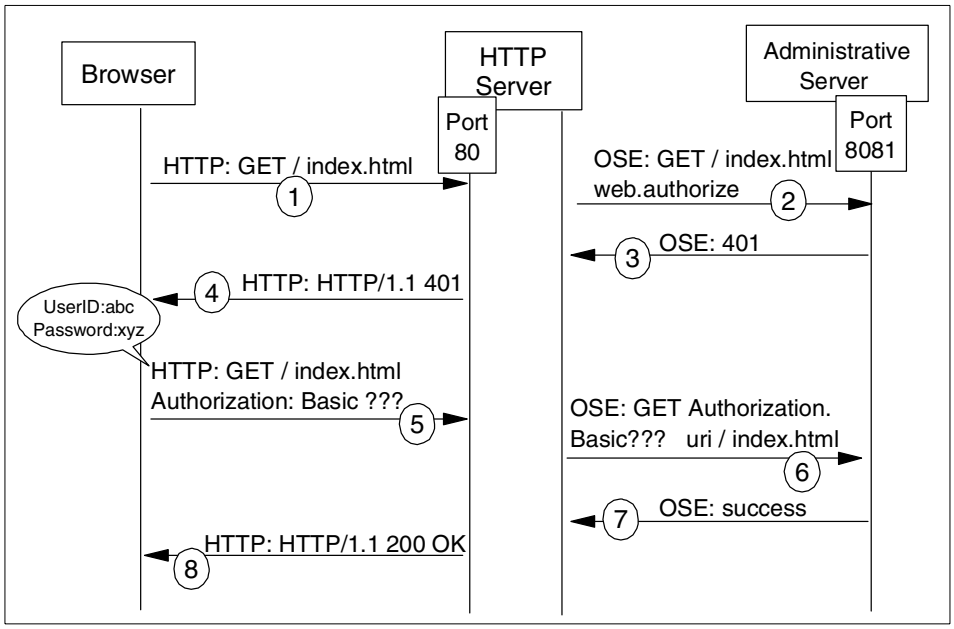

Figure 74. WebSphere security for resources on Web server node with OSE Remote

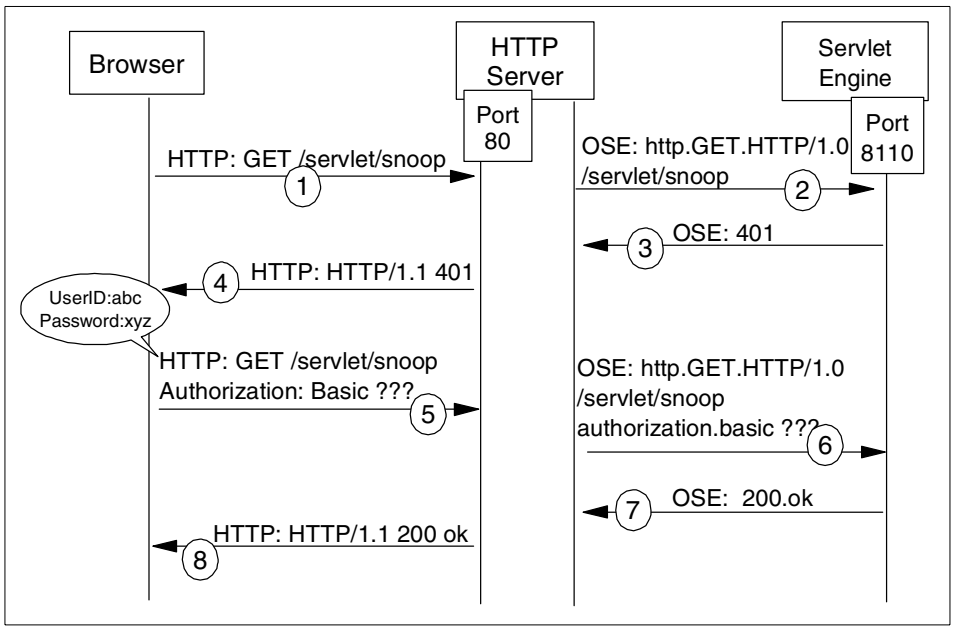

Figure 75. WebSphere security for resources on WebSphere node with OSE Remote

5.6.4 High availability configuration for WebSphere security

If you have multiple WebSphere nodes (we will discuss several topologies later in this chapter), you should configure bootstrap.properties for the Administrative Servers' high availability.

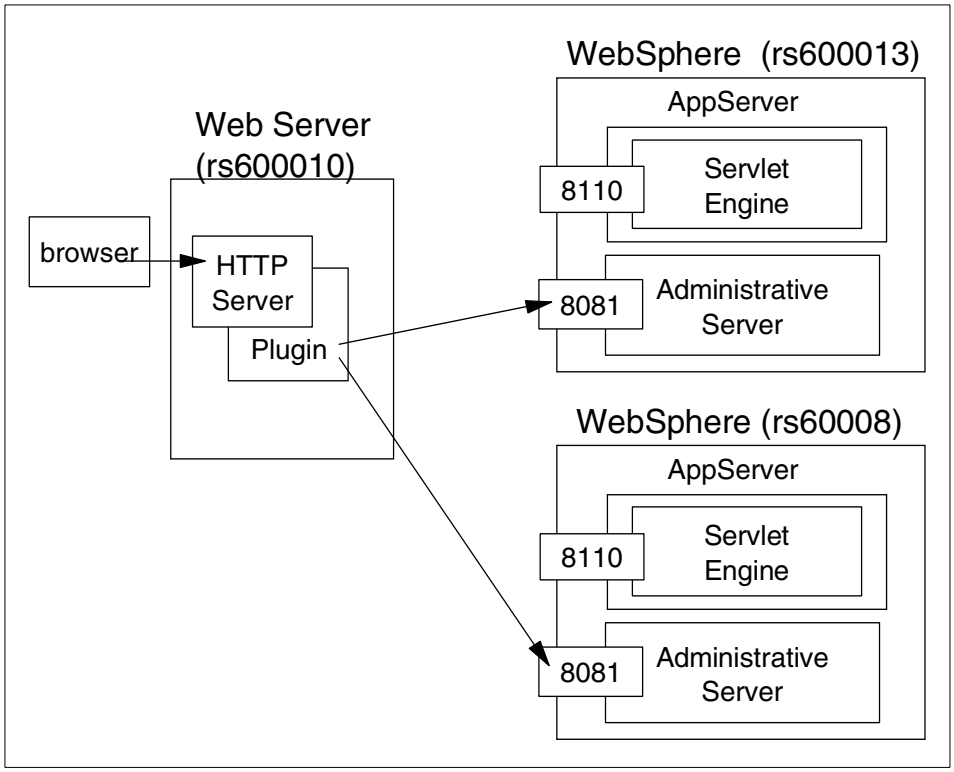

Figure 76. High availability configuration for WebSphere security with OSE Remote

You should configure <was_dir>/properties/boostrap.properties on all WebSphere Administrative Server nodes as described 5.6.1, "Editing bootstrap.properties on the application machine" on page 109.

Then, you need to configure <was_dir>/properties/boostrap.properties on the HTTP server node for the Administrative Servers' high availability. Here is a sample configuration.

```
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=2
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=remote
ose.srvgrp.ibmappserve.clone1.host=rs600013
ose.srvgrp.ibmappserve.clone2.port=8081
ose.srvgrp.ibmappserve.clone2.type=remote
ose.srvgrp.ibmappserve.clone2.host=rs60008
ose.mode=out
```

Figure 77. bootstrap.properties on Machine A for Administrative Servers' high availability

The idea is the same as cloning application servers which we discussed in Chapter 4, "Application server clones" on page 77. We need to specify the number of Administrative Servers (clonescount), queue name, port number which the Administrative Server is listening on and the hostname of the Administrative Server in <was_dir>/properties/boostrap.properties instead of <was_dir>/temp/queues.properties.

5.7 Re-start the servers

Now, you need to re-start the servers.

1. Re-start the Administrative Server

   Since you have changed the bootstrap.properties file for WebSphere security as described in 5.6, "Configure bootstrap.properties for security" on page 109, you need to re-start the Administrative Server on Machine B (rs600013). If you didn't update the bootstrap.properties, you don't need to re-start the Administrative Server.

2. Re-start the application server

   Re-start the application server on Machine B (rs600013). The application server can be started by clicking **Default Server** in the Topology tab and selecting **Start**.

3. Re-start the Web server

   Re-start the Web server on Machine A (rs600010).

5.8 Test the server

Now that the HTTP server and WebSphere are running, we need to test the servers to make sure that everything is configured properly. Using a Web browser, access the following URI (rs600010 is the HTTP server and /webapp/examples/showCfg is the servlet):

Access the following URI:

```
http://rs600010/webapp/examples/showCfg
```

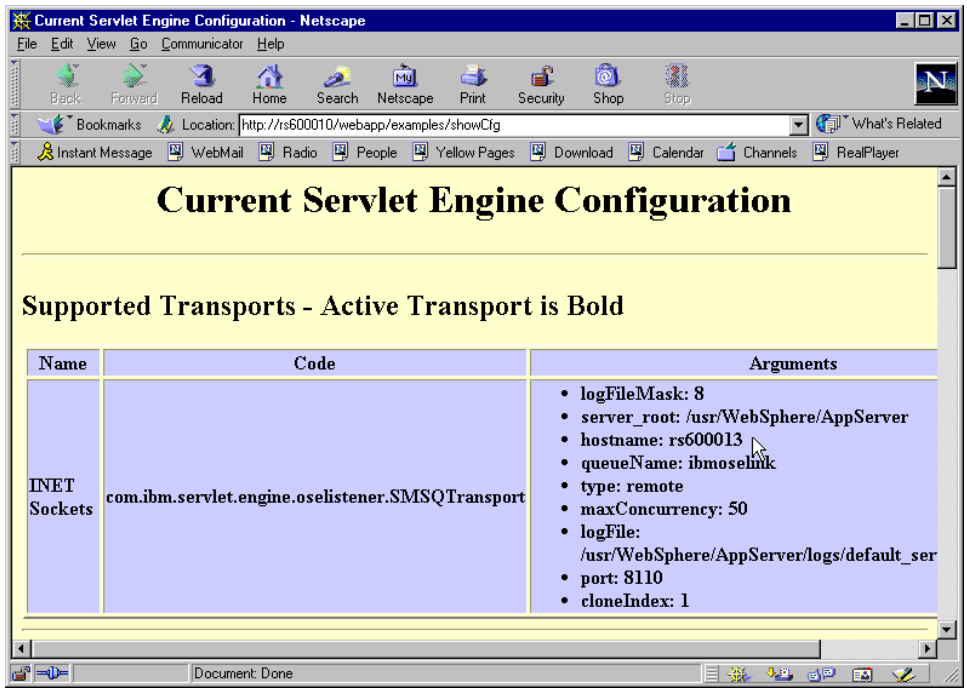

Figure 78. Results in browser

Notice that the hostname (rs600013) is the name of the machine running WebSphere and not the one that is the HTTP Server (rs600010).

Note

You need to repeat the steps in 5.5, “Configure the plug-in for OSE Remote” on page 103 after performing any of the following activities:

- Add/remove a URL (Web resource) to the environment.
- Secure/unsecure a URI (when you add security to a URI, an “A” is added to it, changing its name).
- Add/remove a new host alias.
- Change the queue properties of a servlet engine (name, port).
- Add/remove a servlet engine.
- Add/remove a clone.

5.9 Related topologies

OSE Remote can be used for both vertical and horizontal scaling of the WebSphere environment.

5.9.1 Variation 1: horizontal scaling

A Web server, application servers and a database server are installed in four physically separate machines, Machine A, Machine B, Machine C and Machine D, respectively. All the requests from Web browsers received by the Web server in Machine A are redirected by a Web server plug-in, which is a component of WebSphere, to the application server in Machine B or C for processing. Machines B and C require an Administrative Server to manage their resources.

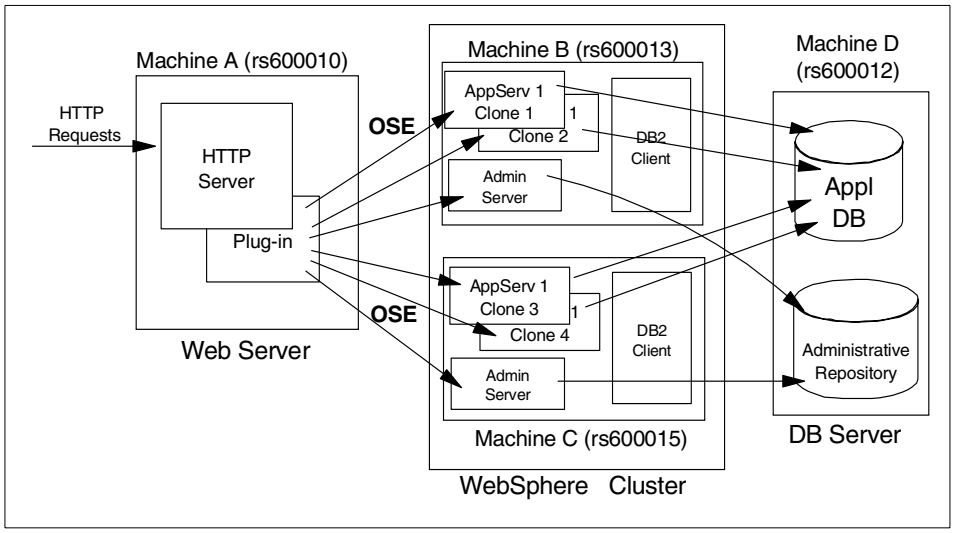

Figure 79. Remote OSE with cloned servers

This configuration can be achieved by creating the WebSphere cluster on Machine B and Machine C. To configure WebSphere clustering, Machines B and C need to share the Administrative Repository.

Then create clones on physically separate machines as described in 10.8, “Creating the clones” on page 286.

Clones of the Default Server on separated machines

For V3.5, the default configuration automatically creates and installs the JDBC Driver (Admin DB Driver).

When creating a clone of the Default Server to a remote node, it requires the Admin DB Driver to be installed on the remote node prior to creating the clone.

To install the Admin DB Driver, right click the Admin DB Driver and select **Install**. The Install Driver window will appear. Then select the remote node and specify the jar file which contains the JDBC Driver. Then click **Install**. After you install the JDBC Driver for the Admin DB Driver on the remote node, you can create clones of the Default Server.

After configuring the cluster, generate the plug-in files on Machine A using the method described in 5.5, “Configure the plug-in for OSE Remote” on page

103 and updating bootstrap.properties on Machines A and B as described in 5.6, “Configure bootstrap.properties for security” on page 109.

If you use the steps shown in 5.5.2, “Configure the plug-in using a script” on page 105, all the nodes will not be listed in the queues.properties files after the files have been generated. These will need to added manually. For each node that is not listed, add the following properties:

1. ose.srvgrp.<queue>.<clone>.host
2. ose.srvgrp.<queue>.<clone>.type
3. ose.srvgrp.<queue>.<clone>.port

The port the clone uses can be found by expanding the clone in the Topology tab, selecting the servlet engine, and then clicking the **Advanced** tab.

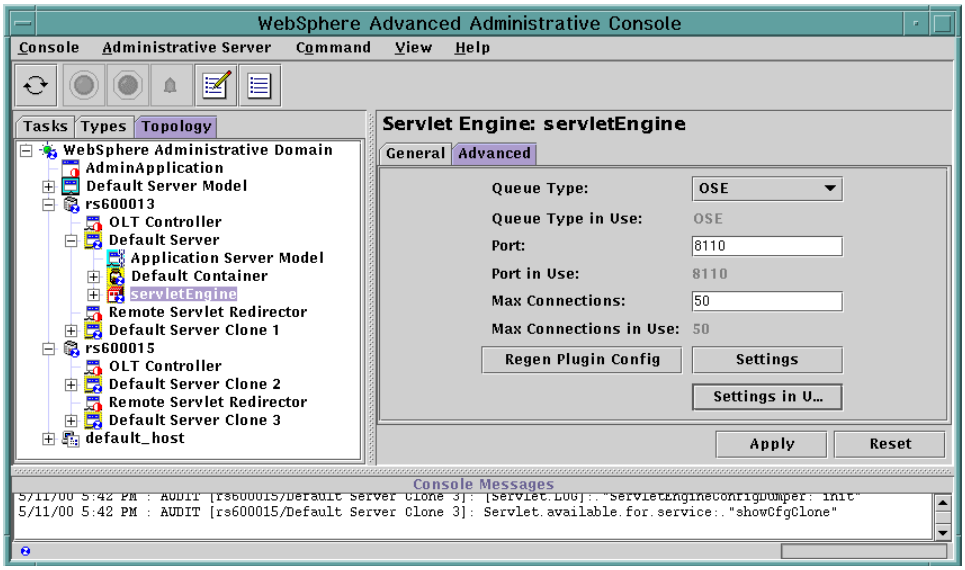

Figure 80. Servlet engine port number for Default Server on Machine B

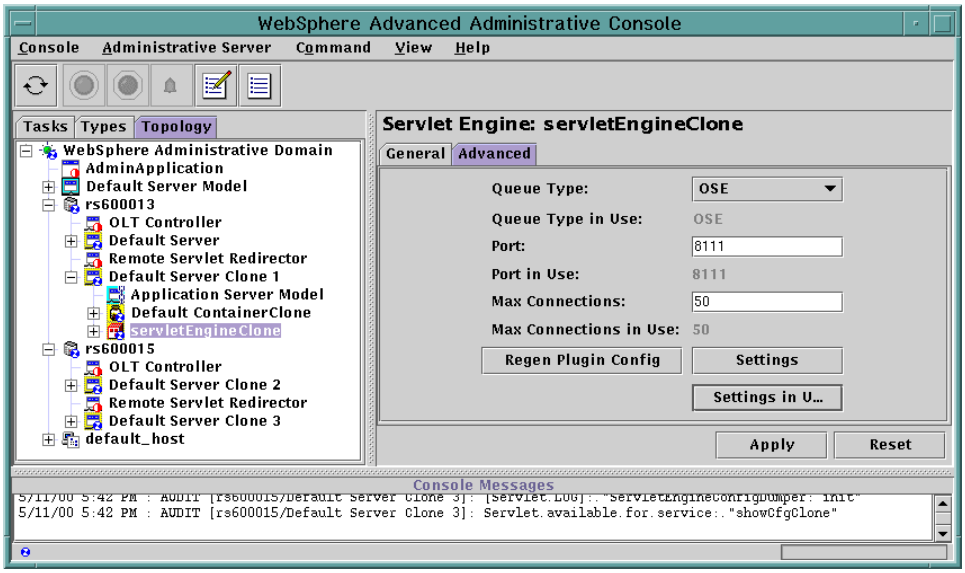

Figure 81. Servlet engine port number for Default Server Clone 1 on Machine B

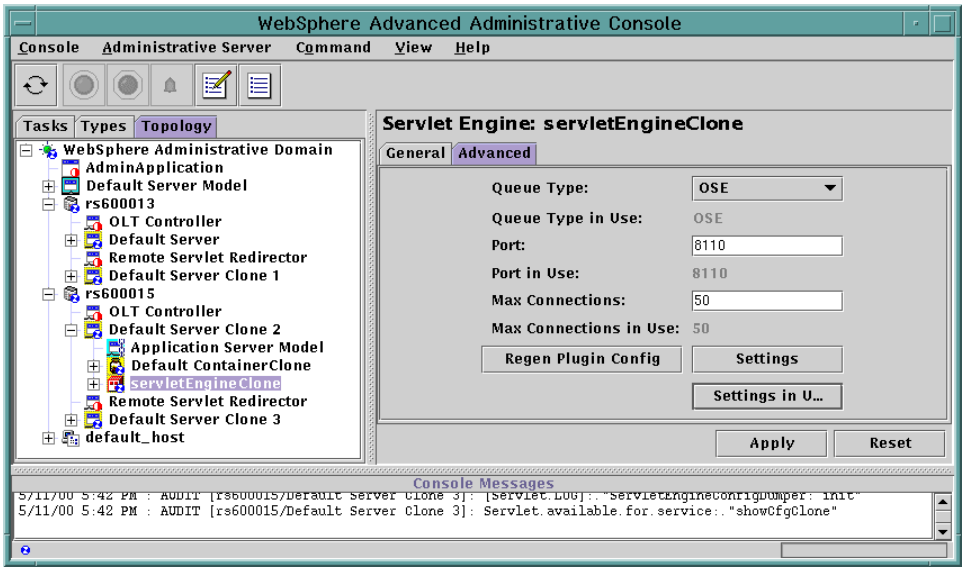

Figure 82. Servlet engine port number for Default Server Clone 2 on Machine C

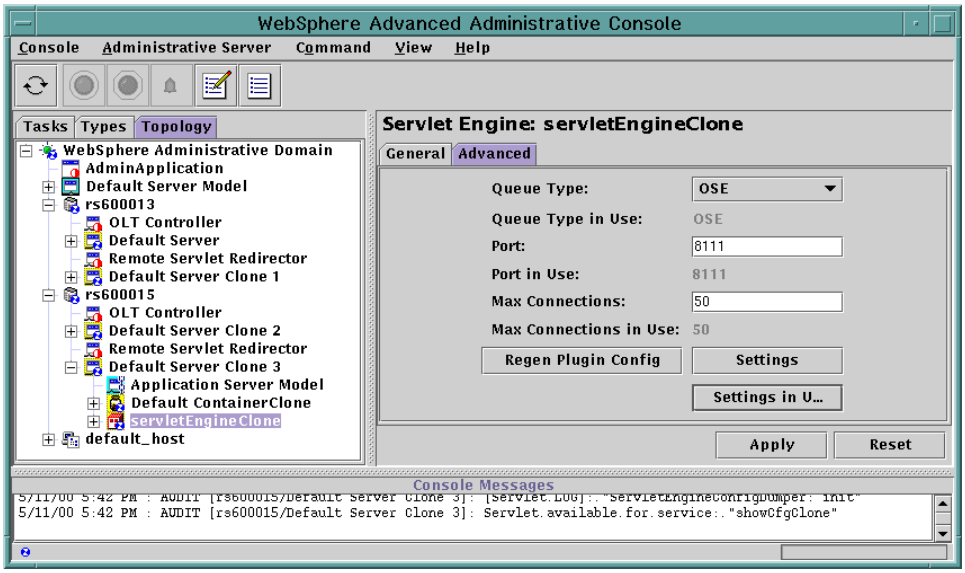

Figure 83. Servlet engine port number for Default Server Clone 3 on Machine C

The files you create look similar to the ones shown below.

1. queues.properties

Here is the original queues.properties file on Machine B (rs600013).

```
#IBM WebSphere Plugin Communication Queues
#Thu May 11 17:27:05 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone2.port=8111
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 84. <was_dir>/temp/queues.properties on Machine B (rs600013): original

This is the original queues.properties file on Machine C (rs600015).

```
#IBM WebSphere Plugin Communication Queues
#Thu May 11 17:30:28 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone2.port=8111
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 85. <was_dir>/temp/queues.properties on Machine C (rs600015)

Here is the queues.properties file after we modified it on Machine A (rs600010).

```
#IBM WebSphere Plugin Communication Queues
#Wed May 10 17:14:57 EDT 2000
ose.srvgrp.ibmoselink.clonescount=4
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone2.port=8111
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone2.host=rs600013
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone1.host=rs600013
ose.srvgrp.ibmoselink.clone4.port=8111
ose.srvgrp.ibmoselink.clone4.type=remote
ose.srvgrp.ibmoselink.clone4.host=rs600015
ose.srvgrp.ibmoselink.clone3.port=8110
ose.srvgrp.ibmoselink.clone3.type=remote
ose.srvgrp.ibmoselink.clone3.host=rs600015
```

Figure 86. <was_dir>/temp/queues.properties on Machine A: after we modified

2. rules.properties

```
#IBM WebSphere Plugin URL Mapping Rules
#Wed May 10 17:14:57 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 87. <was_dir>/temp/rules.properties

3. vhosts.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Wed May 10 17:14:57 EDT 2000
9.24.104.45=default_host
9.24.104.109=default_host
localhost=default_host
rs600015=default_host
rs600013=default_host
rs600010=default_host
127.0.0.1=default_host
9.24.104.119=default_host
rs600013.itso.ral.ibm.com=default_host
rs600010.itso.ral.ibm.com=default_host
rs600015.itso.ral.ibm.com=default_host
```

Figure 88. <was_dir>/temp/vhosts.properties

Note

When you change the properties of resources hosted by the application server processes, you need to get the latest properties files and deploy them with proper modification.

Note

You need to ensure that the queues.properties file on the HTTP server points to all the available servlet engines:

- Make sure all the clones are represented.
- Make sure that the type of the clones is set to remote.
- Make sure that the appropriate port number is set for each clone.
- Make sure that the host entry is set for each clone.
- Make sure that the clones count is equal to the total number of clones.

5.9.2 Variation 2: distinct Web applications

A machine acting as the HTTP server (Machine D) will use OSE Remote to route requests between two nodes (Machine A and Machine B). Each node will be running a distinct application server (assuming this is non-cloned). The nodes will share a repository stored on Machine C.

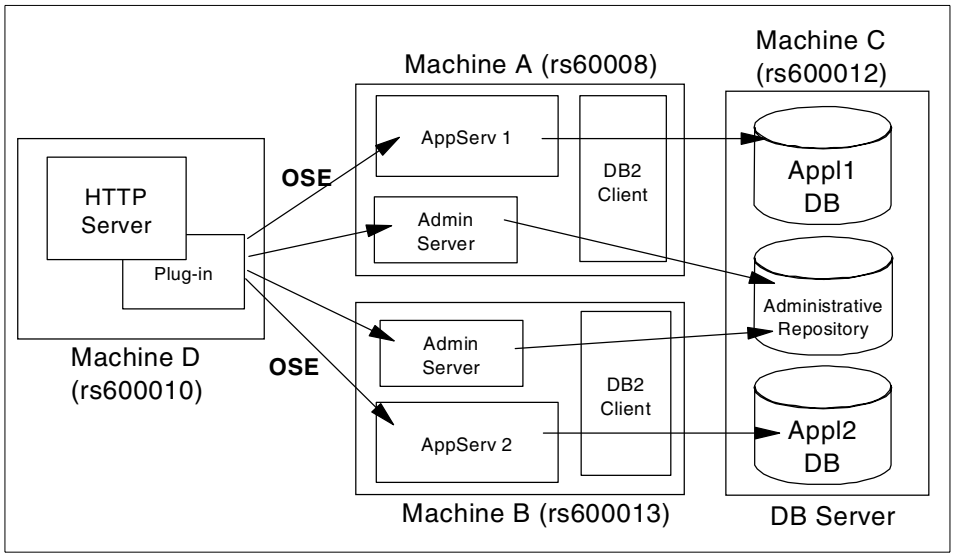

Figure 89. OSE Remote with distinct application servers

This configuration can be achieved by creating the WebSphere cluster of Machine A and Machine B.

To configure WebSphere clustering, Machine A and Machine B need to share the Administrative Repository.

Then configure an application server on each node.

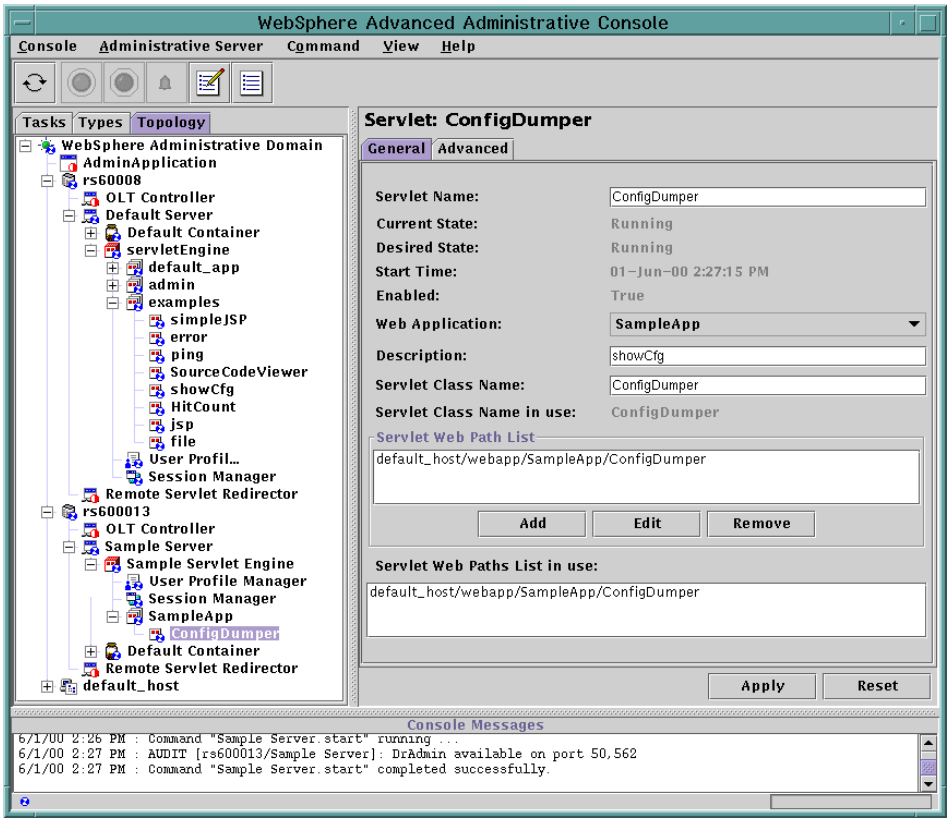

Figure 90. Configuring an application server on each node

After configuring the cluster and application servers, generate the plug-in files on Machine A using the method described in 5.5, “Configure the plug-in for OSE Remote” on page 103 and update the bootstrap.properties file on Machines A and D as described in 5.6, “Configure bootstrap.properties for security” on page 109.

The properties files you create should look similar to the ones shown in Figure 91, Figure 92, and Figure 93.

In our example, we used Default Server as application server on Machine A and created Sample Server as application server on Machine B. In the Sample Server, we created a sample servlet, ConfigDumper. Each application server has a unique queue name. As you can see in Figure 91, “ibmoselink1” is specified as URI of the ConfigDumper as its queue name. Another URI uses the queue name, “ibmoselink.”

```
#IBM WebSphere Plugin URL Mapping Rules
#Thu Jun 01 14:39:04 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
default_host/webapp/SampleApp/ConfigDumper=ibmoselink1
```

Figure 91. <was_dir>/temp/rules.properties on Machine D (rs600010)

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu Jun 01 14:39:04 EDT 2000
rs60008.itso.ral.ibm.com=default_host
9.24.104.109=default_host
9.24.104.30=default_host
localhost=default_host
rs600013=default_host
rs600010=default_host
127.0.0.1=default_host
9.24.104.119=default_host
rs60008=default_host
rs600010.itso.ral.ibm.com=default_host
rs600013.itso.ral.ibm.com=default_host
```

Figure 92. <was_dir>/temp/vhosts.properties on Machine D (rs600010)

In our example, we used two distinct application servers in one domain. Therefore, in the queues.properties file, ose.srvgrp entry has two queue names, ibmoselink and ibmoselink1. As described above, the queue ibmoselink is for the Default Server and ibmoselink1 is for the Sample Server. Neither server has any clones. Therefore, `clonescount` entry for both `ibmoselink` and `ibmoselink1` queues is `1`. Default Server is running on Machine A (rs60008). Therefore, host entry of ibmoselink queues is specified as rs60008 and the host entry of `ibmoselink1` queues is specified as rs600013 (Machine B). Default Server has a servlet engine which is listening on port 8110 for INET sockets and the servlet engine of Sample Server is listening on port 8993.

```
#IBM WebSphere Plugin Communication Queues
#Thu Jun 01 14:39:04 EDT 2000
ose.srvgrp=ibmoselink,ibmoselink1
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone1.host=rs60008
ose.srvgrp.ibmoselink1.clonescount=1
ose.srvgrp.ibmoselink1.clone1.port=8993
ose.srvgrp.ibmoselink1.clone1.type=remote
ose.srvgrp.ibmoselink1.clone1.host=rs600013
ose.srvgrp.ibmoselink1.type=FASTLINK
```

Figure 93. <was_dir>/temp/queues.properties on Machine D (rs600010)

Note that each application server Queue Name has to be a unique queue name. You can verify the queue name from Administrative Console, **Topology -> Application Server -> Servlet Engine -> Advanced -> Settings -> Edit Servlet Engine Transport** as shown in Figure 94.

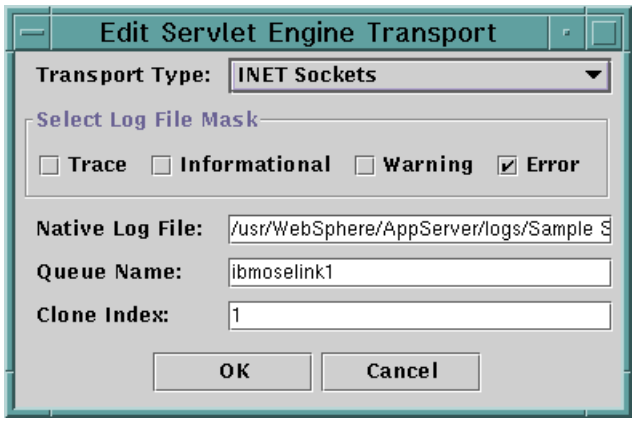

Figure 94. Verify the queue name

The following files are the original files on Machines A and B.

```
#IBM WebSphere Plugin Communication Queues
#Thu Jun 01 15:54:24 EDT 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 95. <was_dir>/temp/queues.properties on Machine A (rs60008)

```
#IBM WebSphere Plugin URL Mapping Rules
#Thu Jun 01 15:54:24 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 96. <was_dir>/temp/rules.properties on Machine A (rs60008)

```
#IBM WebSphere Plugin Communication Queues
#Thu Jun 01 15:51:54 EDT 2000
ose.srvgrp.ibmoselink1.clonescount=1
ose.srvgrp.ibmoselink1.type=FASTLINK
ose.srvgrp=ibmoselink1
ose.srvgrp.ibmoselink1.clone1.port=8993
ose.srvgrp.ibmoselink1.clone1.type=remote
```

Figure 97. <was_dir>/temp/queues.properties on Machine B (rs600013)

```
IBM WebSphere Plugin URL Mapping Rules
#Thu Jun 01 15:51:54 EDT 2000
default_host/webapp/SampleApp/ConfigDumper=ibmoselink1
```

Figure 98. <was_dir>/temp/rules.properties on Machine B (rs600013)

The following are the sample results of this topology.

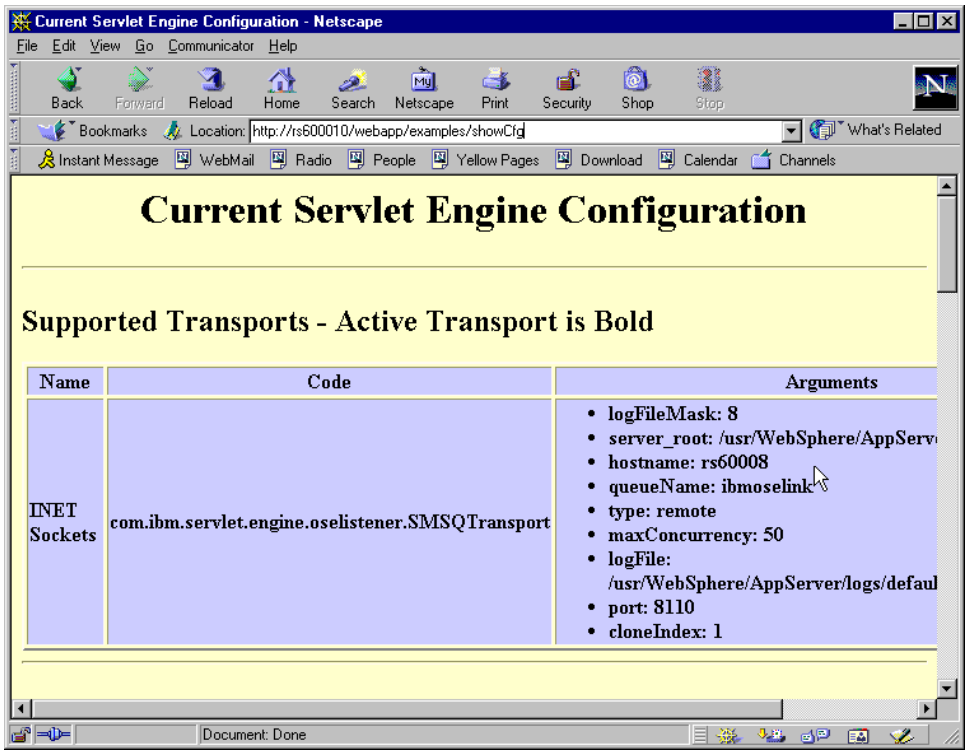

Figure 99. Access to the http://rs600010/webapp/examples/showCfg

Notice that the hostname (rs60008) is the name of the machine running the Default Server and not the HTTP server (rs600010) or the other WebSphere node on which Sample Server is running (rs600013).

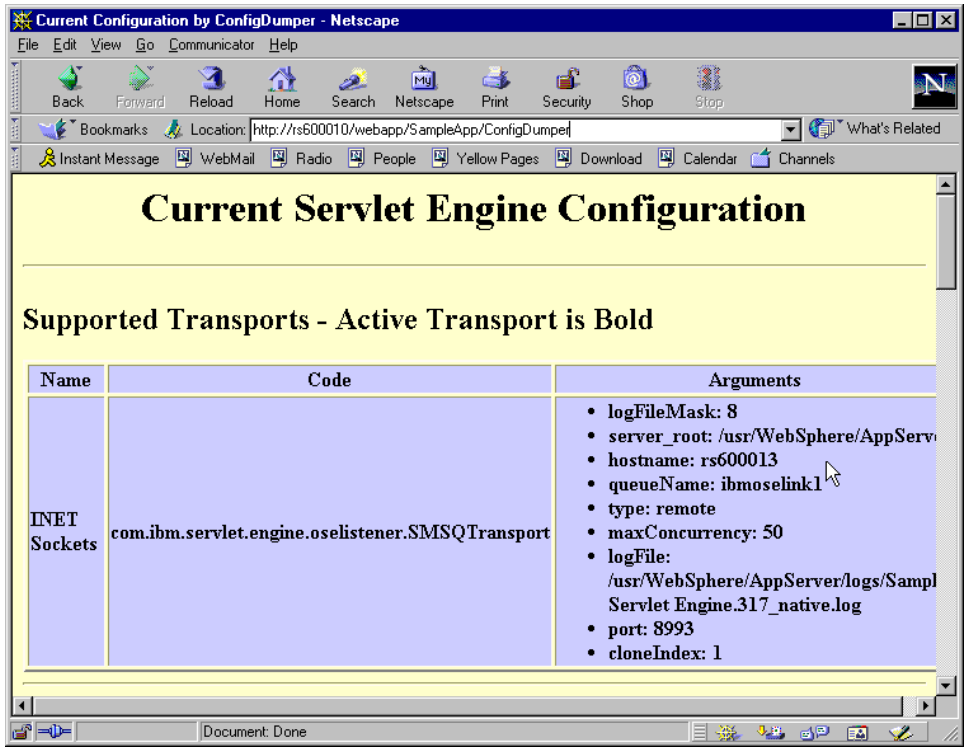

Figure 100. Access to http://rs600010/webapp/SampleApp/ConfigDumper

Notice that the hostname (rs600013) is the name of the machine running Sample Server and not the HTTP server (rs600010) or the other WebSphere node on which Default Server is running (rs60008).

5.9.3 Variation 3: multiple WebSphere nodes with clones

You can configure this topology as an extension to the previous topology. You need to create a clone for each application server on the remote node. Therefore, after you configure clones, both machines should be identical as shown in Figure 101.

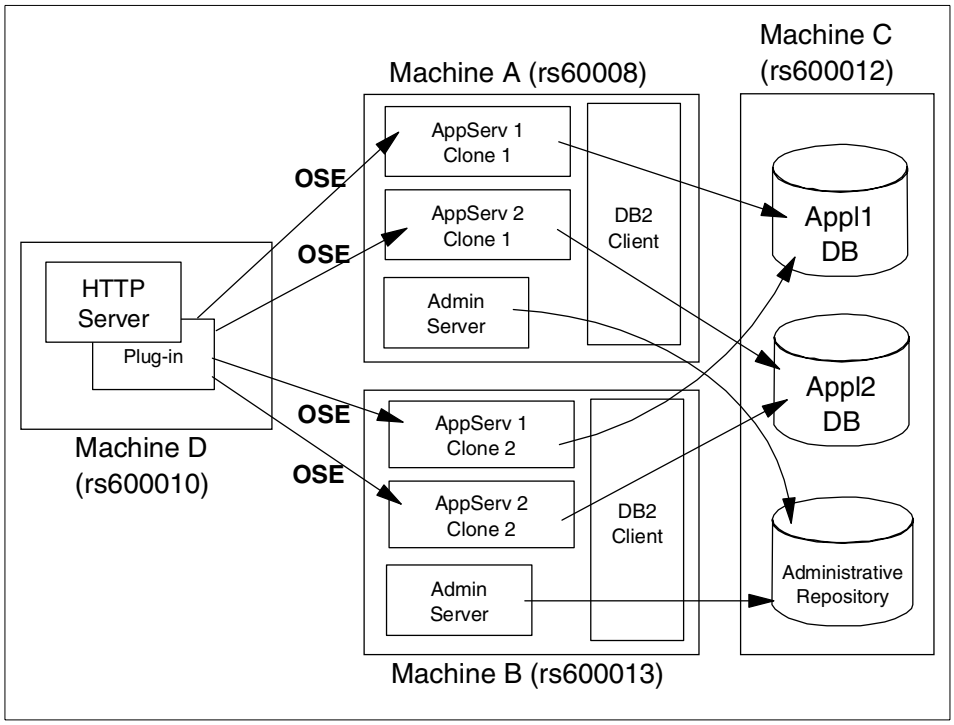

Figure 101. Multiple WebSphere nodes with clones

On the Administrative Console, you can see Default Server and Sample Server on both Machines A (rs60008) and B (rs600013) as shown in Figure 102 on page 135.

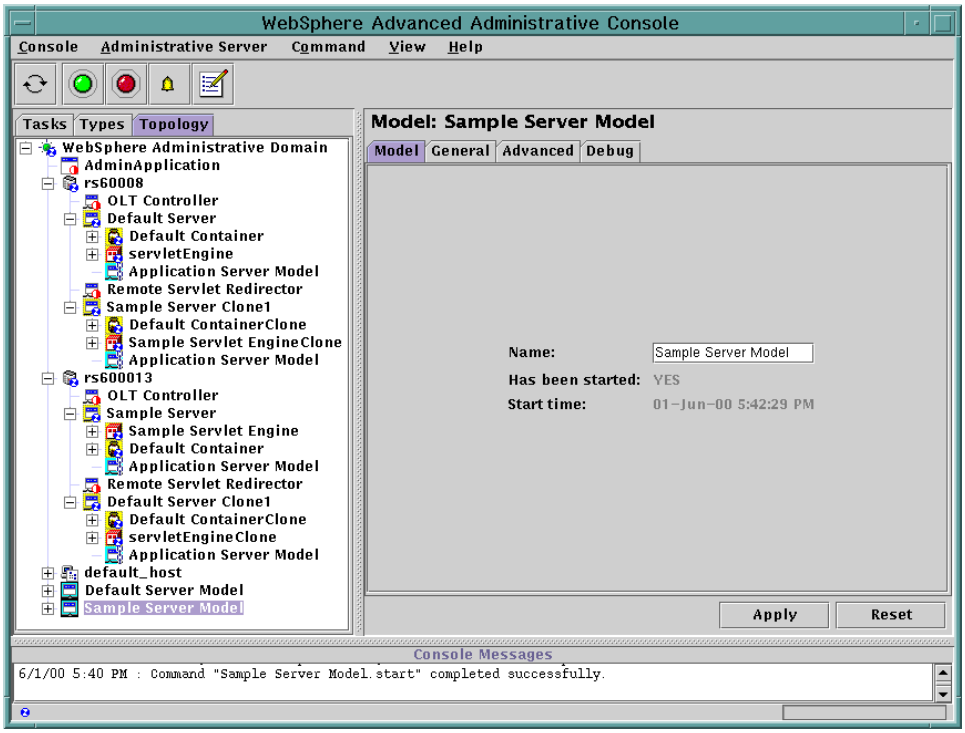

Figure 102. Administrative Console with two clones on each node, two nodes in one domain

The properties files you create should look similar to the ones shown below.

```
#IBM WebSphere Plugin URL Mapping Rules
#Thu Jun 01 17:43:16 EDT 2000
default_host/webapp/SampleApp/ConfigDumper=ibmoselink1
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 103. <was_dir>/temp/rules.properties on Machine D (rs600010)

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu Jun 01 14:39:04 EDT 2000
9.24.104.109=default_host
9.24.104.30=default_host
9.24.104.119=default_host
localhost=default_host
127.0.0.1=default_host
rs600013=default_host
rs600010=default_host
rs60008=default_host
rs600010.itso.ral.ibm.com=default_host
rs600013.itso.ral.ibm.com=default_host
rs60008.itso.ral.ibm.com=default_host
```

Figure 104. <was_dir>/temp/vhosts.properties on Machine D (rs600010)

In this topology, both Default Server and Sample Server have clones. Therefore, the clonescount entry for both the ibmoselink and ibmoselink1 queues is specified as 2. And also, each queue has two host entries to point to both Machines A (rs60008) and B (rs600013).

```
#Thu Jun 01 17:46:16 EDT 2000
ose.srvgrp=ibmoselink,ibmoselink1
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone1.host=rs60008
ose.srvgrp.ibmoselink.clone2.port=8994
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone2.host=rs600013
ose.srvgrp.ibmoselink1.clonescount=2
ose.srvgrp.ibmoselink1.type=FASTLINK
ose.srvgrp.ibmoselink1.clone1.port=8993
ose.srvgrp.ibmoselink1.clone1.type=remote
ose.srvgrp.ibmoselink1.clone1.host=rs600013
ose.srvgrp.ibmoselink1.clone2.port=8111
ose.srvgrp.ibmoselink1.clone2.type=remote
ose.srvgrp.ibmoselink1.clone2.host=rs60008
```

Figure 105. <was_dir>/temp/queues.properties on Machine D (rs600010)

The following are the sample results of this topology.

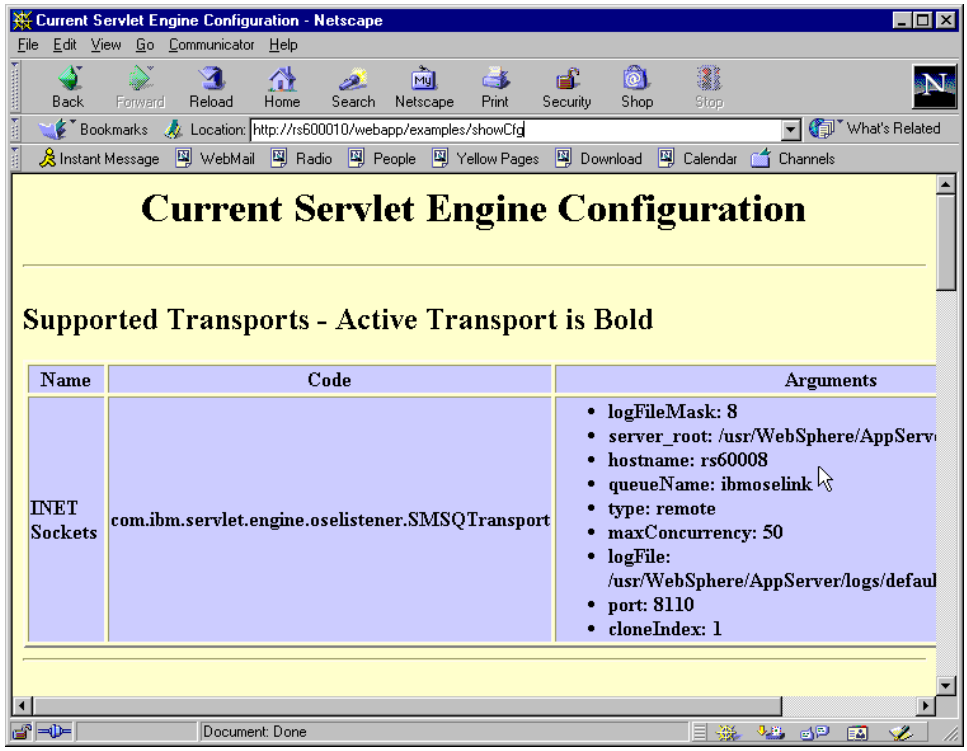

Figure 106. showCfg clone1 (port 8110, cloneindex 1) on Machine A (rs60008)

Notice that the hostname (rs60008) is the name of the machine running Default Server (the servlet engine is listening on port 8110) and not the HTTP server (rs600010) or the other WebSphere node on which Default Server Clone1 is running (rs600013).

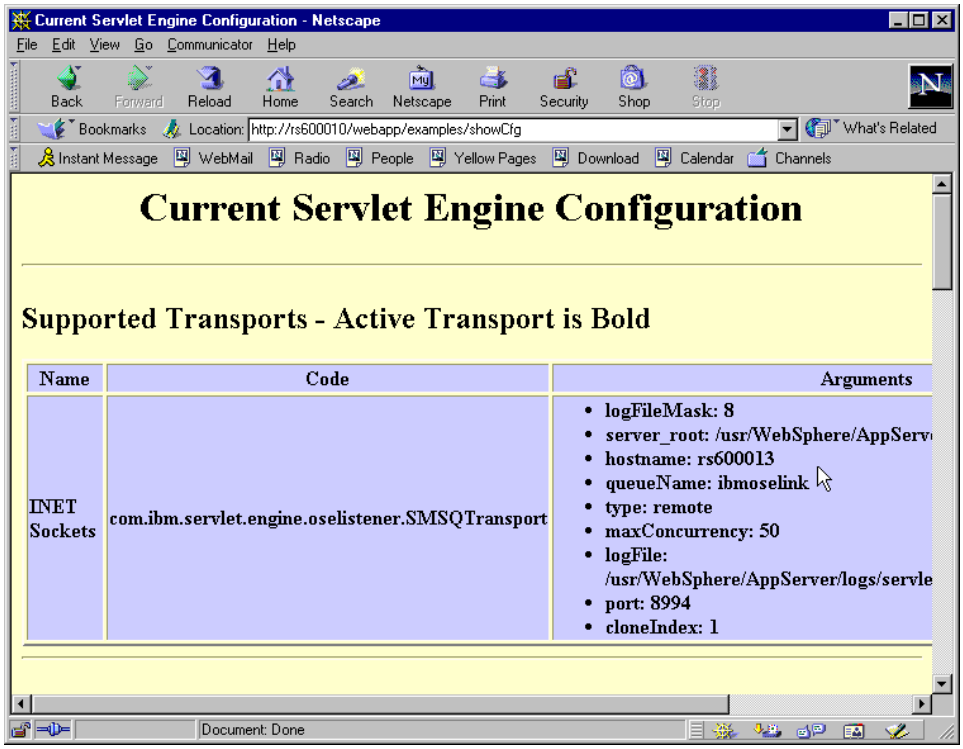

Figure 107. showCfg clone2 (port 8994, cloneindex 1) on Machine B (rs600013)

Notice that the hostname (rs600013) is the name of the machine running Default Server Clone1 (the servlet engine is listening on port 8994) and not the HTTP server (rs600010) or the other WebSphere node on which Default Server is running (rs60008).

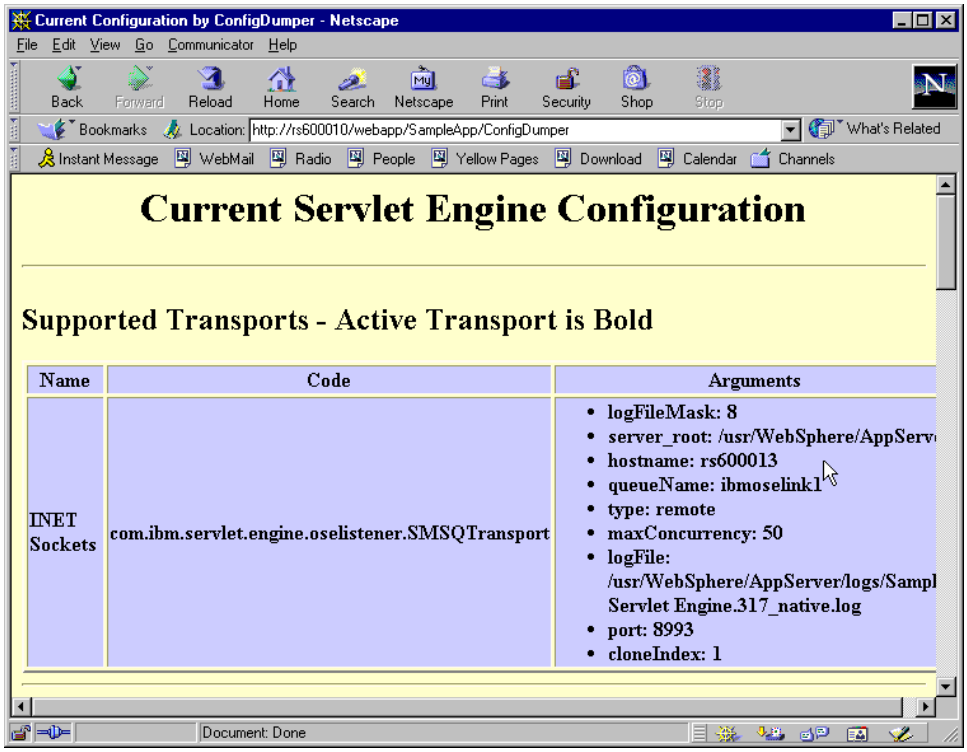

Figure 108. ConfigDumper clone1 (port 8993, cloneindex 1) on Machine B (rs600013)

Notice that the hostname (rs600013) is the name of the machine running Sample Server (the servlet engine is listening on port 8993) and not the HTTP server (rs600010) or the other WebSphere node on which Sample Server Clone1 is running (rs60008).

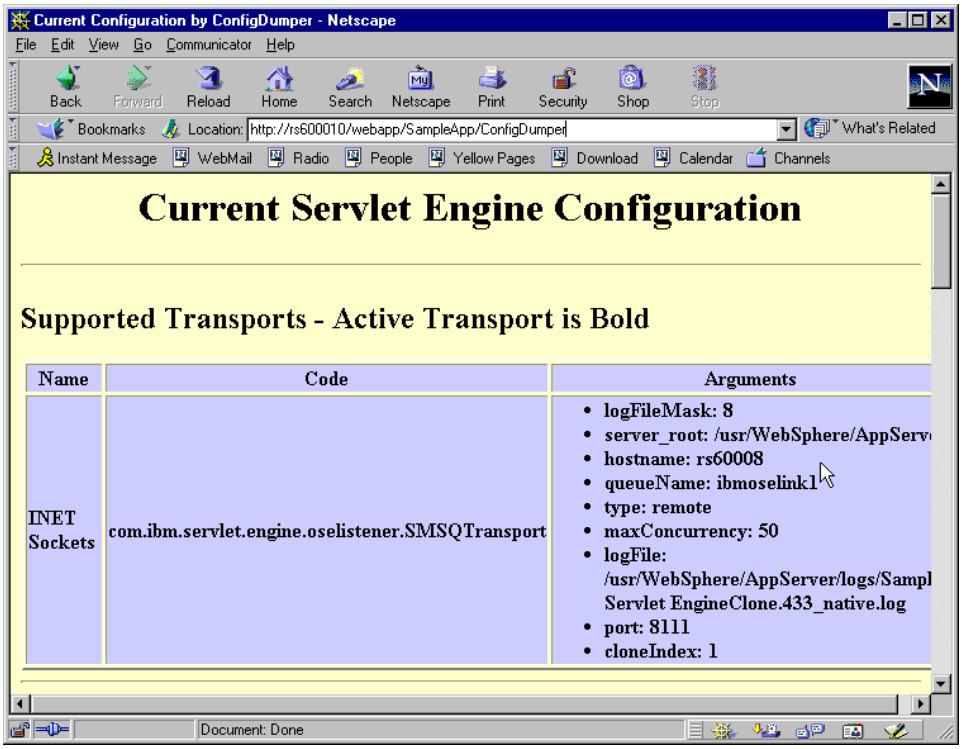

Figure 109. ConfigDumper clone2 (port 8111, cloneindex 1) on Machine A(rs60008)

Notice that the hostname (rs60008) is the name of the machine running Sample Server Clone1 (the servlet engine is listening on port 8111) and not the HTTP server (rs600010) or the other WebSphere node on which Sample Server is running (rs600013).

5.9.4 Variation 4: session affinity

When using multiple clones, the way in the which the Web server selects the application server clone to receive each request is a combination of round-robin with session affinity:

- If session persistence is not enabled (the default), then session affinity is not enabled. In this case the Web requests are distributed among all available clones in a strict round-robin fashion that is each clone gets the next request in turn.
- If session clustering and session affinity are enabled:
  - For all Web requests that are not associated with a session, or for the first request of each session (for example, before a cookie and/or

session ID have been established), the same round-robin distribution policy is used.

- For all Web requests that are associated with a session, WebSphere attempts to consistently route all requests within the same session to the same clone. Different sessions will be assigned semi-randomly to different clones.

  Note however that there is no guarantee that the same clone will absolutely always be used for all requests within one session. The affinity is only a “best-effort”, subject to occasional switches imposed by the implementation (for example when the number of available clones changes during the life of the session). The system relies on the session clustering facilities to ensure that session state will not be lost in the event of such a switch.

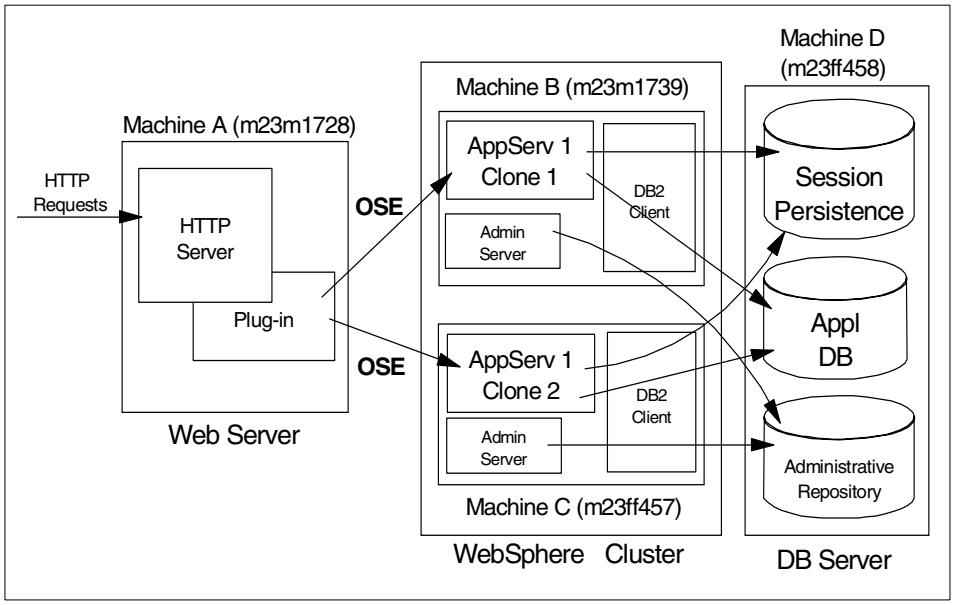

Figure 110. OSE Remote with session affinity

Here are the instructions for setting up OSE Remote in a WebSphere multi-node cluster environment with session affinity:

5.9.4.1 Software installation and starting Administrative Server

Most of the installation and configuration steps are the same as `Variation 1`. Please refer the previous section for more details.

1. Install the Web server on Machine A (m23m1728)

2. Install WebSphere on Machines B and C. Only request the sample configuration to be installed on B and point all WebSphere nodes to the same database to configure WebSphere clustering.
3. Start the Administrative Servers on each WebSphere machine (Machines B and C).
4. Start the Administrative Console on Machine B (m23m1739).

   Note that you can start the Administrative Console from any machine.

5.9.4.2 Configure session persistence

After you start up the Administrative Console, you need to configure session persistence. Session persistence is required for session affinity.

(1)Create a sample servlet for session persistence

In our example, we created a sample servlet, called HttpSessionServlet, in which we updated the showCfg servlet to support HTTP sessions as shown in Figure 112 on page 144 and deployed it to WebSphere as shown in Figure 111.

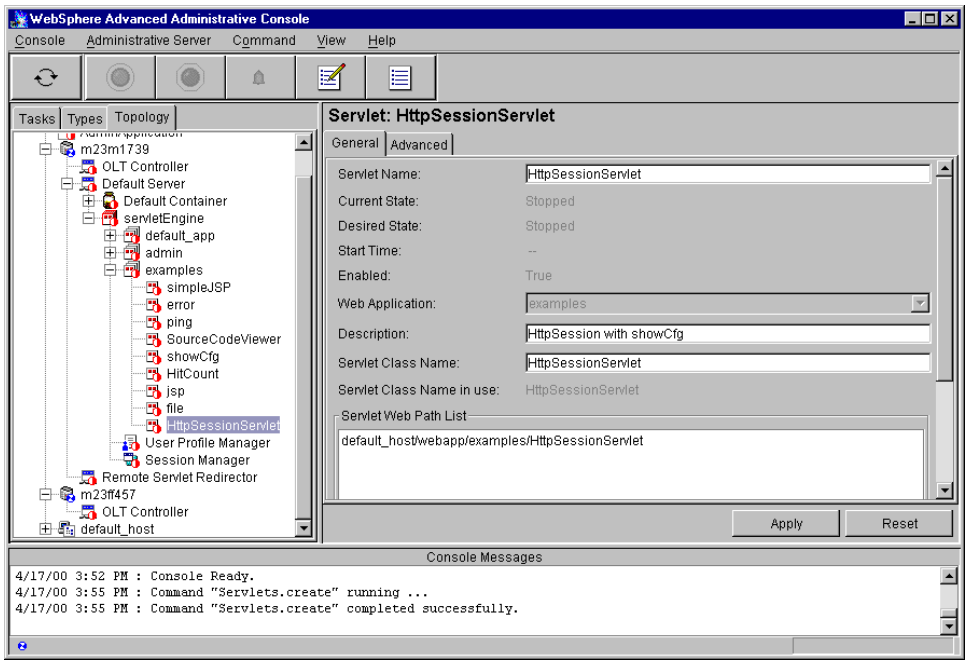

Figure 111. Create HttpSessionServlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Enumeration;
import com.ibm.servlet.engine.*;
import com.ibm.servlet.engine.srt.*;
import com.ibm.servlet.engine.config.*;
import com.ibm.servlet.util.*;
import java.util.*;
/**
 * Config Dumper
 * <P> This servlet dumps the current servlet engine configuration
 */
public class HttpSessionServlet extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException
    {
        String cmd = req.getParameter("cmd");
        if ( cmd == null )
        {
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();

   HttpSession session = req.getSession(true);
      int counter = 1;
      if (!session.isNew())  {
        counter = ((Integer)session.getValue("counter")).intValue();
        counter++;
      }
      session.putValue("counter", new Integer(counter));
out.println("<HTML><HEAD><TITLE>Current Servlet Engine
Configuration</TITLE></HEAD><BODY BGCOLOR=\"#FFFFDD\">");
            ServletEngineInfo engineInfo =
ServletEngine.getEngine().getInfo();
            out.println("<H1><CENTER>Current Servlet Engine
Configuration</CENTER></H1><HR>");

      out.println("<H2>Counter =");
      out.println(counter);
      out.println("</H2>");
      out.println("<H2>Supported Transports - Active Transport is
Bold</H2>");
```

Figure 112. HttpSessionServlet.java (partial)

(2) Create a database for session persistence
To use persistent sessions, you have to create a database to store the session object on your database server.

(3) Configure JDBC drivers
You need to specify the JDBC driver which you will use for session persistence.

From the Types tab, select and right click **JDBC Drivers**. Then click **Create** as shown in Figure 113.

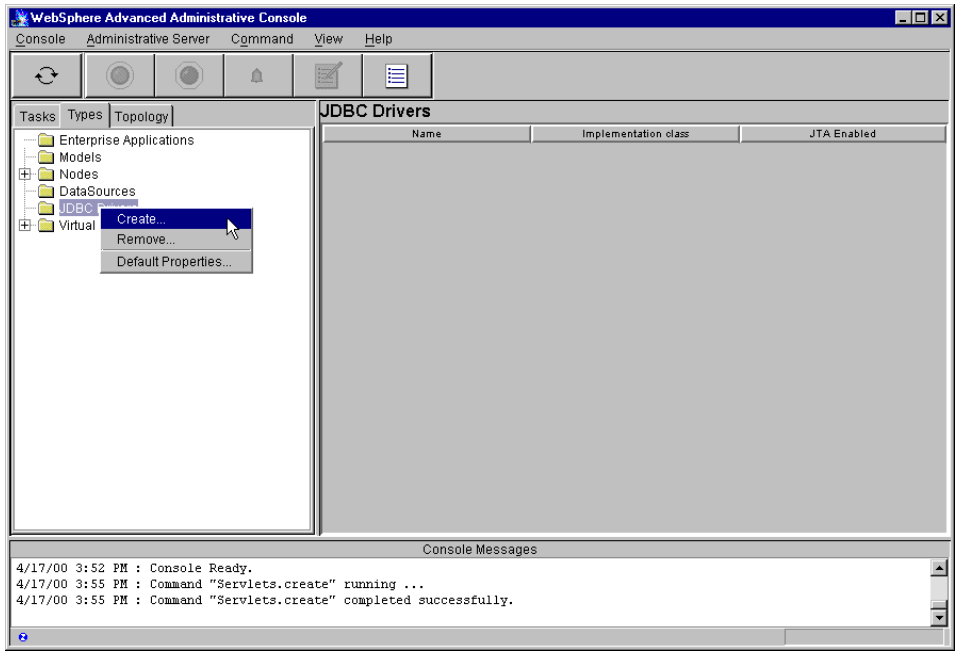

Figure 113. Create JDBC drivers

The Create a JDBC Driver panel will appear. You need to specify the JDBC driver name and implementation class name. Then click the **Create** button as shown in Figure 114 on page 146.

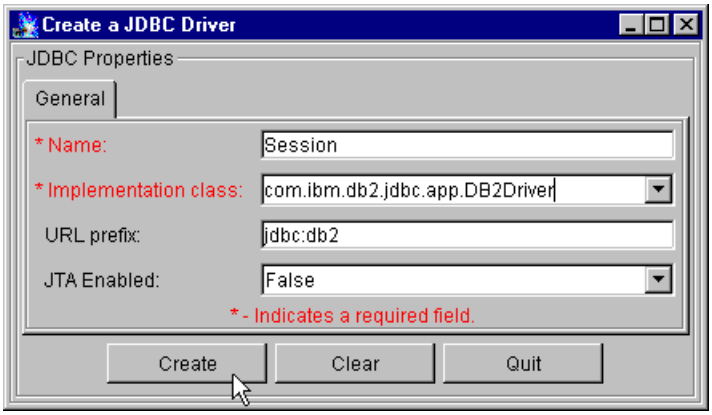

Figure 114. Create a JDBC driver

You will see the following message if you create a JDBC driver successfully.

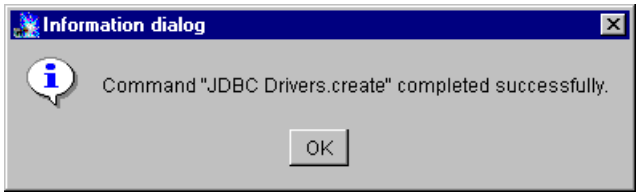

Figure 115. JDBC driver created message

(4) Configure datasource

Next, you need to define the database as a DataSource object in WebSphere.

From the Types tab, select **DataSource** and click **Create** as shown in Figure 116 on page 147.

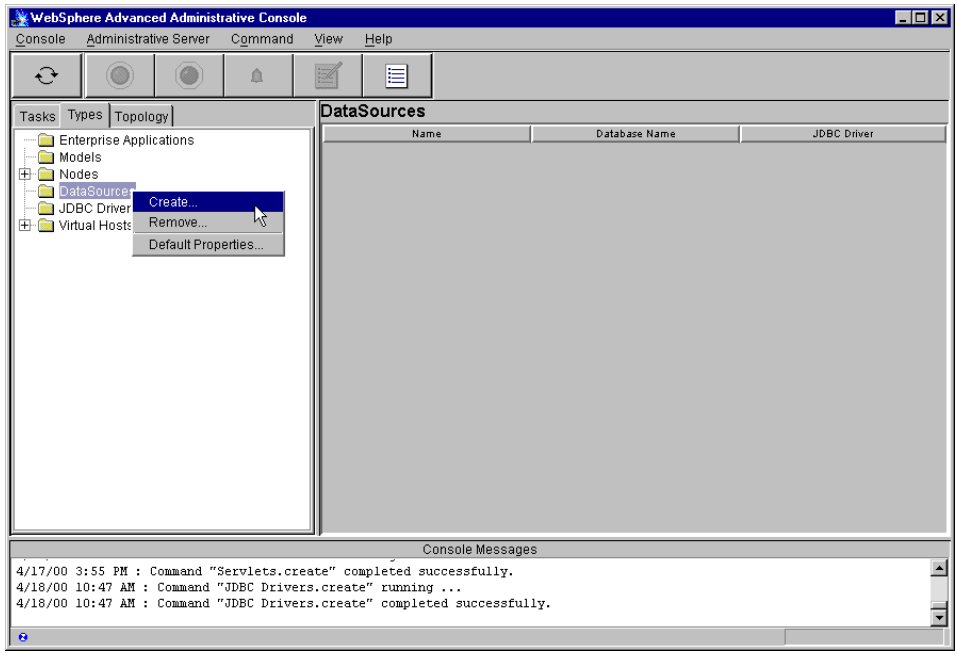

Figure 116. Create DataSource

Then you will get the Create a DataSource window.

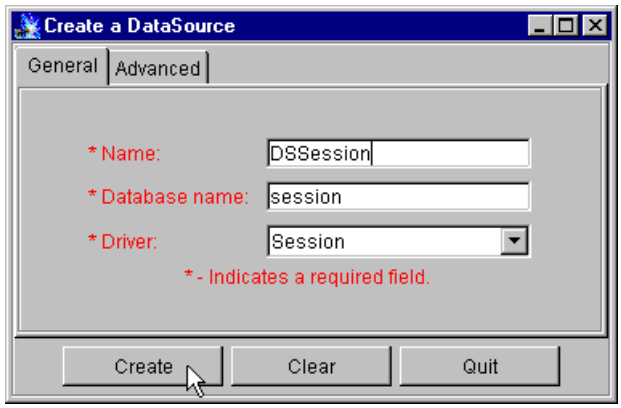

Figure 117. Create a DataSource

You need to specify the name of the datasource, Database name for session persistence which you created, as shown in “(2) Create a database for session persistence” on page 145, and JDBC driver which you created, as

shown in “(3) Configure JDBC drivers” on page 145. Then click the **Create** button.

You will see the following message if you create a datasource successfully.

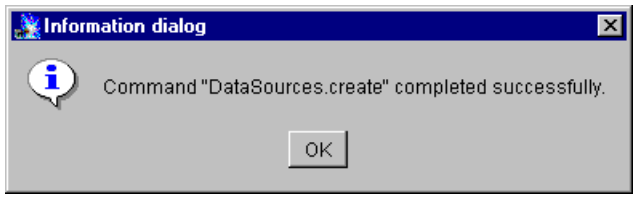

Figure 118. DataSource created message

As you can see in Figure 119, there is a DataSource for session persistence which you configured.

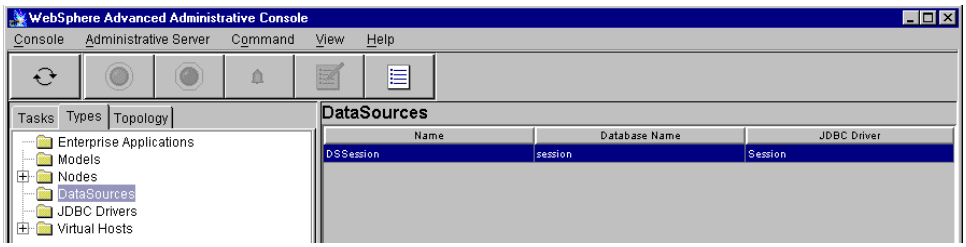

Figure 119. DataSource configuration

(5) Configure session manager

Next, you must configure a session manager.

To modify the Session Manager setting to allow persistent sessions:

Click **Topology -> WebSphere Administrative Domain -> hostname -> Default Server -> servletEngine -> Session Manager -> Enable** tab, set Enable Persistent Sessions to **Yes** and click **Apply** as shown in Figure 120.

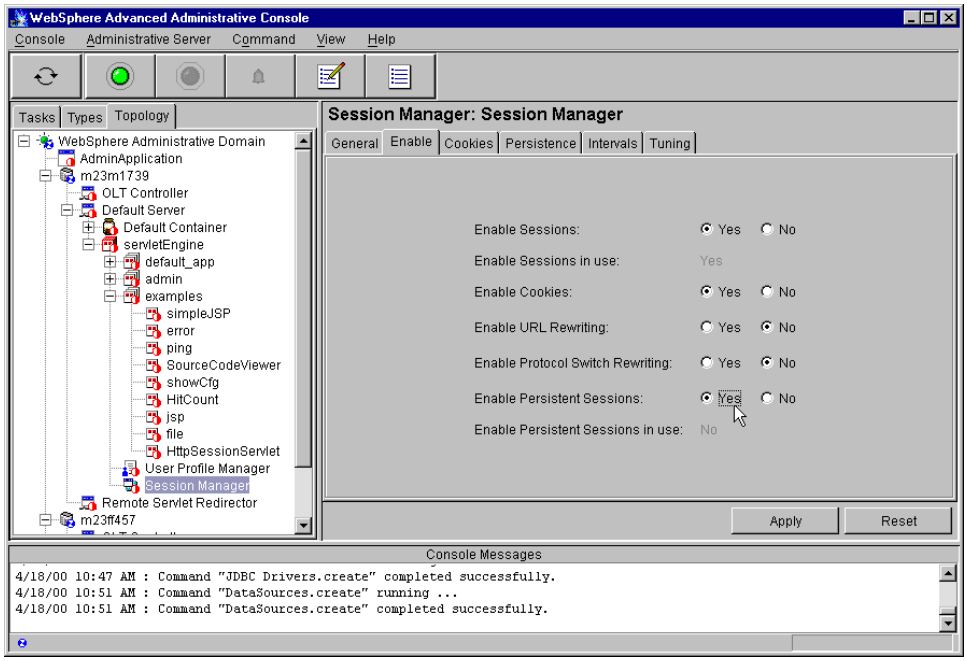

Figure 120. Session Manager: Enable

On the **Session Manager -> Persistence** tab as shown in Figure 121 on page 150, click the datasource **Change** button and select the datasource you defined before. Again, click **Apply**.

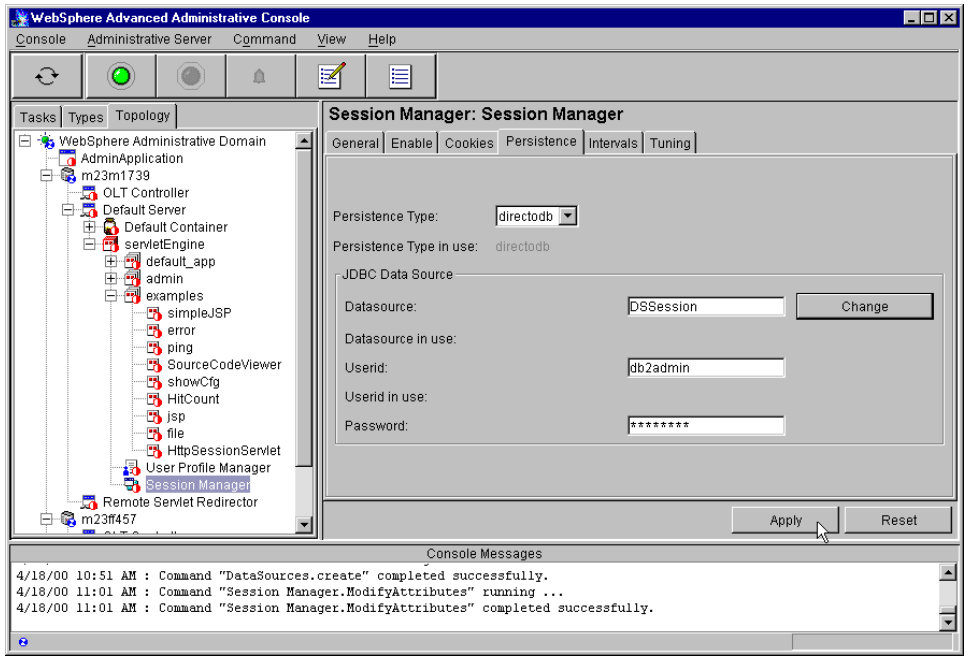

Figure 121. Session Manager: Persistence

5.9.4.3 Configure the application server model and clone

(1) Aliases

Make sure that default_host virtual host is configured with host aliases for all of the machines, including the HTTP server (for example, for A, B, and C).

(2) Create an application server model on Machine B

Make certain that the servlet engine is configured to use OSE INET Sockets (this is the default on Solaris machines but not on AIX or Windows).

(3) Create a clone on Machine C (m23ff457)

Create application server clone (Default Server 1) on Machine C (m23ff457).

(4) Start the application server model

Start the application server model.

(5) Configure plug-in properties files

You need to configure plug-in properties files. In the following example, we use manual configuration for OSE Remote. See 5.5, “Configure the plug-in for OSE Remote” on page 103 for more information, especially if you want to use a script file to configure OSE Remote.

(5-1) Copy properties files

You will then need to manually copy (via ftp, with diskette and so on) the following three files from Machine B (m23m1739) to Machine A (m23m1728):

- <was_dir>/temp/vhosts.properties
- <was_dir>/temp/rules.properties
- <was_dir>/temp/queues.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Tue Apr 18 11:15:58 EDT 2000
9.24.104.73=default_host
m23m1728.itso.ral.ibm.com=default_host
9.24.104.63=default_host
m23m1739=default_host
m23ff457=default_host
m23m1739.itso.ral.ibm.com=default_host
9.24.104.88=default_host
localhost=default_host
m23ff457.itso.ral.ibm.com=default_host
127.0.0.1=default_host
m23m1728=default_host
```

Figure 122. <was_dir>/temp/vhosts.properties on Machine A (m23m1728)

```
#IBM WebSphere Plugin URL Mapping Rules
#Tue Apr 18 11:15:58 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/HttpSessionServlet=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 123. <was_dir>/temp/rules.properties on Machine A (m23m1728)

(5-2) Modify the queues.properties file

We show you the queues.properties file which is on Machine B (m23m1739).

```
#IBM WebSphere Plugin Communication Queues
#Tue Apr 18 11:15:58 EDT 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=local
```

Figure 124. <was_dir>/temp/queues.properties on Machine B (m23m1739)

We copied it (with a floppy diskette) from Machine B and stored it under the <was_dir>/temp directory on Machine A (m23m1728) and modified it as shown in Figure 125.

```
#IBM WebSphere Plugin Communication Queues
#Tue Apr 18 11:15:58 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.keepaffinity=true
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone1.host=m23m1739
ose.srvgrp.ibmoselink.clone2.port=8110
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone2.host=m23ff457
```

Figure 125. modified queues.properties on Machine A (m23m1728)

Make sure that the port number you specify in the queues.properties file is the same number as you see on the Servlet Engine configuration panel. In order to support session affinity, a new entry is introduced. For V3.02x, you can specify the entry “ose.srvgrp.<link_name>.keepaffinity=” (default value is true) in the <was_dir>/temp/queues.properties as shown in Figure 125 and for V3.5, you can specify the entry “ose.session.affinity=” (default value is true) in the <was_dir>/properties/bootstrap.properties as shown in Figure 126.

```
##
# Set HTTP Session affinity
# values of ose.session.affinity may be either
# true|false
ose.session.affinity=true
```

Figure 126. <was_dir>/properties/bootstrap.properties (WebSphere V3.5) on Machine A

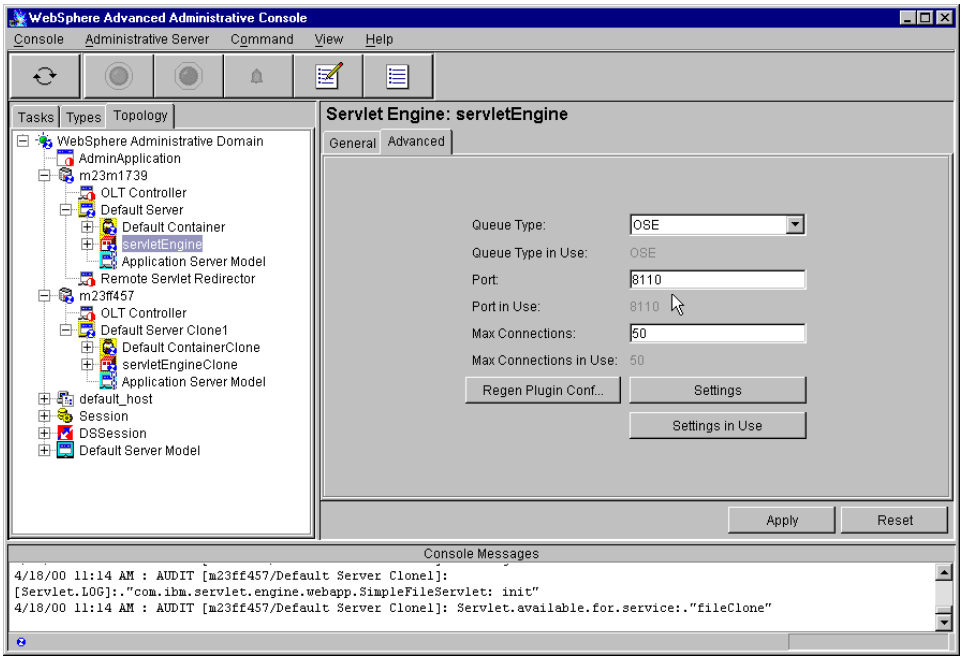

Figure 127. Servlet Engine: Advanced

(6) Modify bootstrap.properties for WebSphere security

If you are using WebSphere security, you need to modify <was_dir>/properties/bootstrap.properties. See 5.6, “Configure bootstrap.properties for security” on page 109 for more information.

5.9.4.4 Start the Web server

Start the Web server on Machine A (m23m1728).

5.9.4.5 Test the configuration

Access the following URI:

```
http://m23m1728/webapp/examples/HttpSessionServlet
```

Notice that the hostname (m23ff457) is always the name of the machine running WebSphere and not that of the HTTP server (m23m1728).

Also notie that the counter number is incremented.

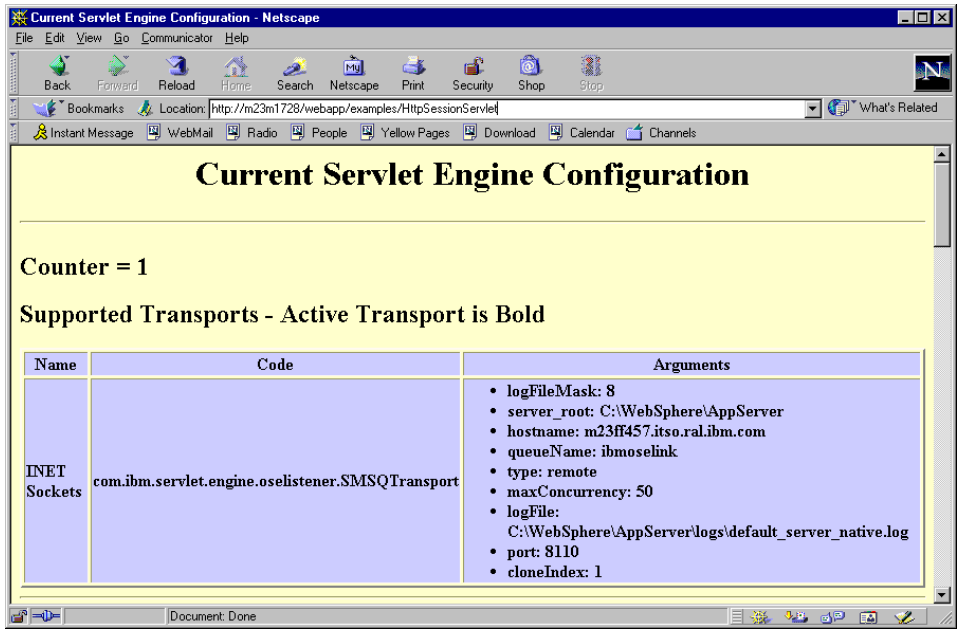

Figure 128. The first response from Machine B (m23ff457) clone1 with the same SessionID

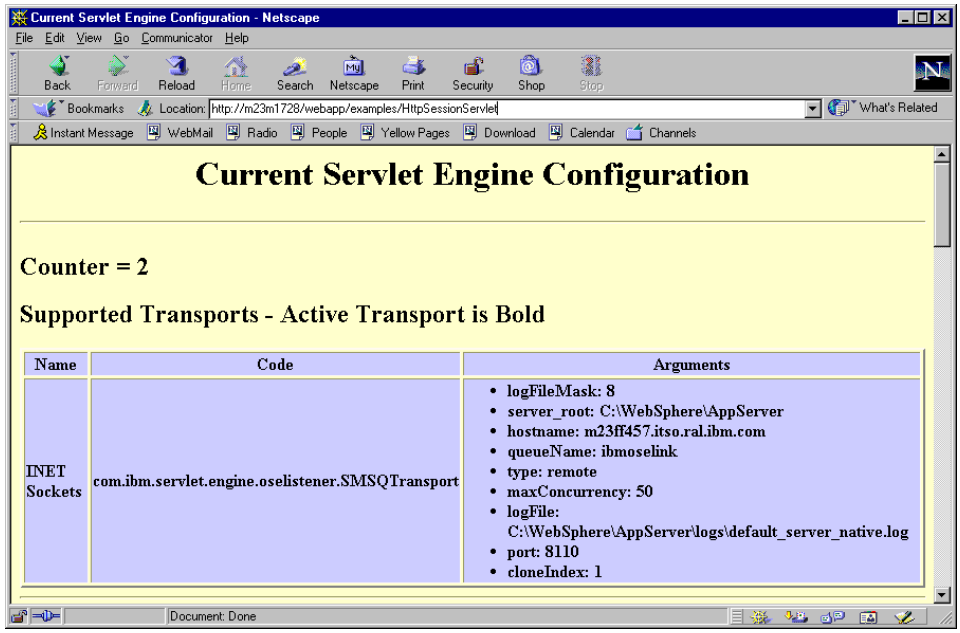

Figure 129. The second response from Machine B (m23ff457) clone1 with the same SessionID

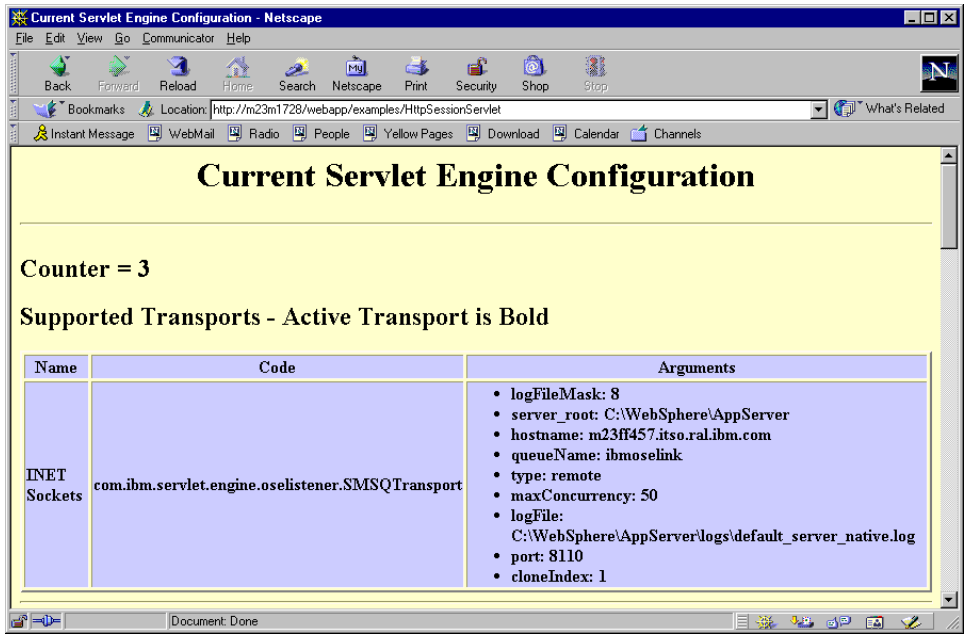

Figure 130. The third response from Machine B (m23ff457) clone1 with the same SessionID

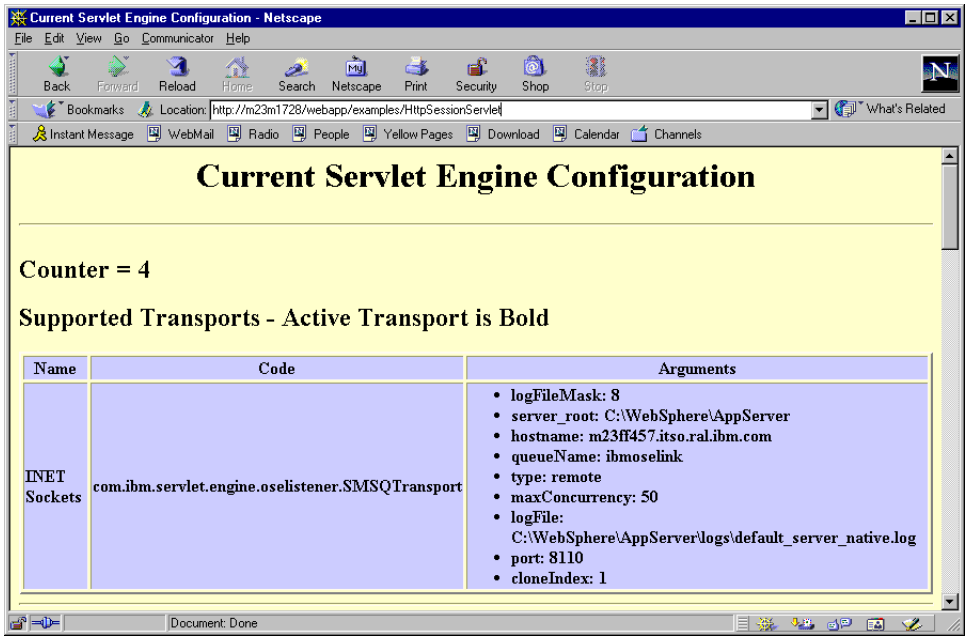

Figure 131. The fourth response from Machine B (m23ff457) clone1 with the same SessionID

5.10 Maintenance/troubleshooting

There are some considerations when performing maintenance on an application server.

5.10.1 Terminating an application server

When an application server aborts suddenly (we used the `kill -9 PID` command to simulate this situation), WebSphere automatically restarts the server. No action is required.

The plugin configured for OSE Remote will automatically detect this failure and route new requests to alternative servers. However, some Web clients may experience brief transient errors.

If there are no clones of this server, no requests will be served and errors will be reported until the application server restarts.

5.10.2 Stopping an application server

When an application server is stopped for administrative reasons (via the Administrative Console or from the command line), the plugin configured for OSE Remote will automatically route new requests to alternative servers

On rare occasions, while a server is terminating, it is possible that some number of Web clients may experience transient errors before the client requests are dispatched to an alternate server.

Of course the plugin configure for OSE Remote can only dispatch requests to alternate servers for application servers that are part of a model. If an the application server has no clones, no client requests will succeed.

5.10.3 Restarting a clone

After you restart a clone, the plug-in detects it and browsers automatically get access to it. No actions are required.

5.10.4 Restarting a machine

After you restart a machine, the Administrative Server, the application server and its clones, the plug-in detects it and browsers automatically get access to it. No actions are required.

5.10.5 OSE and TCP/IP network details

5.10.5.1 Stopping an application server process

When an application server is stopped, assume that the TCP/IP stack is still running on Machine B. Therefore, a WebSphere plug-in on the HTTP server can talk to the TCP/IP stack on the WebSphere machine. But, there is no AppServer1 Clone1 running in our case.

So, the TCP/IP stack on Machine B sends an RST packet back to the plug-in on Machine A and does the same process a couple of times.

Finally, the plug-in stops sending SYN packets to communicate with AppServer1 Clone1. Then, the plug-in sends a request to the next clone, in our case AppServer1 Clone2, and gets a response from it as shown in Figure 132.

The plug-in can find and get the next clone even if it is running on a different machine than the AppServer1 Clone1 as shown in Figure 133 on page 159.

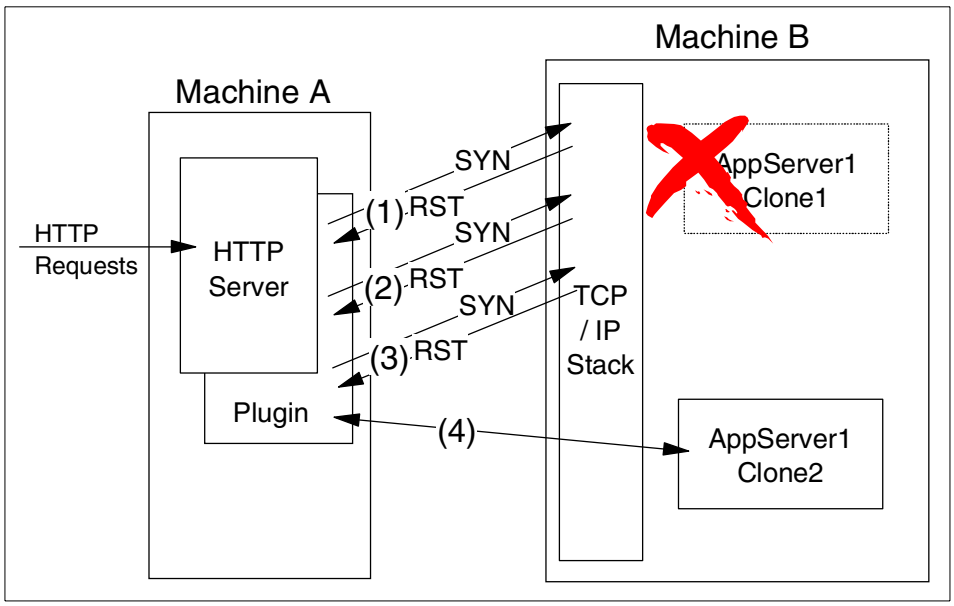

Figure 132. Stopping a clone (case 1, clones on one WebSphere machine)

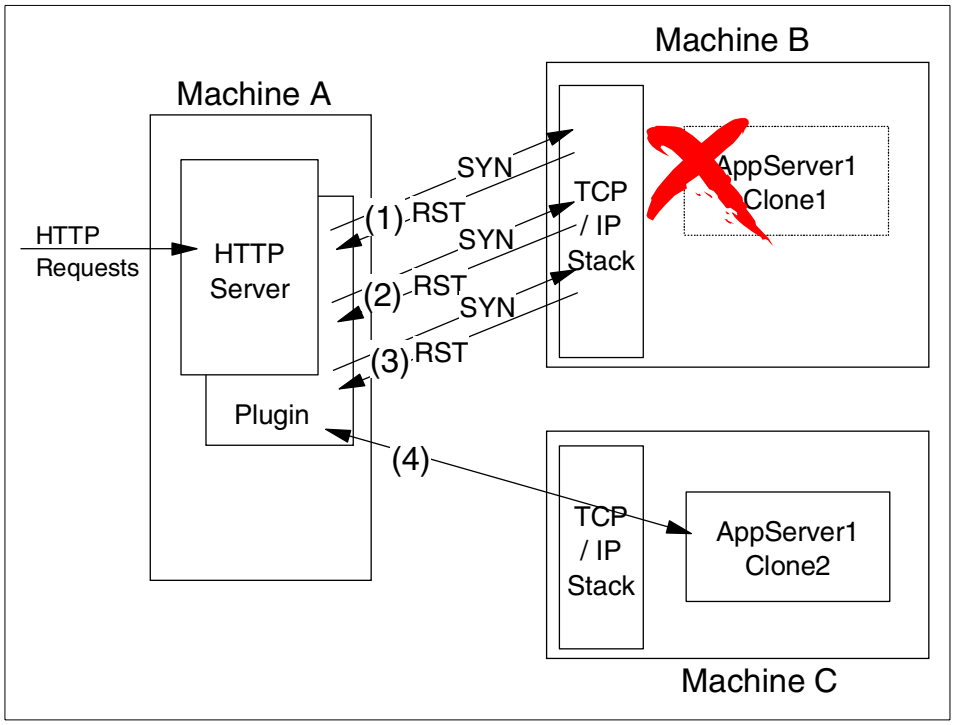

Figure 133. Stopping a clone (case 2, clones on several WebSphere machines)

5.10.5.2 Losing network connection

When a node loses its network connection, requests will automatically be routed to other clones on other nodes. No additional actions are required.

In this case, a WebSphere plug-in on the HTTP server cannot talk to the TCP/IP stack on the WebSphere machine. Therefore, even when the plug-in sends SYN packets a couple of times, the plug-in does not get any response packet back from Machine B. After the plug-in sends a SYN packet a couple of times, the plug-in gives up sending a request to the node and forwards the request packet to the next clone. If the next clone is on the same node which is not running, the plug-in sends the request to the node and does not receive a response. It then sends the request to the next clone as well. Finally, the plug-in will get a response from a clone which is running on another machine as shown in Figure 134.

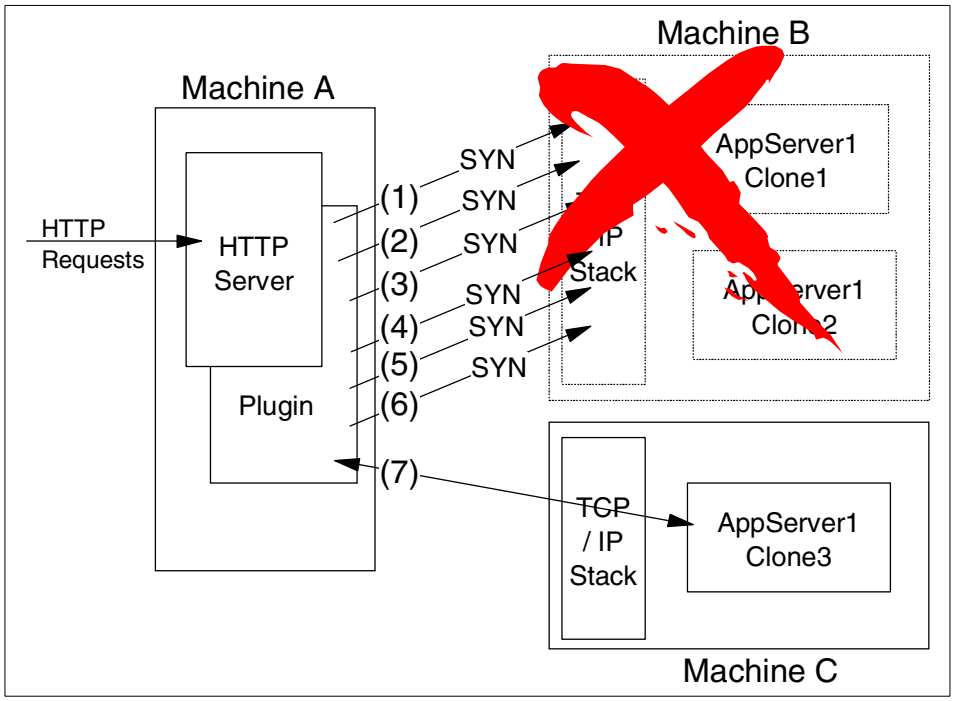

Figure 134. Losing a network connection

5.10.6 Modifying the application server

If any changes are made to the application server that affects the queues.properties, rules.properties file and vhosts.propeties file for the HTTP server node, the plug-in files need to be regenerated as described in 5.5, “Configure the plug-in for OSE Remote” on page 103.

Such changes include:

- Add/remove a URL (Web resources) to the environment.
- Secure/unsecure a URI (when you add security to a URI, an “A” is added to it, changing its name).
- Add/remove a host alias.
- Change the queue properties of a servlet engine (name, port).
- Add/remove a servlet engine.
- Add/remove a clone.

After the files are regenerated, the HTTP Server must be restarted.

Chapter 6. Thick Servlet Redirector

Thick Servlet Redirector runs as a process on the same server as the HTTP server, intercepts the OSE protocol messages and forwards each servlet request over IIOP (or IIOP/SSL) to an appropriate servlet engine. In this configuration a WebSphere Administrative Server runs on the same box as the Redirector and takes care of process configuration management. In addition, this topology requires configuration of the appropriate database server client software on the HTTP server machine, which may not be prudent in a secure environment.

This chapter provides instructions for the administrative tasks required to configure the Servlet Redirector and remote application server machines.

We will discuss the following steps for setting up this configuration:

1. Installation summary
2. Configure the Administrative Server
3. Start the Administrative Server and Console
4. Configure the application server
5. Configure the Servlet Redirector
6. Start the servers
7. Test the servers
8. Maintenance/trableshooting
9. Related topologies

6.1 Overview of the configuration

A Web server, an application server and a database server are installed in 3 physically separate machines, Machine A, Machine B, and Machine C, respectively. All the requests from Web browsers received by the Web server in Machine A are redirected by a Servlet Redirector, which is a component of WebSphere, to the application server in Machine B for processing. Machines A and B each requires an Administrative Server to manage its resources.

Note

All the WebSphere configuration information is stored in a shared database installed on Machine C.

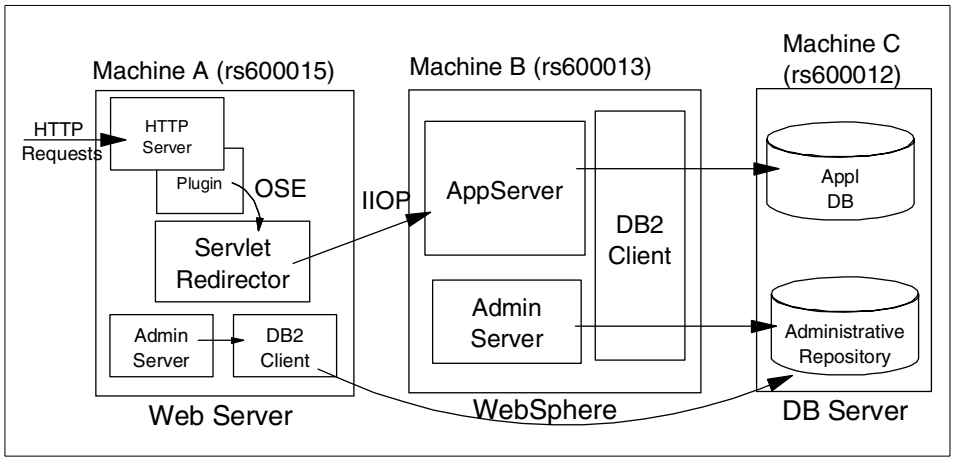

Figure 135. Thick Servlet Redirector

6.2 Installation summary

Table 5 is a summary of the software installation steps. We will discuss only steps 7 and 8 in this chapter. For steps 1 to 6, please refer to the *Getting Started* documentation which is part of the WebSphere product.

Table 5. Software installation steps

Step No.	MachineA (HTTP)	MachineB (WAS)	MachineC (DB)
1			Install DB2 UDB Apply FixPack
2			Create DB
3	Install DB2 client Apply FixPack	Install DB2 client Apply FixPack	
4	Catalog node Catalog db	Catalog node Catalog db	
5	Install JDK (V3.02x)	Install JDK (V3.02x)	
6	Install Web server		
7	Install WebSphere (Production Application Server: -Plugins, -Core Server)		

Step No.	MachineA (HTTP)	MachineB (WAS)	MachineC (DB)
8		Install WebSphere (Production Application Server)	

6.2.1 Install WebSphere on the Web server machine

On Machine A, in addition to the HTTP server (installation Step 4) you must install the DB2 client (installation Step 3).

Then, you need to install the WebSphere Administrative Server and the Web server plug-in (installation Step 7).

To install both of them, you need to select **Custom Installation** and select **Production Application Server** (**Core Server** and appropriate plug-in for your Web server). If you need to install other components, you should select them. For the production environment, you should not need to select anything other than **Production Application Server** (**Core Server** and **plug-in**).

6.2.2 Install WebSphere on the application server machine

On Machine B, where the application server resides, a DB2 client is required (installation Step 3).

Then, you need to install the WebSphere (installation Step 8).

To install it, you need to select **Custom Installation** and select **Production Application Server** (**Core Server**). If you need to install other components, you should select them. For the production environment, you should not need to select anything other than **Production Application Server**. For our example, we installed another component for demonstration purposes.

6.3 Configure the Administrative Server CORBA listener port

Before you start the Administrative Server on Machine B, we recommend you to specify the CORBA listener port for the Administrative Server. Since Servlet Redirector uses IIOP protocol to communicate with WebSphere Administrative Server, you can not specify port numbers without configuring CORBA Listener Port if you have a firewall between Web server (Servlet Redirector) and WebSphere Administrative Server. To do so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the line com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs in the <was_dir>/bin/admin.config file as shown in Figure 136. In our example, we

specify “33000” for the CORBA listener port. The line has been divided with line break for easier reading. In your actual admin.config file, ensure that the line is on a single line. In other words, combine all of the lines from “com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs” to “ListenerPort=33000” on one line, each separated by a space.

```
#Admin Config Rewritten to disable Initial Config
#Tue Jun 06 13:41:41 EDT 2000
com.ibm.ejs.sm.adminServer.qualifyHomeName=true
com.ibm.ejs.sm.util.process.Nanny.maxtries=3
com.ibm.ejs.sm.adminServer.dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
com.ibm.ejs.sm.adminServer.disableEPM=true
com.ibm.ejs.sm.adminServer.dbUser=db2inst1
.....
.....
com.ibm.ejs.sm.adminServer.initializer=com.ibm.ejs.security.Initializer
,com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer,com.ibm.servl
et.config.InitialSetupInitializer
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
-Dcom.ibm.CORBA.ListenerPort=33000
com.ibm.ejs.sm.util.process.Nanny.path=/usr/WebSphere/AppServer/bin:/us
r/WebSphere/AppServer/lib/odbc/lib:/home/db2inst1/sqllib/bin:/home/db2i
nst1/sqllib/function:/usr/jdk_base//bin
com.ibm.CORBA.ConfigURL=file:/usr/WebSphere/AppServer/properties/sas.se
rver.props
```

Figure 136. CORBA listener port for the Administrative Server on Machine B

6.4 Start the Administrative Server and Console

You need to start the Administrative Server and Console.

1. Start a DB2 instance for the Administrative Repository on Machine C (in our case, rs600012) if it is not started.
2. Start Administrative Server on Machine B (in our case, rs600013).

   In our example, we specified “install.initial.config=true” in the admin.config file to install the Default Server.
3. Start Administrative Server on Machine A (in our case, rs600015).
4. Start Administrative Console on Machine B.

   Note: You can use any machine for your Administrative Console.

5. When the console finishes initializing, make sure you see both Machine A and Machine B in the Topology page as shown in Figure 137.

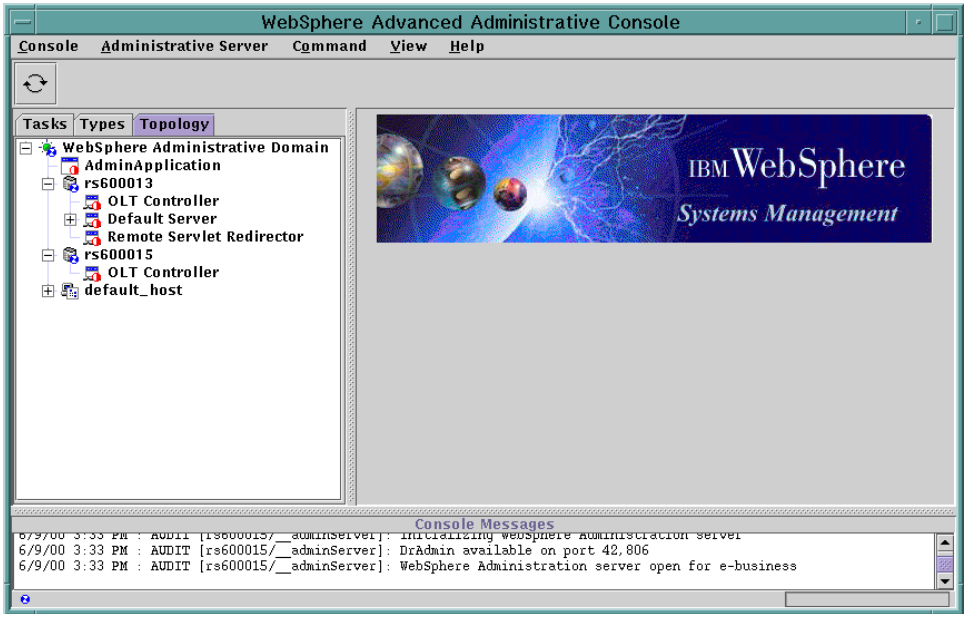

Figure 137. WebSphere Advanced Administrative Console

6.4.1 Add host alias for Machine A

Add aliases for Machine A. These aliases are needed so that the virtual host will accept requests redirected from the Servlet Redirector on Machine A with Machine A's URI instead of Machine B's URI. You may add Machine A's hostname (rs600015), IP address, and fully qualified name.

To add the host aliases:

1. Click the **Topology** tab on the Administrative Console.
2. Click **default_host** in the Topology tree.
3. Click the **Advanced** tab in the Virtual Host pane. Enter the hostname (also IP address and FQDN if you want) for the HTTP server machine in the Host Aliases and click the **Apply** button as shown in Figure 138 on page 166.

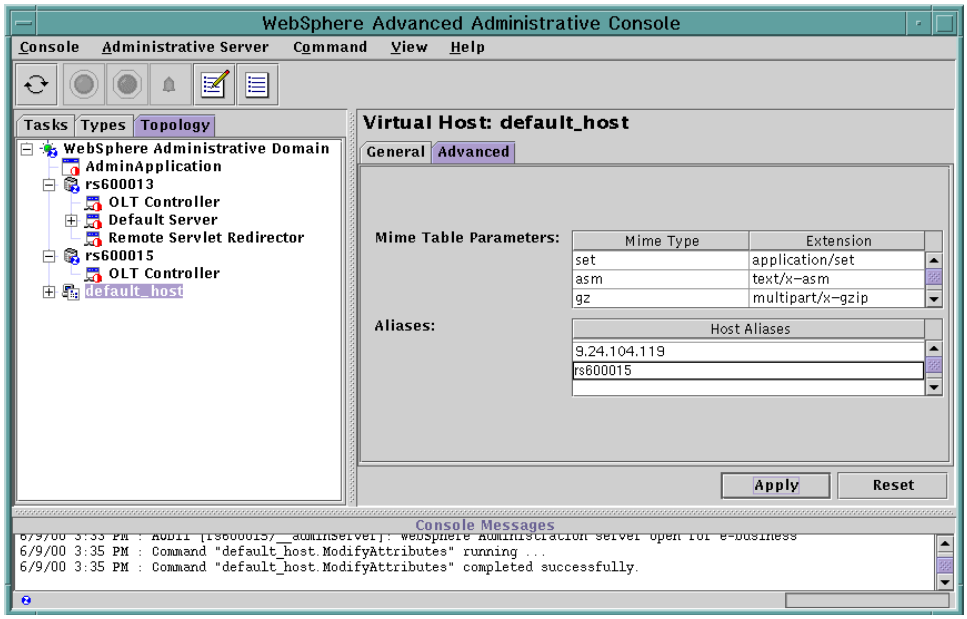

Figure 138. Add aliases

The host aliases have been added.

6.5 Add the CORBA listener port for the application server

We recommend you specify the CORBA listener port for the application server. Since Servlet Redirector uses IIOP protocol to communicate with Application Server, you can not specify port numbers without configuring CORBA Listener Port if you have a firewall between Web server (Servlet Redirector) and Application Server. To do so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the command line arguments in the General tab in the application server as shown in Figure 139. In our example, we specify 35000. Please choose a port that is not already in use.

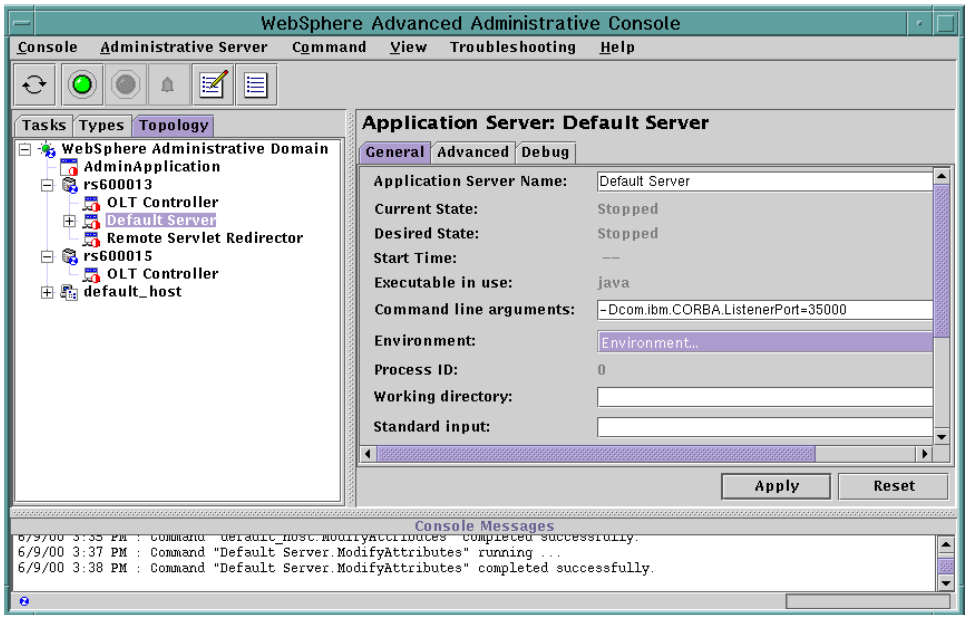

Figure 139. CORBA listener port for an application server

If you need to create clones, do so now. Note that each application server clone on the same machine has to be specified whith a different port number for the -Dcom.ibm.CORBA.ListernerPort argument. If clones have been specified with the same port number on the same machine, you cannot start the clone. But, you can specify the same number for a clone which runs on a different machine.

6.6 Configure and enable Servlet Redirector

You need to create the Servlet Redirector on Machine A (rs600015) following the instructions below.

From the Topology tab in the Administrative Console, expand the tree under Machine A (rs600015).

You might see a Remote Servlet Redirector, which is created automatically during WebSphere installation if you installed components other than what we installed on Machine A (rs600015). If you see one, you just need to update its properties. We will not discuss this but you are able to get the idea from the following sections.

6.6.1 Create the Servlet Redirector

1. From the Topology tab, select a node for the Servlet Redirector. For our example, we selected rs600015 (Machine A).
2. Right-click the node (rs600015) and select **Create..** and then click **Servlet Redirector** as shown in Figure 140.

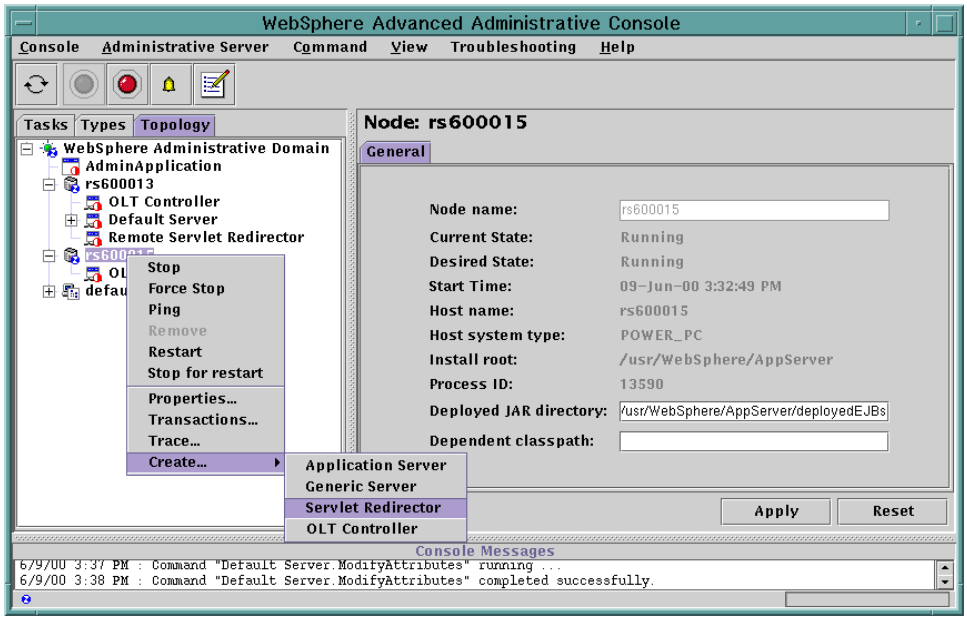

Figure 140. Create Servlet Redirectors

3. Then a Create Servlet Redirector dialog comes up. Enter a name in the Servlet Redirector Name field. In our example, we put the name “Redirector” as the Servlet Redirector on Machine A (rs600015). In the Create Servlet Redirector dialog, choose the node name for Machine A (rs600015) from the drop down list and specify the -Dcom.ibm.CORBA.ListenerPort (in our example, we specify -Dcom.ibm.CORBA.ListenerPort=40000) in the Command line args field as shown in Figure 141. Then click the **Create** button. Please choose a port that is not already in use.

Figure 141. Create Servlet Redirector: General tab

6.6.2 Enable remote Servlet Redirector

You need to enable the Servlet Redirector in order to route requests to nodes other than the node that contains the Web server.

1. Click the Topology tab to display the Topology tree.
2. In the tree, find and right-click the Servlet Redirector (Redirector, in our example) on the node rs600015.
3. On the resulting menu, select and click the **Enabled** menu as shown in Figure 142 on page 170.

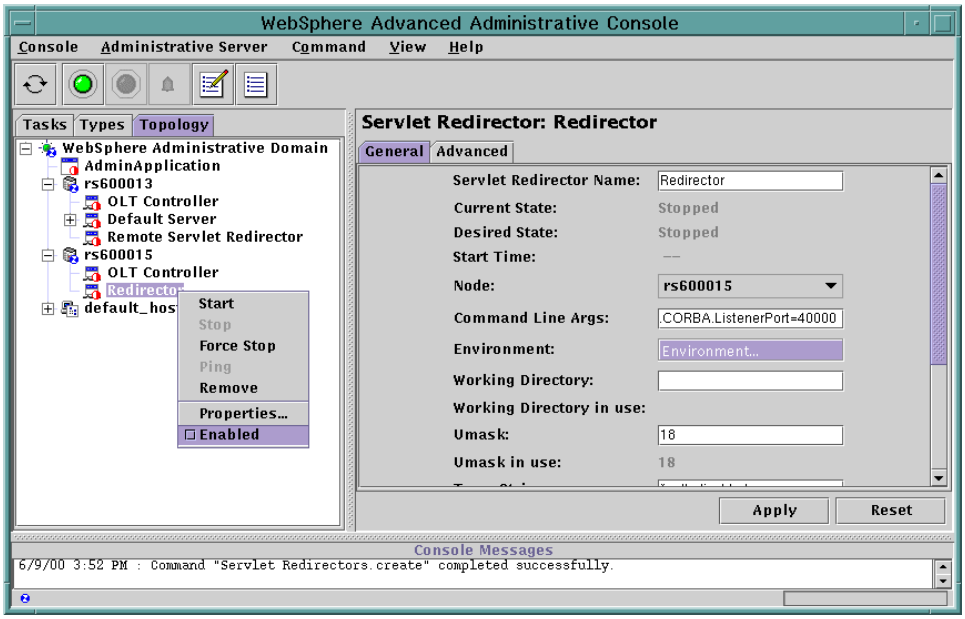

Figure 142. Enable Servlet Redirector on Machine A (rs600015)

4. Right-click the **Servlet Redirector (Redirector)**. Make sure the **Enabled** menu item is checked.

6.7 Start the servers

Now you are ready to start the application server, Servlet Redirector and Web server.

6.7.1 Start the application server

Start the application server (in our example, Default Server) if it is not running. If the application server has been running, stop and start it again.

6.7.2 Start the Servlet Redirector

Now you can start the Servlet Redirector. To do so, select **Start** as shown in Figure 143 on page 171.

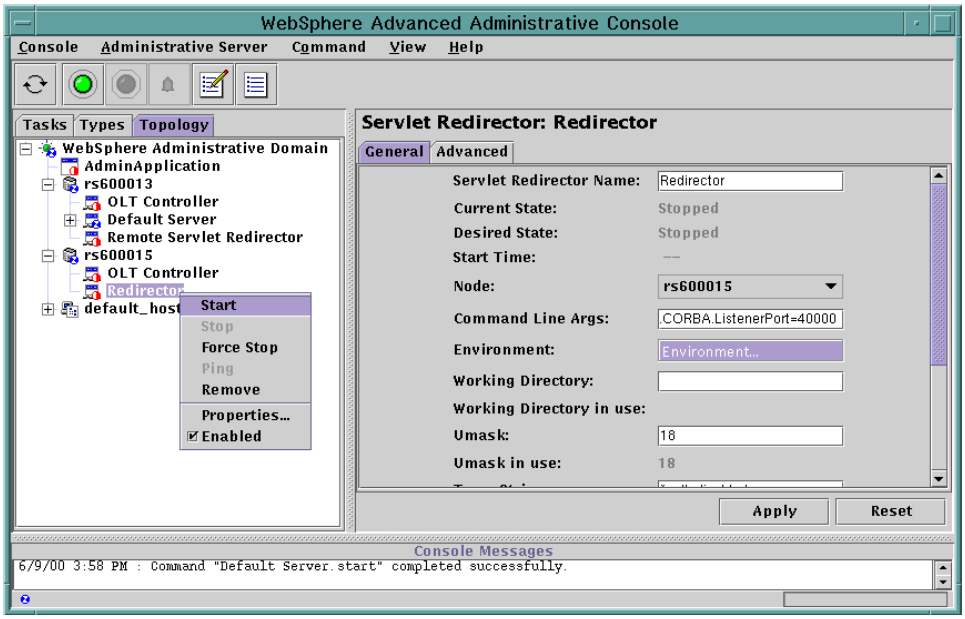

Figure 143. Start Servlet Redirector on Machine A (rs600015)

6.7.3 Verify the plug-in properties files

Verify that the plug-in properties files: rules.properties, queue.properties, and vhost.properties, in the WebSphere <was_dir>/temp directory on Machine A (rs600015) were generated correctly.

The files can reveal which requests are being handled by the Servlet Redirector (using RMI/IIOP), and which are being handled on the machine local to the Web server (using an OSE transport).

1. vhosts properties

This file is used to map the virtual hosts mentioned in a request to an actual host. If the virtual host in the request is a valid virtual host for an actual host in the administrative domain, the application server product adds the rest of the URL. It resides in the <was_dir>/temp directory.

The vhosts.properties file should contain Machine A's host name which you added as an alias. If you added Machine A's IP address as an alias, it should also be displayed. Figure 144 is an example of the vhosts.properties file.

```
#IBM WebSphere Plugin Virtual Host Mappings
#Fri Jun 09 16:02:13 EDT 2000
localhost=default_host
rs600015=default_host
rs600013=default_host
127.0.0.1=default_host
9.24.104.119=default_host
```

Figure 144. <was_dir>/temp/vhosts.properties file on Machine A (rs600015)

2. queues properties

The queues.properties file is used to find the connection parameters for the Web server queue. It also resides in the <was_dir>/temp directory.

```
#IBM WebSphere Plugin Communication Queues
#Fri Jun 09 16:02:13 EDT 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8993
ose.srvgrp.ibmoselink.clone1.type=local
```

Figure 145. <was_dir>/temp/queues.properties on Machine A (rs600015)

3. rules properties

The rules.properties file is used for looking up the URL constructed from the information in the vhost file. The application server locates the closest match to that URL. It resides in the <was_dir>/temp directory.

```
#IBM WebSphere Plugin URL Mapping Rules
#Fri Jun 09 16:02:13 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 146. <was_dir>/temp/rules.properties on Machine A (rs600015)

6.7.4 Start the Web server

Start the Web server on Machine A (rs600015). If it's already started, you need to stop and restart it.

6.8 Test the configuration

Now that the HTTP server, thick Servlet Redirector, and WebSphere are running, we need to test the servers to make sure that everything is configured properly.

Using a Web browser, access the following URI (rs600015 is the HTTP server and /webapp/examples/showCfg is the servlet):

```
http://rs600015/webapp/examples/showCfg
```

You should see a page like the one shown below in Figure 147.

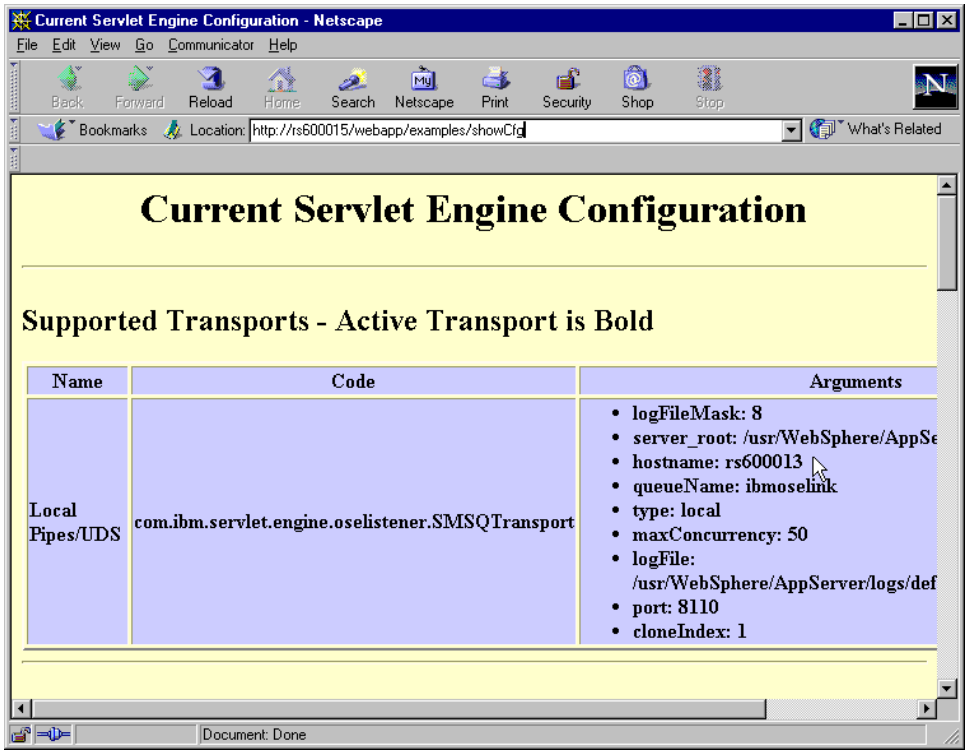

Figure 147. http://rs600015/webapp/examples/showCfg

Notice that the hostname is the name of the node running the application server (rs600013), and not the HTTP server (rs600015).

6.9 Related topologies

Thick Servlet Redirector can be used for both vertical and horizontal scaling of the WebSphere environment.

6.9.1 Variation 1: cloning an application server

WebSphere supports the use of clones and workload management with Servlet Redirector.

An HTTP server and a thick Servlet Redirector will be installed on Machine A. Two application server clones will run on Machine B. Requests will be redirected from Machine A to the application server clones running on Machine B.

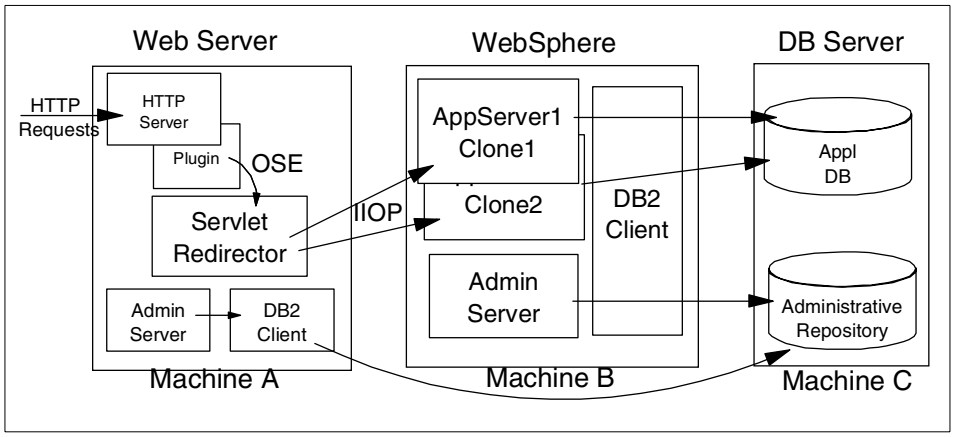

Figure 148. Thick Servlet Redirector with cloned application servers

The clones are configured using the process described in 4.4, “Create the model” on page 80. Note that each application server clone needs to be specified with a different port number for the CORBA listener in the command line args field.

Note

With V3.5, you should convert your initial Web application into a clone. If not, you can create a model but you cannot create a clone.

With V3.021, you do not have to convert your initial Web application into a clone; however since we are using Servlet Redirector we need to select the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model’s clones to fail unless this server is made into a clone.

The configuration of the HTTP server and Servlet Redirector are the same as described previously in this chapter.

6.9.2 Variation 2: cloning a remote application server

An HTTP server and a thick Servlet Redirector will be installed on Machine A. Application server clones will run on Machine B and Machine C. Requests will be redirected from Machine A to the application server clones running on Machine B and Machine C.

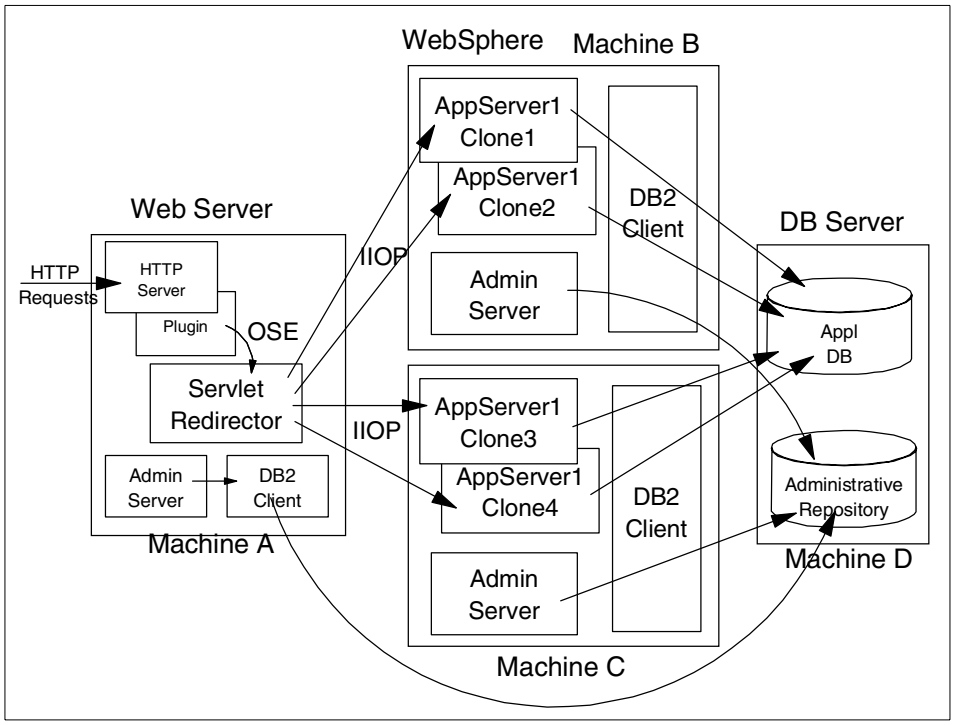

Figure 149. Cloned application servers on multiple WebSphere machines

Clones of the Default Server on separated machines

For V3.5, the default configuration automatically creates and installs the JDBC Driver (Admin DB Driver).

When creating a clone of the Default Server to a remote node, it requires the Admin DB Driver to be installed on the remote node prior to creating the clone.

To install the Admin DB Driver, right click the Admin DB Driver and select **Install**. The Install Driver window will appear. Then select the remote node and specify the jar file which contains the JDBC Driver. Then click **Install**. After you install the JDBC Driver for the Admin DB Driver on the remote node, you can create clones of the Default Server.

6.10 Maintenance/troubleshooting

There are some considerations when doing maintenance on the application server.

6.10.1 Terminating the application server

When an application server aborts suddenly (we used the `kill -9 PID` command to simulate this situation), WebSphere automatically restarts the server. No action is required.

6.10.2 Restart the application server/node

For maintenance or any other reason, you might stop and restart your application server. When an application server is restarted, browsers can automatically get access to it. No action is required.

We assume that you do not have clones.

6.10.3 Stopping an application server

When an application server is stopped from the Administrative Console, the Servlet Redirector automatically routes the request to a clone.

If the application server has no clones, no client requests will succeed. No additional actions are required.

6.10.4 Restarting a clone

When a clone is restarted, browsers will be able to access it. No action is required. If a quicker turn-around is needed, you should restart the Servlet Redirector.

6.10.5 Restarting a machine

With multiple clone/multiple node environments, when a node is restarted, the Servlet Redirector will access it. If a quicker turn-around is needed, the refresh can be forced by restarting the Servlet Redirector.

Chapter 7. Thick Servlet Redirector with Admin Server agent

With the Servlet Redirector, each WebSphere Administrative Server connects directly to the WebSphere Administrative Repository. This requires that either the database be installed locally or that the appropriate database drivers be installed and configured on the machine. More importantly from a security standpoint a database user ID and password must be stored on the machine for use by the database processes. And also the Administrative Server on the HTTP server machine requires a TCP connection to the remote database. Therefore you must open a port in the firewall to pass DB traffic. In topologies where security is a concern, such as a server running outside a firewall, this may not be acceptable.

An alternative to the normal Administrative Server is to configure an Administrative Server to run in "agent" mode. In agent mode the Administrative Server process on one machine attaches to the Administrative Server process on another machine, and uses the remote Administrative Server to connect to the WebSphere Administrative Repository. Doing so obviates the need to configure the required database components on a machine and also reduces the number of ports that must be opened through a firewall by eliminating the need to open a port or ports for database connections. This also reduces the number of processes running on a machine (for example, DB2 UDB server or a DB2 client) and helps to improve performance.

This chapter provides instructions for the administrative tasks required to configure the Servlet Redirector and remote application server machines.

We will discuss the following steps for setting up this configuration:

1. Installation summary
2. Configure the Administrative Server agent
3. Configure the Administrative Server
4. Start the Administrative Server and Console
5. Configure the application server
6. Configure the Servlet Redirector
7. Start the servers
8. Test the servers
9. Maintenance/troubleshooting
10. Related topologies

7.1 Overview of the configuration

An HTTP server and WebSphere Administrative Server will be installed on Machine A (rs600015) and a Servlet Redirector configured on that node. A WebSphere Administrative Server, application server and DB2 client will be configured on Machine B (rs600013). The DB2 UDB server running the application data and the WebSphere Administrative Repository will be configured on Machine C (rs600012).

All incoming HTTP requests will be redirected from Machine A to Machine B where the application server process is running.

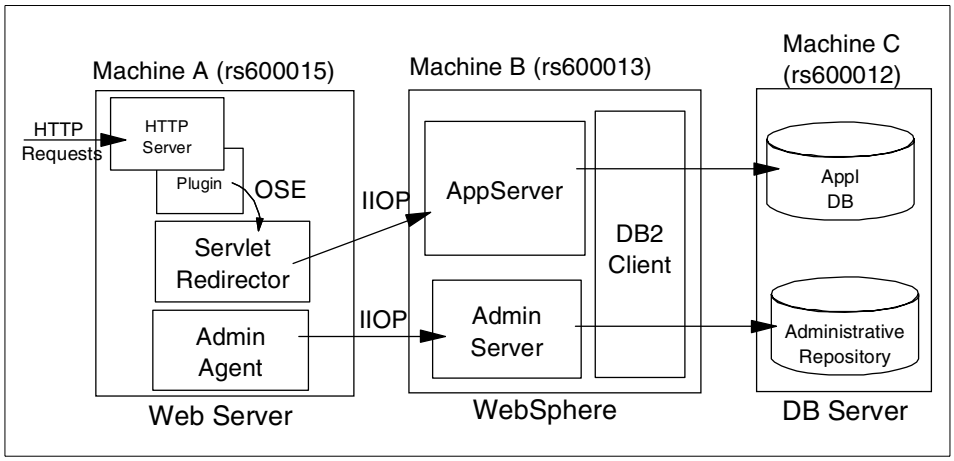

Figure 150. Thick Servlet Redirector with administrative agent

7.2 Installation summary

Table 6 is a summary of the software installation steps. We will discuss only steps 7 and 8 in this chapter. For the steps 1 to 6, please refer to the *Getting Started* documentation which is part of the WebSphere product.

Table 6. Software installation steps

Step No.	MachineA (HTTP)	MachineB (WAS)	MachineC (DB)
1			Install DB2 UDB Apply FixPack
2			Create DB

Step No.	MachineA (HTTP)	MachineB (WAS)	MachineC (DB)
3		Install DB2 client Apply FixPack	
4		Catalog node Catalog db	
5	Install JDK (V3.02x)	Install JDK (V3.02x)	
6	Install Web server		
7	Install WebSphere (Production Application Server; -Plugins, -Core Server)		
8		Install WebSphere (Production Application Server)	

7.2.1 Install WebSphere on the Web server machine

On Machine A, you must install the HTTP server (installation Step 6). Then, you need to install the WebSphere Administrative Server and the Web server plug-in (installation Step 7).

To install both of them, you need to select **Custom Installation** and select **Production Application Server** (**Core Server** and appropriate plug-in for your Web server). If you need to install other components, you should select them. For the production environment, you should not need to select anything other than **Production Application Server** (**Core Server** and plug-in).

7.2.2 Install WebSphere on the application server machine

On Machine B, where the application server resides, a DB2 client is required (installation Step 3).

Then, you need to install WebSphere (installation Step 8).

To install it, you need to select **Custom Installation** and select **Production Application Server** (**Core Server**). If you need to install other components, you should select them. For the production environment, you should not need to select anything other than the **Production Application Server** (**Core Server**). For our example, we installed another component for demonstration purposes.

7.3 Configure the Administrative Server agent

Install WebSphere on the HTTP server machine. **Do not** create a default configuration in this case. Once WebSphere is installed you will need to edit the <was_dir>/bin/admin.config file. Prior to modification the admin.config file should appear as below.

```
#Admin Config Rewritten to disable Initial Config
#Thu Jun 08 16:56:09 EDT 2000
com.ibm.ejs.sm.adminServer.qualifyHomeName=true
com.ibm.ejs.sm.util.process.Nanny.maxtries=3
com.ibm.ejs.sm.adminServer.dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
com.ibm.ejs.sm.adminServer.disableEPM=true
com.ibm.ejs.sm.adminServer.dbUser=db2inst1
com.ibm.ejs.sm.adminserver.classpath=/usr/WebSphere/AppServer/lib/ibmwe
bas.jar:/usr/WebSphere/AppServer/properties:.....
....
com.ibm.ejs.sm.adminServer.logFile=/usr/WebSphere/AppServer/tranlog/rs6
00013_tranlog1,/usr/WebSphere/AppServer/tranlog/rs600013_tranlog2
com.ibm.ejs.sm.adminServer.traceFile=/usr/WebSphere/AppServer/logs/trac
efile
com.ibm.ejs.sm.adminServer.dbUrl=jdbc:db2:was
server.root=/usr/WebSphere/AppServer
com.ibm.ejs.sm.util.process.Nanny.traceFile=/usr/WebSphere/AppServer/lo
gs/nanny.trace
com.ibm.ws.jdk.path=/usr/jdk_base/
com.ibm.ejs.sm.adminServer.jarFile=/usr/WebSphere/AppServer/lib/reposit
ory.jar,/usr/WebSphere/AppServer/lib/tasks.jar
com.ibm.ejs.sm.adminServer.dbPassword=db2inst1
com.ibm.ejs.sm.adminServer.nameServiceJar=/usr/WebSphere/AppServer/lib/
ns.jar
install.initial.config=false
com.ibm.ejs.sm.adminServer.initializer=com.ibm.ejs.security.Initializer
,com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer,com.ibm.servl
et.config.InitialSetupInitializer
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
-Dcom.ibm.CORBA.ListenerPort=33000
com.ibm.ejs.sm.util.process.Nanny.path=/usr/WebSphere/AppServer/bin:/us
r/WebSphere/AppServer/lib/odbc/lib:/home/db2inst1/sqllib/bin:/home/db2i
nst1/sqllib/function:/usr/jdk_base//bin
com.ibm.CORBA.ConfigURL=file:/usr/WebSphere/AppServer/properties/sas.se
rver.props
```

Figure 151. <was_dir>/bin/admin.config for the Administrative Server on Machine B

You will need to delete or comment out the following entries:

- com.ibm.ejs.sm.adminServer.nameServiceJar
- com.ibm.ejs.sm.adminServer.dbUser
- com.ibm.ejs.sm.adminServer.dbUrl
- com.ibm.ejs.sm.adminServer.dbPassword

These are the entries for the name service jar file and for the database server, respectively. (Note you may wish to save the original file as a backup, prior to making your modifications.)

Note

In V3.02, the agent node is implicitly turned on by modifying the properties, such as nameServiceJar, dbUser, dbUrl and dbPassword in admin.config. In V3.5, those properties will be ignored when the agent mode is on (com.ibm.ejs.sm.adminServer.agentMode). Therefore, you don't need to delete these entories. So it is much easier to switch the admin server working mode. We will explain the parameter agentMode later.

You will then need to add the following entry:

- `com.ibm.ejs.sm.adminServer.bootstrapHost`
- `com.ibm.ejs.sm.adminServer.lsdPort` property (default is 9000)
- `com.ibm.ejs.sm.adminServer.bootstrapPort` property (default is 900)
- `com.ibm.ejs.sm.adminServer.primaryNode`
- `com.ibm.ejs.sm.adminServer.agentMode`

The value of bootstrapHost property will need to be the name of the node that is running the Administrative Server. In our case this is the application server machine, rs600013, so our entry is:

```
com.ibm.ejs.sm.adminServer.bootstrapHost=rs600013
```

The lsdPort and bootstrapPort entries specify the Location Service Daemon (LSD) and bootstrap port that the adminserver is to run on. In cases where you are configuring a firewall and want to use other than the default port assignments for security reasons you would specify these entries in the admin.config file for *both* the "regular" Administrative Server and the Administrative Server running in agent mode. (In our example these are rs600013 and rs600015 respectively.)

If the agent is running under the same node as primary node, the bootstrapHost and bootstrapPort properties are optional when the primary node uses the default value .

If the agent is running under different node, and primary node uses default bootstrap and LSD port, the bootstrapPort and lsdPort for agent is optional, the bootstrapHost is required.

The value assigned to the primaryNode should be the name of the node with a full fledged Administrative Server (this would be rs600013 in our example). If running on the same host as the primary node, it is optional. The agent will running under ${primaryNodeName}_agent name space to avoid collision. It is only recommended for testing purpose, and you can only starts one agent on the same host as primary node.

The primaryNode name has to be the name of the Administrative Server, not necessary to be the name of the host Administrative Server is on. In default, Administrative Server uses the short name of the host name. If com.ibm.ejs.sm.adminServer.nodeName property is set to be the name other than the short name of the host name, the primaryNode needs to be set to that name rather than the short host name.

If you are using V3.5 of WebSphere, you need to specify the agentMode property. In other word, with V3.02x, you don't need to specify this property. In our case with V3.5 of WebSphere, we specified as follows.

```
com.ibm.ejs.sm.adminServer.agentMode=true
```

In V3.5, as long as the property is present, the AdminServer will work in agent mode. So, you don't have to specify agentMode=true. But for the practice, we recommend to specify "true".

The modified admin.config file for V3.021 is depicted in Figure 152.

```
#Admin Config Rewritten to disable Initial Config
#Tue Jun 06 13:41:41 EDT 2000
com.ibm.ejs.sm.adminServer.qualifyHomeName=true
com.ibm.ejs.sm.util.process.Nanny.maxtries=3
com.ibm.ejs.sm.adminServer.disableEPM=true
com.ibm.ejs.sm.adminServer.dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
;com.ibm.ejs.sm.adminServer.dbUser=db2inst1
com.ibm.ejs.sm.adminserver.classpath=/usr/WebSphere/AppServer/lib/secur
ity0505.jar:.....
......
com.ibm.ejs.sm.adminServer.logFile=/usr/WebSphere/AppServer/tranlog/rs6
00015_tranlog1,/usr/WebSphere/AppServer/tranlog/rs600015_tranlog2
com.ibm.ejs.sm.adminServer.traceFile=/usr/WebSphere/AppServer/logs/trac
efile
;com.ibm.ejs.sm.adminServer.dbUrl=jdbc:db2:was
server.root=/usr/WebSphere/AppServer
com.ibm.ejs.sm.util.process.Nanny.traceFile=/usr/WebSphere/AppServer/lo
gs/nanny.trace
com.ibm.ejs.sm.adminServer.bootstrapHost=rs600013
com.ibm.ejs.sm.adminServer.bootstrapPort=900
com.ibm.ejs.sm.adminServer.lsdPort=9000
com.ibm.ejs.sm.adminServer.primaryNode=rs600013
com.ibm.ws.jdk.path=/usr/jdk_base/
;com.ibm.ejs.sm.adminServer.dbPassword=db2inst1
com.ibm.ejs.sm.adminServer.jarFile=/usr/WebSphere/AppServer/lib/reposit
ory.jar,/usr/WebSphere/AppServer/lib/tasks.jar
;com.ibm.ejs.sm.adminServer.nameServiceJar=/usr/WebSphere/AppServer/lib
/ns.jar
install.initial.config=false
com.ibm.ejs.sm.adminServer.initializer=com.ibm.ejs.security.Initializer
,com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer,com.ibm.servl
et.config.InitialSetupInitializer
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
-Dcom.ibm.CORBA.ListenerPort=33000
com.ibm.ejs.sm.util.process.Nanny.path=/usr/WebSphere/AppServer/bin:/us
r/WebSphere/AppServer/lib/odbc/lib:/home/db2inst1/sqllib/bin:/home/db2i
nst1/sqllib/function:/usr/jdk_base//bin
com.ibm.CORBA.ConfigURL=file:/usr/WebSphere/AppServer/properties/sas.se
rver.props
```

Figure 152. <was_dir>/bin/admin.config for the Administrative Server agent on Machine A

7.4 Configure the Administrative Server CORBA listener port

Before you start the Administrative Server on Machines A and B, we recommend you specify the CORBA listener port for the Administrative Server. Since Servlet Redirector uses IIOP protocol to communicate with WebSphere Administrative Server, you can not specify port numbers without configuring CORBA listener port if you have a firewall between Web server (Servlet Redirector) and WebSphere Administrative Server. To do so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the line com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs in the <was_dir>/bin/admin.config file as shown in Figure 153. In our example, we specify “33000” for the CORBA listener port. The line has been divided with line break for easier reading. In your actual admin.config file, ensure that the line is on a single line. In other words, combine all of the lines from “com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs” to “ListenerPort=33000” on one line, each separated by a space.

```
#Admin Config Rewritten to disable Initial Config
#Tue Jun 06 13:41:41 EDT 2000
com.ibm.ejs.sm.adminServer.qualifyHomeName=true
com.ibm.ejs.sm.util.process.Nanny.maxtries=3
com.ibm.ejs.sm.adminServer.dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
com.ibm.ejs.sm.adminServer.disableEPM=true
.....
.....
com.ibm.ejs.sm.adminServer.initializer=com.ibm.ejs.security.Initializer
,com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer,com.ibm.servl
et.config.InitialSetupInitializer
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
-Dcom.ibm.CORBA.ListenerPort=33000
com.ibm.ejs.sm.util.process.Nanny.path=/usr/WebSphere/AppServer/bin:/us
r/WebSphere/AppServer/lib/odbc/lib:/home/db2inst1/sqllib/bin:/home/db2i
nst1/sqllib/function:/usr/jdk_base//bin
com.ibm.CORBA.ConfigURL=file:/usr/WebSphere/AppServer/properties/sas.se
rver.props
```

Figure 153. CORBA listener port for the Administrative Server on Machines A and B

7.5 Start the Administrative Server and Console

You need to start the Administrative Server and Console:

1. Start a DB2 instance for the Administrative Repository on Machine C (rs600012) if it is not started
2. Start Administrative Server on Machine B (rs600013).

   In our example, we specified “install.initial.config=true” in the admin.config file to install the Default Server.
3. Start Administrative Server agent on Machine A (rs600015).
4. Start Administrative Console on Machine B (rs600013).

Note

You can use any machine for your Administrative Console. However, an Administrative Console cannot "attach" to an Administrative Server running in agent mode.

This means that in our example, "adminclient.sh rs600015 900" from the command line will *not* work, but “adminclient.sh rs600013 900" will work.

5. When the console finishes initializing, make sure you see both Machine A and Machine B in the Topology page as shown in Figure 154.

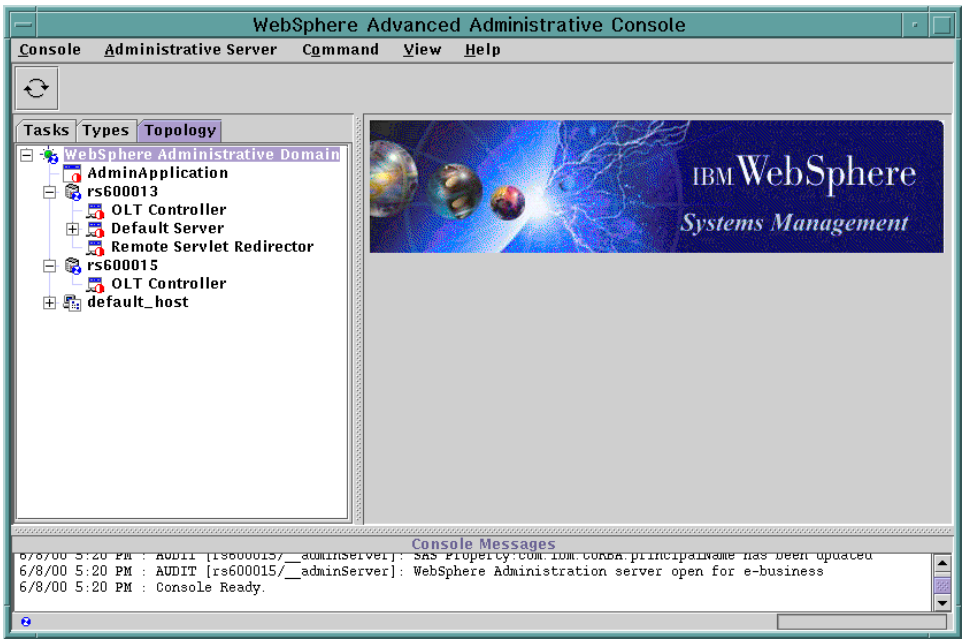

Figure 154. WebSphere Advanced Administrative Console

7.5.1 Adding host alias for Machine A

Add aliases for Machine A. These aliases are needed so that the virtual host will accept requests redirected from the Servlet Redirector on Machine A with Machine A's URI instead of Machine B's URI. You may add Machine A's hostname (rs600015), IP address, and fully qualified name.

To add the host aliases:

1. Click the **Topology** tab on the Administrative Console.
2. Click **default_host** in the Topology tree.
3. Click the **Advanced** tab in the Virtual Host pane. Enter the hostname (also IP address and FQDN if you want) for the HTTP server machine in the Host Aliases and click the **Apply** button as shown in Figure 155.

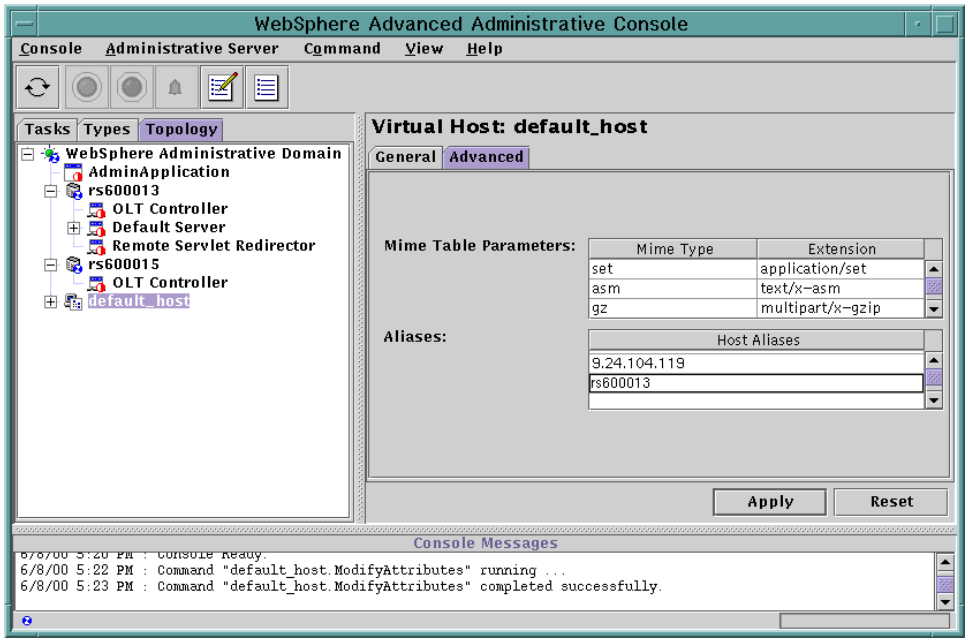

Figure 155. Add aliases

The host aliases have been added.

7.6 Adding the CORBA listener port for the application server

We recommend you specify the CORBA listener port for the application server. Since Servlet Redirector uses IIOP protocol to communicate with application server, you can not specify port numbers without configuring

CORBA Listener Port if you have a firewall between Web server (Servlet Redirector) and Application Server. To do so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the command line arguments in the General tab of the application server as shown in Figure 156. In our example, we specify 35000. Please choose a port that is not already in use.

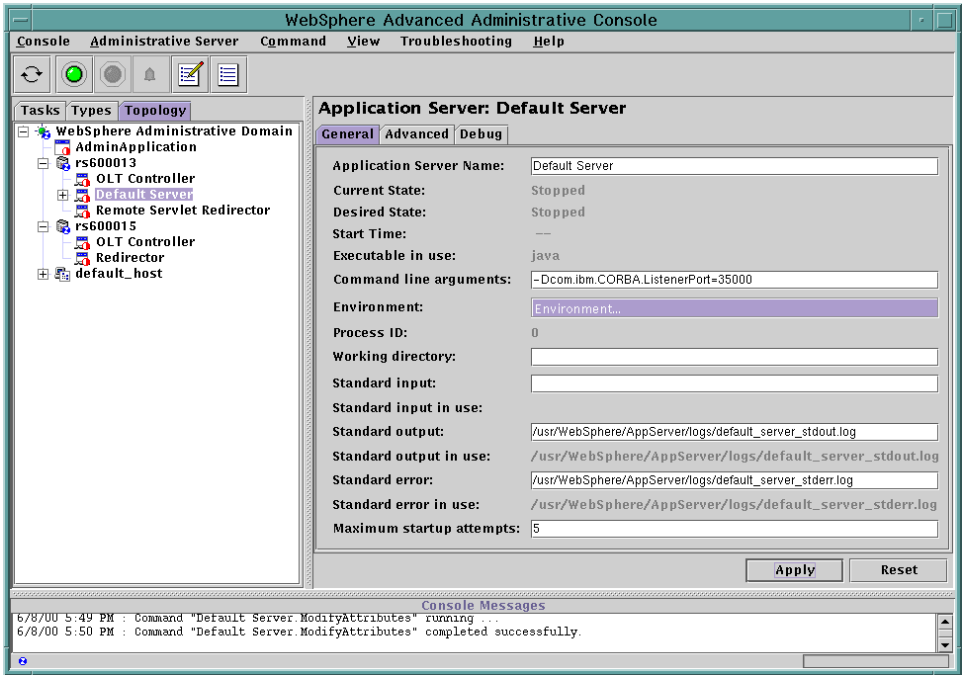

Figure 156. CORBA listener port for an application server

If you need to create clones, do so now. Note that each application server clone on the same machine has to be specified with a different port number for the -Dcom.ibm.CORBA.ListernerPort argument. If clones have been specified with the same port number on the same machine, you cannot start the clone. But, you can specify the same number for a clone which runs on a different machine.

7.7 Configure and enable Servlet Redirector

You need to create the Servlet Redirector on Machine A (rs600015) following the instructions below.

From the Topology tab in the Administrative Console, expand the tree under Machine A (rs600015).

You might see a Remote Servlet Redirector, which is created automatically during WebSphere installation if you installed components other than those that we installed on Machine A (rs600015). If you see one, you just need to update its properties. We will not discuss this but you are able to get the idea from the following sections.

7.7.1 Create the Servlet Redirector

1. From the Topology tab, select a node for the Servlet Redirector. For our example, we selected rs600015 (Machine A).
2. Right-click the node (rs600015) and select **Create..** and then click **Servlet Redirector** as shown in Figure 157.

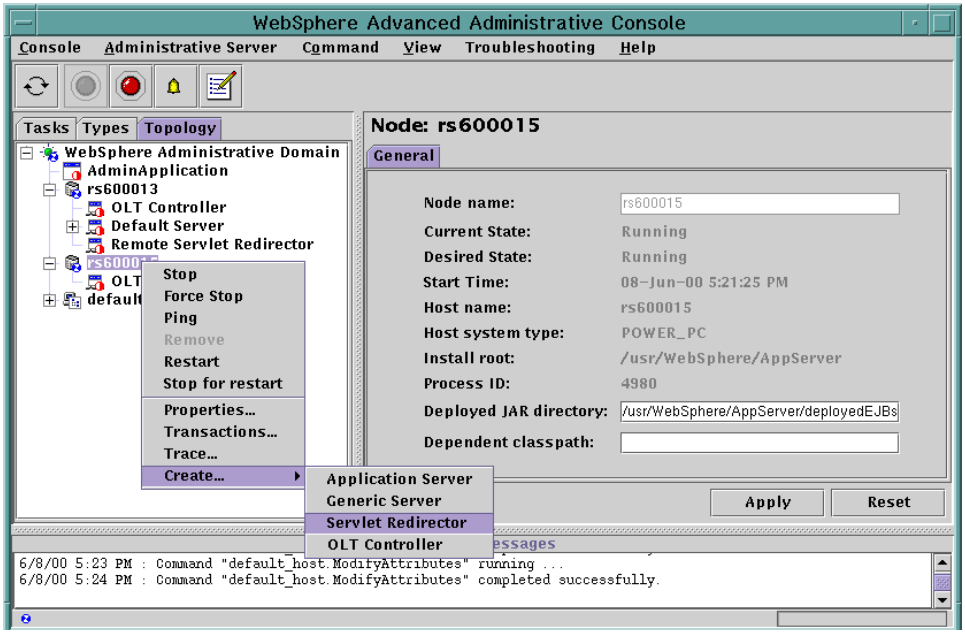

Figure 157. Create Servlet Redirector

3. Then the Create Servlet Redirector dialog comes up. Enter a name in the Servlet Redirector Name field. In our example, we put the name “Redirector” as the Servlet Redirector on Machine A (rs600015). In the Create Servlet Redirector dialog, choose the node name for Machine A (rs600015) from the drop down list and specify the -Dcom.ibm.CORBA.ListenerPort (in our example, we specified -Dcom.ibm.CORBA.ListenerPort=40000) in the Command Line Args field as shown in Figure 158. Then click the **Create** button. Please choose a port that is not already in use.

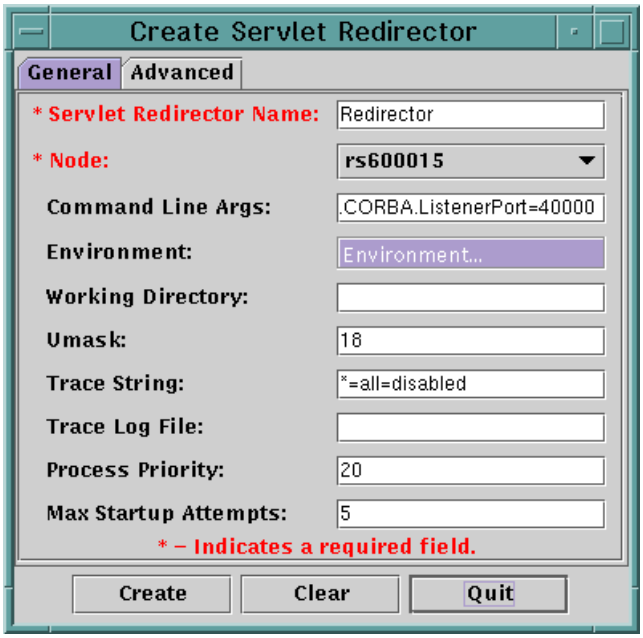

Figure 158. Create Servlet Redirector: General tab

7.7.2 Enable remote Servlet Redirector

You need to enable the Servlet Redirector in order to route requests to nodes other than the node that contains the Web server.

1. Click the **Topology** tab to display the Topology tree.
2. In the tree, find and right-click the **Servlet Redirector** (**Redirector**, in our example) on the node rs600015.
3. On the resulting menu, select and click the **Enabled** menu as shown in Figure 159 on page 192.

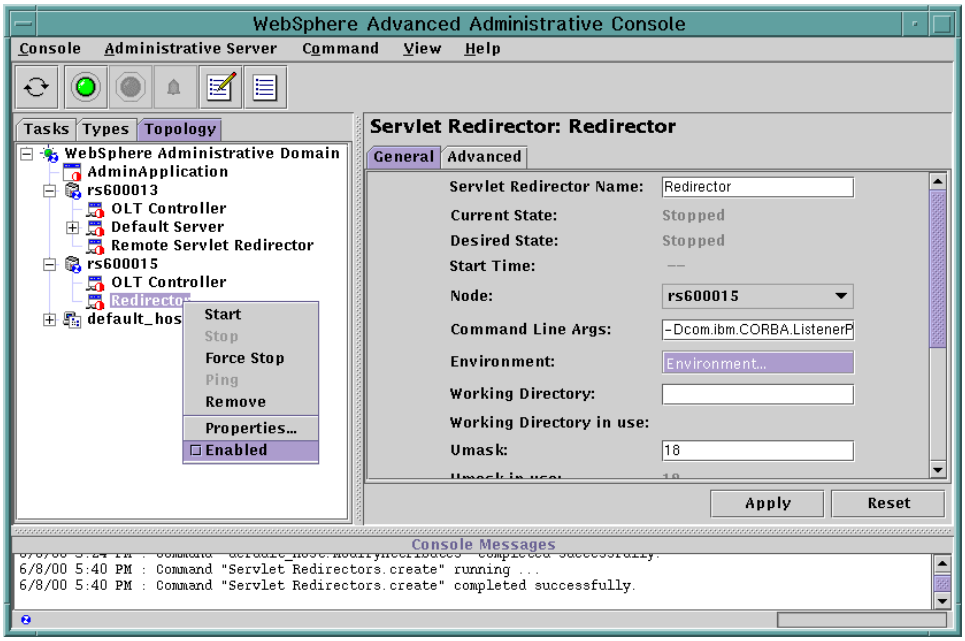

Figure 159. Enable Servlet Redirector on Machine A (rs600015)

4. Right-click the **Servlet Redirector** (**Redirector**). Make sure the **Enabled** menu item is checked.

7.8 Start the servers

Now you are ready to start the application server, Servlet Redirector and Web server.

7.8.1 Start the application server

Start the application server (in our example, Default Server) if it is not running.

If the application server has been running, stop and start it again.

7.8.2 Start the Servlet Redirector

Now you can start the Servlet Redirector.

To do so, select **Start** to start the Servlet Redirector as shown in Figure 160.

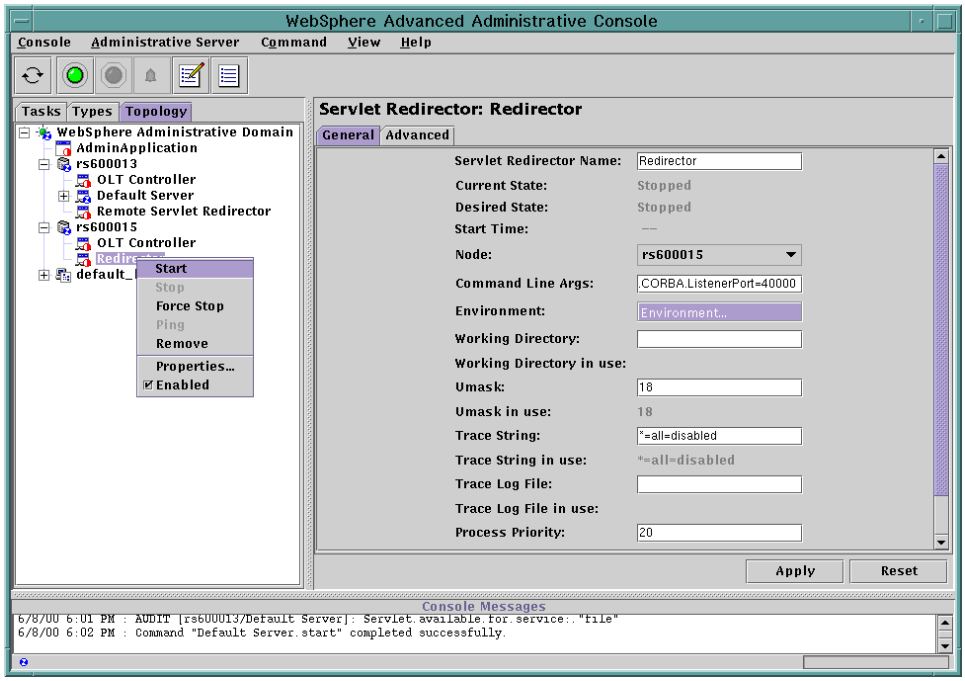

Figure 160. Start Servlet Redirector on Machine A (rs600015)

7.8.3 Verify the plug-in properties files

Verify that the plug-in properties files: rules.properties, queue.properties, and vhosts.properties, in the WebSphere <was_dir>/temp directory on Machine A (rs600015) were generated correctly.

The files can reveal which requests are being handled by the Servlet Redirector (using RMI/IIOP), and which are being handled on the machine local to the Web server (using an OSE transport).

1. vhosts properties

This file is used to map the virtual hosts mentioned in a request to an actual host. If the virtual host in the request is a valid virtual host for an actual host in the administrative domain, the Application Server product adds the rest of the URL. It resides in the <was_dir>/temp directory.

The vhosts.properties file should contain Machine A's host name which you added as an alias. If you added Machine A's IP address as an alias, it should also be displayed. Figure 161 on page 194 is an example of the vhosts.properties file.

```
#IBM WebSphere Plugin Virtual Host Mappings
#Fri Jun 09 16:02:13 EDT 2000
localhost=default_host
rs600015=default_host
rs600013=default_host
127.0.0.1=default_host
9.24.104.119=default_host
```

Figure 161. <was_dir>/temp/vhosts.properties file on Machine A (rs600015)

2. queues properties

The queues.properties file is used to find the connection parameters for the Web server queue. It also resides in the <was_dir>/temp directory.

```
#IBM WebSphere Plugin Communication Queues
#Fri Jun 09 16:02:13 EDT 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8993
ose.srvgrp.ibmoselink.clone1.type=local
```

Figure 162. <was_dir>/temp/queues.properties on Machine A (rs600015)

3. rules properties

The rules.properties file is used for looking up the URL constructed from the information in the vhost file. Application server locates the closest match to that URL. It resides in the <was_dir>/temp directory.

```
#IBM WebSphere Plugin URL Mapping Rules
#Fri Jun 09 16:02:13 EDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 163. <was_dir>/temp/rules.properties on Machine A (rs600015)

7.8.4 Start the Web server

Start the Web server on Machine A (rs600015). If it's already started, you need to stop and restart it.

7.9 Test the configuration

Now that the HTTP server, thick Servlet Redirector, and WebSphere are running, we need to test the servers to make sure that everything is configured properly.

Using a Web browser, access the following URI (rs600015 is the HTTP server and /webapp/examples/showCfg is the servlet):

```
http://rs600015/webapp/examples/showCfg
```

You should see a page like the one shown below in Figure 164.

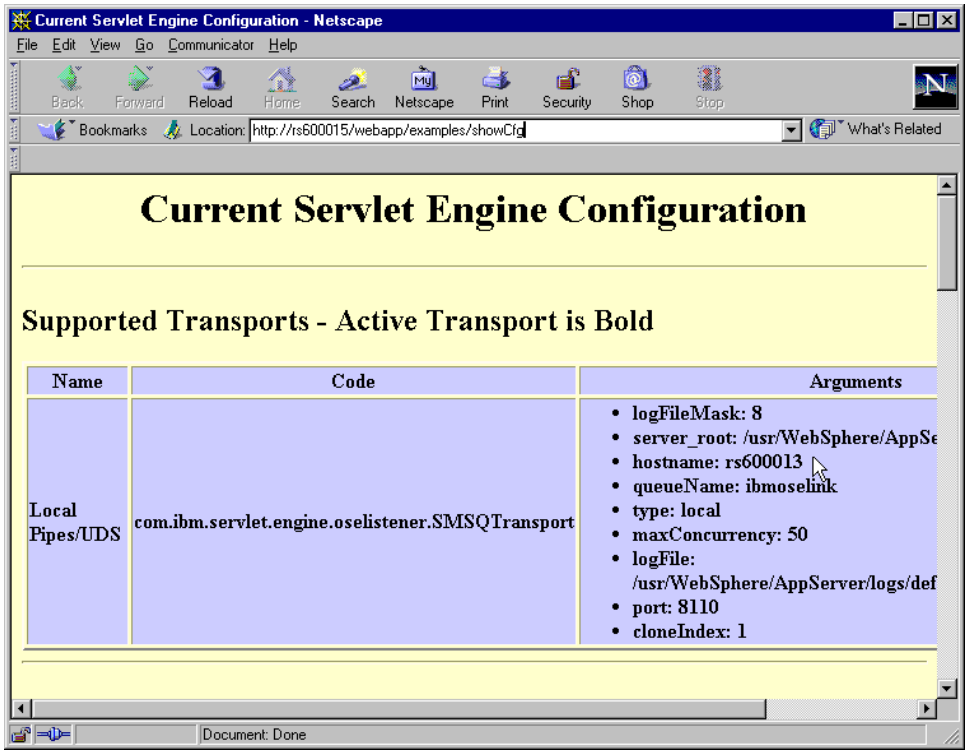

Figure 164. http://rs600015/webapp/examples/showCfg

Notice that the hostname is the name of the node running the application server (rs600013), and not the HTTP server (rs600015).

7.10 Related topologies

Thick Servlet Redirector with an Administrative Server agent can be used for both vertical and horizontal scaling of the WebSphere environment.

7.10.1 Variation 1: cloning an application server

WebSphere supports the use of clones and workload management with Servlet Redirector.

An HTTP Server and a thick Servlet Redirector will be installed on Machine A. Two application server clones will run on Machine B. Requests will be redirected from Machine A to the application server clones running on Machine B.

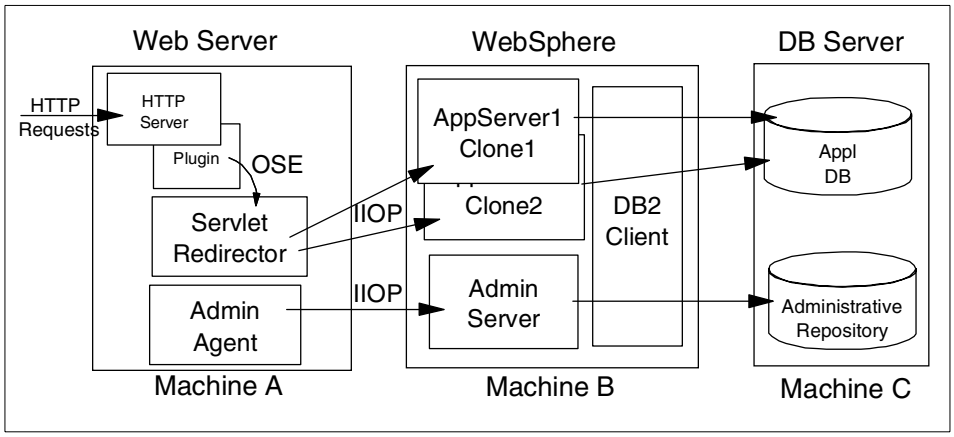

Figure 165. Thick Servlet Redirector with cloned application servers

The clones are configured using the process described in 4.4, “Create the model” on page 80. Note that each application server clone needs to be specified with a different port number for the CORBA listener in the Command Line Args field.

Note

With V3.5, you should convert your initial Web application into a clone. If not, you can create a model but you cannot create a clone.

With V3.021, you do not have to convert your initial Web application into a clone; however since we are using Servlet Redirector we need to select the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model’s clones to fail unless this server is made into a clone.

The configuration of the HTTP server and Servlet Redirector are the same as described previously in this chapter.

WebSphere 3.021 requires a patch for this configuration.

7.10.2 Variation 2: Cloning a remote application server

An HTTP server and a thick Servlet Redirector will be installed on Machine A. Application server clones will run on Machine B and Machine C. Requests will be redirected from Machine A to the application server clones running on Machine B and Machine C.

WebSphere 3.021 requires a patch for this configuration.

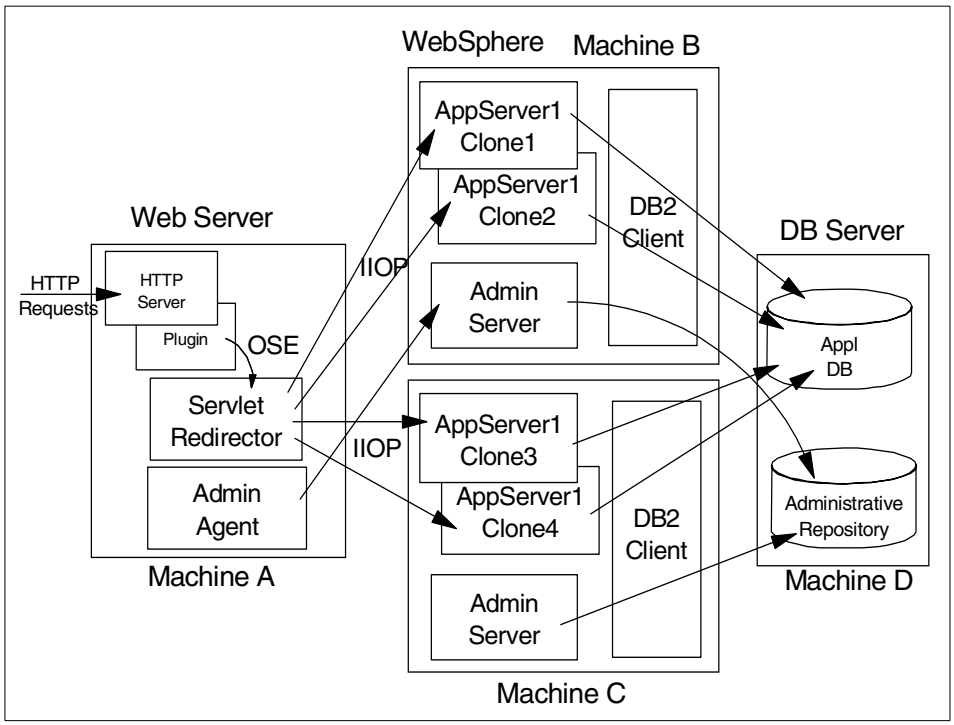

Figure 166. Cloned application servers on multiple WebSphere machines

Clones of the Default Server on separated machines

For V3.5, the default configuration automatically creates and installs the JDBC Driver (Admin DB Driver).

When creating a clone of the Default Server to a remote node, it requires the Admin DB Driver to be installed on the remote node prior to creating the clone.

To install the Admin DB Driver, right click the Admin DB Driver and select **Install**. The Install Driver window will appear. Then select the remote node and specify the jar file which contains the JDBC Driver. Then click **Install**. After you install the JDBC Driver for the Admin DB Driver on the remote node, you can create clones of the Default Server.

7.11 Maintenance/troubleshooting

There are some considerations when doing maintenance on the application server.

7.11.1 Terminating the application server

When an application server aborts suddenly (we used the `kill -9 PID` command to simulate this situation), WebSphere automatically restarts the server. No action is required.

Servlet Redirector will automatically route the request to alternative servers.

If there are no clones of this server, no requests will be served and errors will be reported until the application server restarts.

7.11.2 Restart the application server/node

For maintenance or any other reason, you might stop and restart your application server.

When an application server is restarted, browsers can automatically get access to it. No action is required.

We assume that you do not have clones.

7.11.3 Stopping an application server

When an application server is stopped from the Administrative Console, the routing of requests will continue on to another clones. Of course, if the application server has no clones, no client requests will succeed. No additional actions are required.

7.11.4 Restarting a clone

When a clone is restarted, browsers will be able to access it. No action is required. If a quicker turn-around is needed, you should restart the Servlet Redirector.

7.11.5 Restarting a machine

With multiple clone/multiple node environments, when a node is restarted, the Servlet Redirector will access it.

If a quicker turn-around is needed, the refresh can be forced by restarting the Servlet Redirector.

Chapter 8. Standalone (thin) Servlet Redirector

Normally an Administrative Server will reside on the same machine as the HTTP server. This requires that the user ID and password for the database repository also be stored on that machine. In addition, the HTTP plug-in normally uses the OSE transport to direct requests to an application server. OSE does not support encryption. In topologies where security is a concern, using OSE Remote may not be acceptable.

The Standalone Servlet Redirector allows us to send the requests over IIOP (which supports encryption) without having direct access to the Administrative Server. The Redirector uses pre-generated files to route the requests to the application server. Hence, if the application server is changed in any way, such as adding a Web application, servlet, or EJB, the pre-generated files must be regenerated. The HTTP server and Redirector will deal with the changes automatically without restarting.

This chapter describes how to install and configure a standalone Servlet Redirector based on eXtensible Markup Language (XML).

We will discuss the following steps for setting up this configuration:

1. Installation summary
2. Configure the application server
3. Configure the Servlet Redirector
4. Generate the HTTP server plug-in configuration files
5. Start the servers
6. Test the servers
7. Maintenance/troubleshooting
8. Related topologies

8.1 Overview of the configuration

In the configuration described below, you want to install a Web server and a standalone Servlet Redirector on Machine A. On Machine B, you want to install a WebSphere Administrative Server and application server processes. On Machine A, you redirect requests from a Web browser to the application server in Machine B for processing.

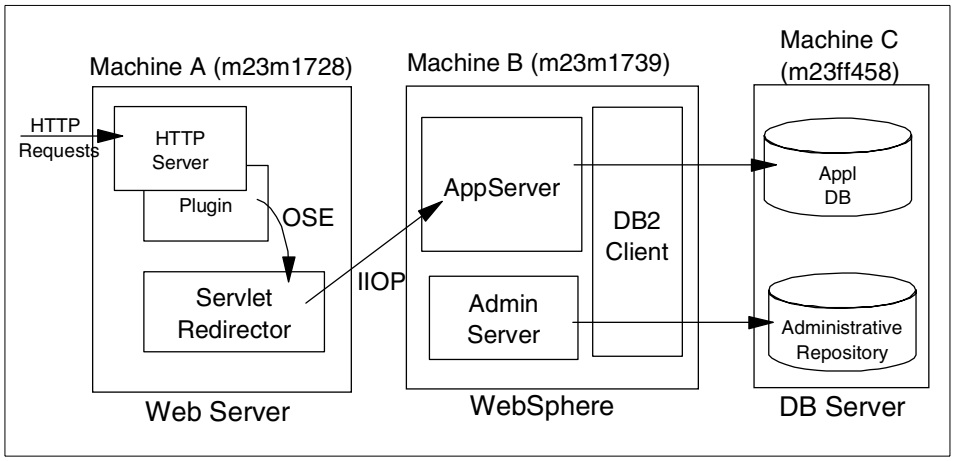

Figure 167. Standalone Servlet Redirector

8.2 Installation summary

Table 7 is a summary of the software installation steps. We will discuss only steps 7 and 8 in this chapter. For steps 1 to 6, please refer to the *Getting Started* documentation which is part of the WebSphere product.

Table 7. Software installation steps

Step No.	Machine A (SR)	Machine B (WAS)	Machine C (DB)
1			Install DB2 UDB Apply FixPack
2			Create DB
3		Install DB2 Client Apply FixPack	
4		Catalog node Catalog db	
5	Install JDK (V3.02x)	Install JDK (V3.02x)	
6	Install Web server		
7	Install WebSphere (Production Application Serve: -Plugins, -Core Server)		

Step No.	Machine A (SR)	Machine B (WAS)	Machine C (DB)
8		Install WebSphere (Production Application Server)	

8.2.1 Install WebSphere on the machine containing the Web server

You need to install the Administrative Server and the Web server plug-in (installation Step 7).

To install both of them, you need to select **Custom Installation** and select **Production Application Server** (**Core Server** and appropriate plug-in for your Web server). If you need to install other components, you should select them. For the production environment, you should not need to select anything other than **Production Application Server** (**Core Server** and plug-in).

8.2.2 Install WebSphere where the application server resides

On Machine B, where the application server resides, you must also have a DB2 client (installation Step 3).

Then, you need to install WebSphere (installation Step 8).

To install it, you need to select **Custom Installation** and then select **Production Application Server** (**Core Server**). If you need to install other components, you should select them. For the production environment, you should not need to select anything other than **Production Application Server** (**Core Server**).

8.3 Configure the Administrative Server CORBA listener port

Before you start the Administrative Server, we recommend you specify the CORBA listener port for the Administrative Server. Since Servlet Redirector uses IIOP protocol to communicate with WebSphere Administrative Server, you can not specify port numbers without configuring CORBA listener port if you have a firewall between Web server (Servlet Redirector) and WebSphere Administrative Server. To do so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the line com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs in the <was_dir>/bin/admin.config file as shown in Figure 168 on page 204. In our example, we specify “33000” for the CORBA listener port. The line has been divided with line breaks for easier reading. In your actual admin.config file, ensure that the line is on a single line. In other words, combine all of the lines

from “com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs” to “ListenerPort=33000” on one line, each separated by a space.

```
#Admin Config Rewritten to disable Initial Config
#Tue Jun 06 13:41:41 EDT 2000
com.ibm.ejs.sm.adminServer.qualifyHomeName=true
com.ibm.ejs.sm.util.process.Nanny.maxtries=3
com.ibm.ejs.sm.adminServer.dbDriver=COM.ibm.db2.jdbc.app.DB2Driver
com.ibm.ejs.sm.adminServer.disableEPM=true
com.ibm.ejs.sm.adminServer.dbUser=db2admin
.....
.....
com.ibm.ejs.sm.adminServer.initializer=com.ibm.ejs.security.Initializer
,com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer,com.ibm.servl
et.config.InitialSetupInitializer
com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs=-mx128m
-Dcom.ibm.CORBA.ListenerPort=33000
```

Figure 168. CORBA listener port for the Administrative Server (Machine B)

8.4 Start the Administrative Server and Console

You need to start the Administrative Server and Console.

1. Start a DB2 instance for the Administrative Repository on Machine C (in our case, m23ff458) if it is not started.
2. Start the WebSphere Administrative Server on Machine B (in our case, m23m1739).

   In our example, we specified “install.initial.config=true” in the admin.config file to install the Default Server.
3. Start the Administrative Console on Machine B.

   Note: You can use any machine for your Administrative Console.

   When the console finishes initializing, make sure you see Machine B in the Topology tab as shown in Figure 169.

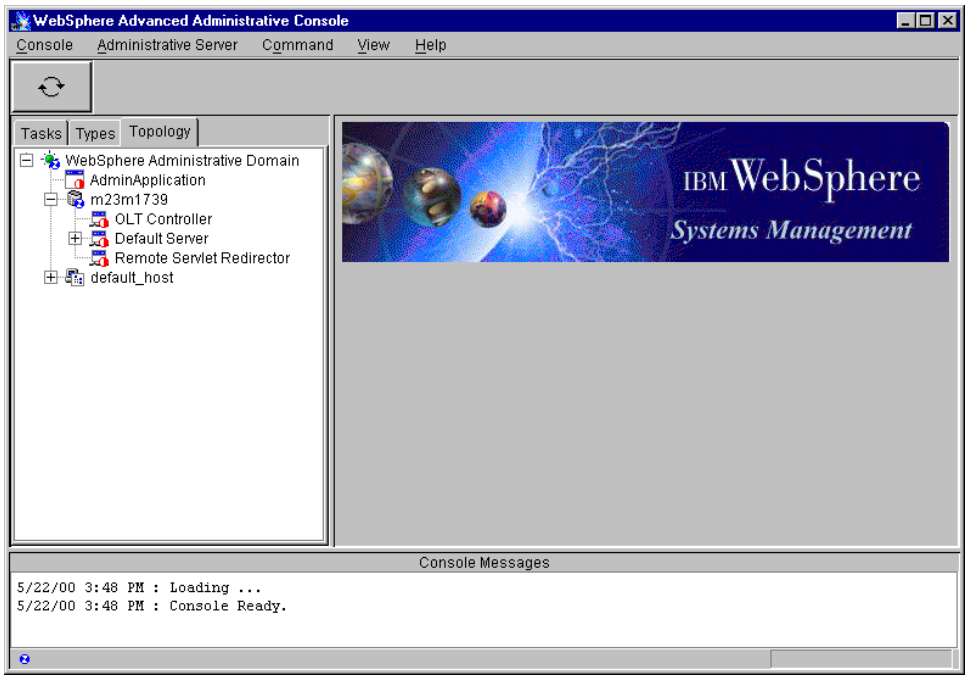

Figure 169. WebSphere Advanced Administrative Console

8.5 Configure the application server

Before you start configuring the thin Servlet Redirector, you need to add host aliases and start an application server.

8.5.1 Add host aliases for the HTTP server

The virtual host uses these aliases to accept requests redirected from the plug-in on Machine A with Machine A's URI. You may add Machine A's hostname, IP address, and fully qualified name.

To add the host aliases:

a. Click the **Topology** tab on the Administrative Console.

b. Select **default_host** in the Topology tab.

c. Click the **Advanced** tab in the Virtual Host pane. Enter the hostname and IP address for the HTTP server machine as shown in Figure 170 on page 206.

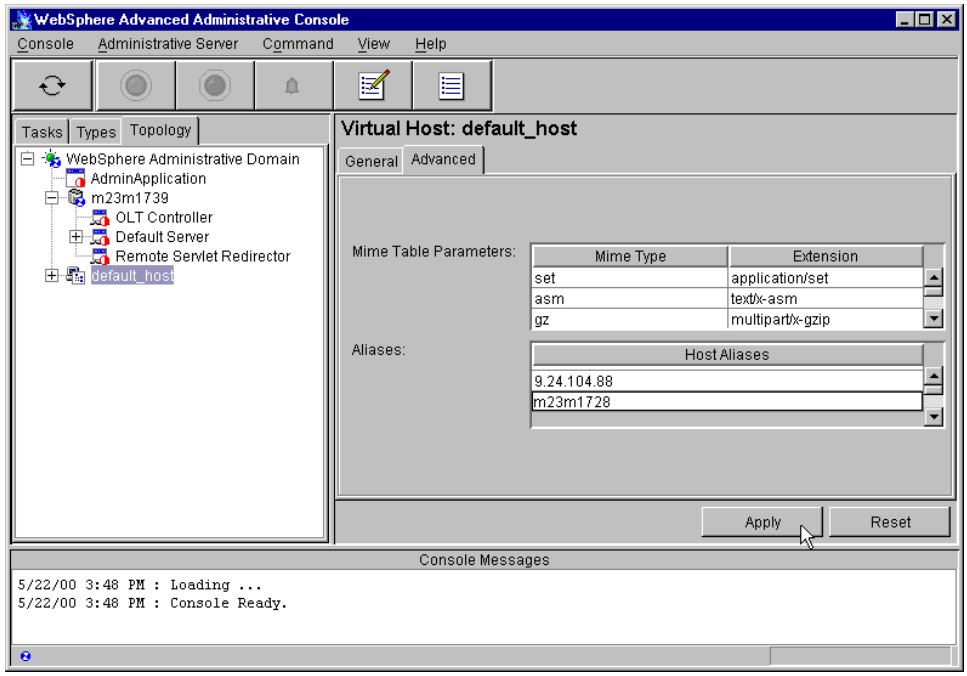

Figure 170. Add alias

Then click the **Apply** button.

When the apply completes successfully, the host aliases will have been updated.

8.5.2 Add the CORBA listener port for the application server

We recommend you specify the CORBA listener port for the application server. Since Servlet Redirector uses IIOP protocol to communicate with Application Server, you can not specify port numbers without configuring CORBA Listener Port if you have a firewall between Web server (Servlet Redirector) and application server. To do so, you need to add the parameter `-Dcom.ibm.CORBA.ListenerPort` to the Command Line Arguments filed in the Advanced tab in the application server as shown in Figure 171 on page 207. In our example, we specify 35000. Please choose a port that is not already in use.

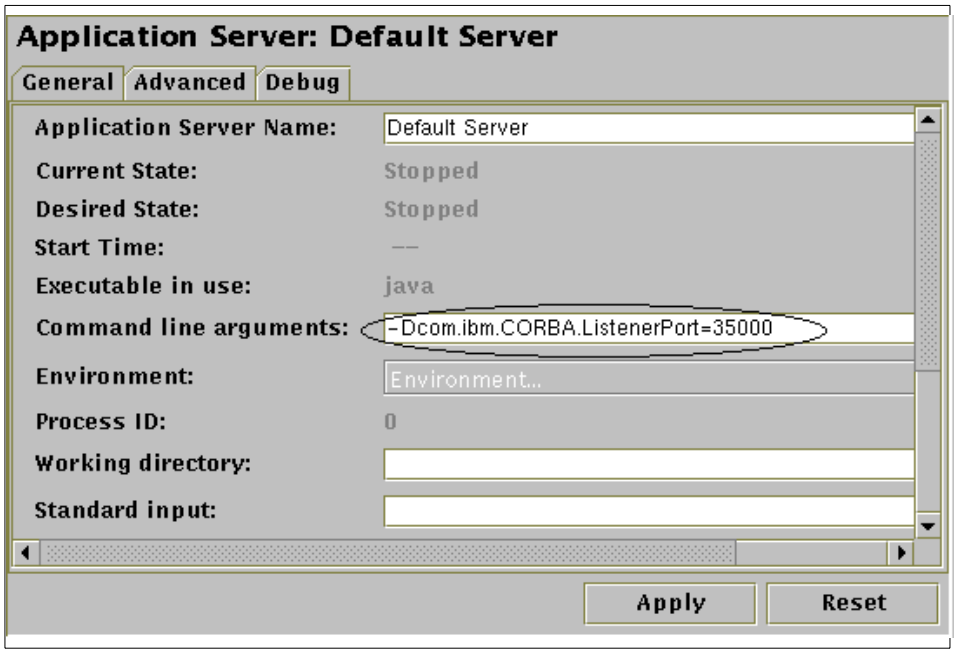

Figure 171. CORBA listener port for an application server

Note

Any changes to the application server such as adding a Web application, servlet, or EJB, should be made now. If you plan to create clones, do so now. See Chapter 4, “Application server clones” on page 77 for more information. Any changes you make after you generate the plug-in files require you to re-create these files by repeating the steps in 8.7, “Generate the Web server plug-in configuration” on page 209.

8.5.3 Start the application server

The application server (in our case, Default Server) on Machine B can be started by clicking **Default Server** in the Topology tab and then clicking the **Start** button.

8.6 Configure the Servlet Redirector

The iiopredirector.xml file is used to provide a configuration for starting the Servlet Redirector and for generating Web server plug-in configuration files. It

contains the Servlet Redirector transport information as well as Administrative Server information. The file currently resides in the <was_dir>/properties directory.

For a description of the entries in the file, refer to H.1, “Standalone Servlet Redirector iiopredirector.xml file parameters” on page 549.

You do not need to modify any of the transport entries.

You need to modify the Administrative Server information in the iiopredirector.xml file on the HTTP server (m23m1728) to point to the WebSphere (m23m1739). We did this by changing the following entries in the XML file:

1. admin-node-name

   The original line is:

   ```
   <admin-node-name>localhost</admin-node-name>
   ```

   In our case we need to change it to:

   ```
   <admin-node-name>m23m1739</admin-node-name>
   ```

2. name-service-node-name

   This is generally the same as admin-node-name.

   The original line is:

   ```
   <name-service-node-name>localhost</name-service-node-name>
   ```

   In our case we need to change it to:

   ```
   <name-service-node-name>m23m1739</name-service-node-name>
   ```

3. name-service-port

   If you are using a port other than 900 (the default) for the application server you will need to change this value.

The resulting file will look similar to Figure 172.

```
<?xml version="1.0"?>
<iiop-redirector>
   <active-transport>ose</active-transport>
   <!-- Specifies which transport block to use --!>
   <transport>
       <!-- Transport properties, 1 of n blocks --!>
      <name>ose</name>
      <code>com.ibm.servlet.engine.oselistener.SMSQTransport</code>
      <arg name="port" value="8110"></arg>
      <arg name="queueName" value="queue1"></arg>
      <!-- Must match rules.properties and queue.properties files--!>
      <arg name="maxConcurrency" value="50"></arg>
      <!-- Max number of concurrent request threads --!>
      <arg name="type" value="local"></arg>
      <!-- Choose between local and remote --!>
      <arg name="server_root" value="$server_root$"></arg>
      <arg name="cloneIndex" value="1"></arg>
      <!-- Should be 1 --!>
   </transport>
   <admin-node-name>m23m1739</admin-node-name>
   <!-- Name of the administrative node with which to communicate--!>
   <qualify-home-names>true</qualify-home-names>
   <!-- Optional, should always be true --!>
   <name-service-node-name>m23m1739</name-service-node-name>
   <!-- TCP/IP host running the name service --!>
   <name-service-port>900</name-service-port>
   <!-- Port number of the name service --!>
</iiop-redirector>
```

Figure 172. <was_dir>/properties/iiopredirector.xml

Note

The information in the XML file would otherwise be provided by command line arguments. In fact, the file is optional if you will instead use command line arguments.

8.7 Generate the Web server plug-in configuration

On the machine containing the Web server and standalone Redirector, run the Standalone Plugin Config program to generate a properties file defining the configuration of the Web server plug-in.

8.7.1 Generate the plug-in files

For our project, we created a sample batch file (thinRedirectorConfig.bat) for Windows NT and a sample shell script file for UNIX. See Figure 173 on page 212 and Figure 174 on page 213. It sets all environment variables which you need and then runs the Sandalone Plugin Config program. Place the file in the <was_dir>/bin directory on the machine containing the Servlet Redirector.

Note that the thinRedirectorConfig.bat/sh script is not included with the V3.021 WebSphere product. Therefore, we created it for this project. It is included with Version 3.5 WebSphere product. So, you don't have to create it by hand if you are using V3.5 WebSphere.

Note that before you generate the plug-in files, make sure that the application server is running on Machine B.

To generate the plug-in files, you need to run the batch file or shell script. Type the following command from a command prompt, for example:

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>
C:\>\WebSphere\AppServer\bin\thinRedirectorConfig.bat
000.030 2018fad StandalonePlu A Generating.Plugin.Configuration.For.This.Node

C:\>
```

Ensure three parameters are defined correctly in the script or batch file:

serverRoot: Specifies the root of the application server product installation on the machine containing the administrative server.

adminNodeName: Specifies the TCP hostname of the machine containing the WebSphere adminstrative server with which the remote, standalone Redirector will communicate.

queueProps: Specifies the full path to the XML file containing information for configuring the plug-in, for example, c:\WebSphere\AppServer\properties\iiopredirector.xml. The file resides on the machine containing the Web server and Redirector.

Note

- The first two arguments (-serverRoot and -adminNodeName) are required.
- If the name service options (-nameServiceNodeName and -nameServicePort) are not specified, localhost/900 will be used.

The queue properties (-queueName, -queuePort and -queueType) can be loaded either from the iiopredirector.xml file (using the -queueProps argument) or from the command line arguments.

```
@echo off
setlocal
call setupCmdLine.bat
rem setup the classpath
set WAS_CP=%WAS_HOME%\lib\ibmwebas.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\properties
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\servlet.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\webtlsrn.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\lotusxsl.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ns.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ejs.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ujc.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\repository.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\admin.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\swingall.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\console.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\tasks.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\xml4j.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\x509v1.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\vaprt.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\iioprt.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\iioptools.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\dertrjrt.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\sslight.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ibmjndi.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\deployTool.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\databeans.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\classes
set WAS_CP=%WAS_CP%;%JAVA_HOME%\lib\classes.zip
set CLASSPATH=%WAS_CP%

java com.ibm.servlet.engine.oselistener.systemsmgmt.StandalonePluginCfg
-serverRoot c:\WebSphere\AppServer -adminNodeName m23m1739
-queueProps c:\WebSphere\AppServer\properties\iiopredirector.xml

endlocal
```

Figure 173. <was_dir>\bin\thinRedirectorConfig.bat

```
#!/bin/ksh
. $PWD/setupCmdLine.sh

CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ibmwebas.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/properties
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/servlet.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/webtlsrn.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/lotusxsl.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ns.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ejs.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ujc.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/repository.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/admin.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/swingall.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/console.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/tasks.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/xml4j.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/x509v1.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/vaprt.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/iioprt.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/iioptools.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/dertrjrt.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/sslight.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ibmjndi.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/deployTool.jar
CLASSPATH=$CLASSPATH:$JAVA_HOME/lib/classes.zip

export CLASSPATH=$CLASSPATH

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WAS_HOME/bin:$WAS_HOME/lib

$JAVA_HOME/bin/java
com.ibm.servlet.engine.oselistener.systemsmgmt.StandalonePluginCfg
-serverRoot /usr/WebSphere/AppServer -adminNodeName rs600013
-queueProps /usr/WebSphere/AppServer/properties/iiopredirector.xml
```

Figure 174. <was_dir>/bin/thinRedirectorConfig.sh

The line beginning "java com.ibm.servlet.engine ..." has been divided with line breaks for easier reading. In your actual script or bat file, ensure that the java command is on a single line.

In other words, combine all of the lines from "java com.ibm.servlet.engine..." to "-queueProps ..." on one line, each separated by a space.

8.7.2 Verify the plug-in properties files

Verify that the plug-in properties files: rules.properties, queue.properties, and vhost.properties, in the <was_dir>/temp directory on Machine A were generated correctly. The files should look similar to the ones shown below.

The files can reveal which requests are being handled by the Servlet Redirector (using RMI/IIOP), and which are being handled on the machine local to the Web server (using an OSE transport).

The values in the queues.properites file should be the same as the values specified in the iiopredirector.xml file.

```
#IBM WebSphere Plugin Communication Queues
#Mon May 22 16:12:50 EDT 2000
ose.srvgrp.queue1.clone1.port=8110
ose.srvgrp.queue1.clone1.type=local
ose.srvgrp.queue1.type=FASTLINK
ose.srvgrp=queue1
ose.srvgrp.queue1.clonescount=1
```

Figure 175. <was_dir>/temp/queues.properties on the HTTP server machine

```
#IBM WebSphere Plugin Virtual Host Mappings
#Mon May 22 16:12:50 EDT 2000
9.24.104.73=default_host
m23m1728.itso.ral.ibm.com=default_host
m23m1739=default_host
m23m1739.itso.ral.ibm.com=default_host
9.24.104.88=default_host
localhost=default_host
127.0.0.1=default_host
m23m1728=default_host
```

Figure 176. <was_dir>/temp/vhosts.properties on the HTTP server machine

```
#IBM WebSphere Plugin URL Mapping Rules
#Mon May 22 16:12:50 EDT 2000
default_host/webapp/examples/HitCount=queue1
default_host/webapp/examples/ErrorServlet=queue1
default_host/webapp/examples/simpleJSP.servlet=queue1
default_host/servlet=queue1
default_host/ErrorReporter=queue1
default_host/admin/install=queue1
default_host/webapp/examples/showCfg=queue1
default_host/webapp/examples/*.jsp=queue1
default_host/*.jsp=queue1
default_host/webapp/examples/verify=queue1
default_host/servlet/snoop2=queue1
default_host/servlet/snoop=queue1
default_host/webapp/examples/ping=queue1
default_host/admin/*.jsp=queue1
default_host/webapp/examples/SourceCodeViewer=queue1
default_host/webapp/examples/=queue1
default_host/webapp/examples=queue1
default_host/admin=queue1
default_host/admin/servlet=queue1
default_host/servlet/hello=queue1
default_host/admin/=queue1
default_host/admin/ErrorReporter=queue1
default_host/webapp/examples/simpleJSP=queue1
```

Figure 177. <was_dir>/temp/rules.properties on the HTTP server machine

Note

The files in the <was_dir>/temp directory on the HTTP server machine (Machine A) will not be the same as the files on the application server machine (Machine B).

8.8 Start the servers

Now you are ready to start your servers.

1. Start the application server on Machine B if not running.
2. Start the HTTP server on Machine A. If it's already started, restart it.
3. Start the Standalone Servlet Redirector on Machine A.

We created a sample batch file (thinRedirectorStart.bat) for Windows NT as shown in Figure 178 on page 217 and a sample shell script file (thinRedirectorStart.sh) for UNIX as shown in Figure 179 on page 218 that sets environment values which you need and then starts the Java process for the standalone Redirector. The Java process will communicate with application servers through an Administrative Server installed on a different machine.

Place the file in the <was_dir>/bin directory on the machine containing the Servlet Redirector and Web server.

Note that the thinRedirectorStart script/batch file is also included with the V3.5 product and does not need to be modified.

To start the thin Servlet Redirector, you simply need to run the batch file or shell script. For example, type the following command from a command prompt:

```
c:\WebSphere\AppServer\bin\thinRedirectorStart.bat
```

If you start thin Servlet Redirector successfully, you will see the following output on the command prompt.

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>\WebSphere\AppServer\bin\thinRedirectorStart.bat
Using Formatted Trace Buffer
000.411 2018fad IIOPRedirecto A Servlet.Redirector.Starting...
003.185 2018fad IIOPRedirecto A Servlet.Redirector.Running...
```

To stop thin Servlet Redirector, you need to input Ctrl+C from the command prompt on which you started it.

```
@echo off
setlocal
call setupCmdLine.bat
rem setup the classpath
set WAS_CP=%WAS_HOME%\lib\ibmwebas.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\properties
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\servlet.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\webtlsrn.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\lotusxsl.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ns.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ejs.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ujc.jar
set WAS_CP=%WAS_CP%;%DB2DRIVER%
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\repository.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\admin.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\swingall.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\console.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\tasks.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\xml4j.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\x509v1.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\vaprt.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\iioprt.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\iioptools.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\dertrjrt.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\sslight.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\ibmjndi.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\deployTool.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\lib\databeans.jar
set WAS_CP=%WAS_CP%;%WAS_HOME%\classes
set WAS_CP=%WAS_CP%;%JAVA_HOME%\lib\classes.zip
set CLASSPATH=%WAS_CP%

java -Dcom.ibm.CORBA.ListenerPort=40000 -Dserver.root=%WAS_HOME%
com.ibm.servlet.engine.ejs.IIOPRedirector -traceString *=all=disabled

endlocal
```

Figure 178. <was_dir>\bin\thinRedirectorStart.bat

```
#!/bin/ksh
. $PWD/setupCmdLine.sh

CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ibmwebas.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/properties
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/servlet.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/webtlsrn.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/lotusxsl.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ns.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ejs.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ujc.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/repository.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/admin.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/swingall.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/console.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/tasks.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/xml4j.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/x509v1.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/vaprt.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/iioprt.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/iioptools.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/dertrjrt.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/sslight.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/ibmjndi.jar
CLASSPATH=$CLASSPATH:$WAS_HOME/lib/deployTool.jar
CLASSPATH=$CLASSPATH:$JAVA_HOME/lib/classes.zip

export CLASSPATH=$CLASSPATH

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WAS_HOME/bin:$WAS_HOME/lib

$JAVA_HOME/bin/java -Dcom.ibm.CORBA.ListenerPort=40000
-Dserver.root=$WAS_HOME com.ibm.servlet.engine.ejs.IIOPRedirector
-traceString com.ibm.*=all=disabled
```

Figure 179. <was_dir>/bin/thinRedirectorStart.sh

CORBA listener port

In the Java command, put a valid TCP port number to -Dcom.ibm.CORBA.ListenerPort. Ports can range from 1024 to 64000. Please choose a port that is not already in use. You can check in-use ports in the /etc/services file on UNIX, and in the c:\winnt\system32\drivers\etc\services file on Windows.

This port will be used to receive the response data from WebSphere.

The line beginning "java -Dcom.ibm.CORBA ..." has been divided with a line break for easier reading. In your actual script or bat file, ensure that the java command is on a single line.

In other words, combine the two lines "java -Dcom.ibm.CORBA ..." and "com.ibm.servlet.engine ..." onto one line.

8.9 Test configuration

Now that the HTTP server, standalone Servlet Redirector, and WebSphere are running, we need to test the servers to make sure that everything is configured properly.

Using a Web browser, access the following URI (m23m1728 is the HTTP server and /webapp/examples/showCfg is the servlet):

```
http://m23m1728/webapp/examples/showCfg
```

Figure 180 on page 220 shows the result of the request.

Notice that the hostname is the name of the node running the application server (m23m1739), and not the HTTP server (m23m1728).

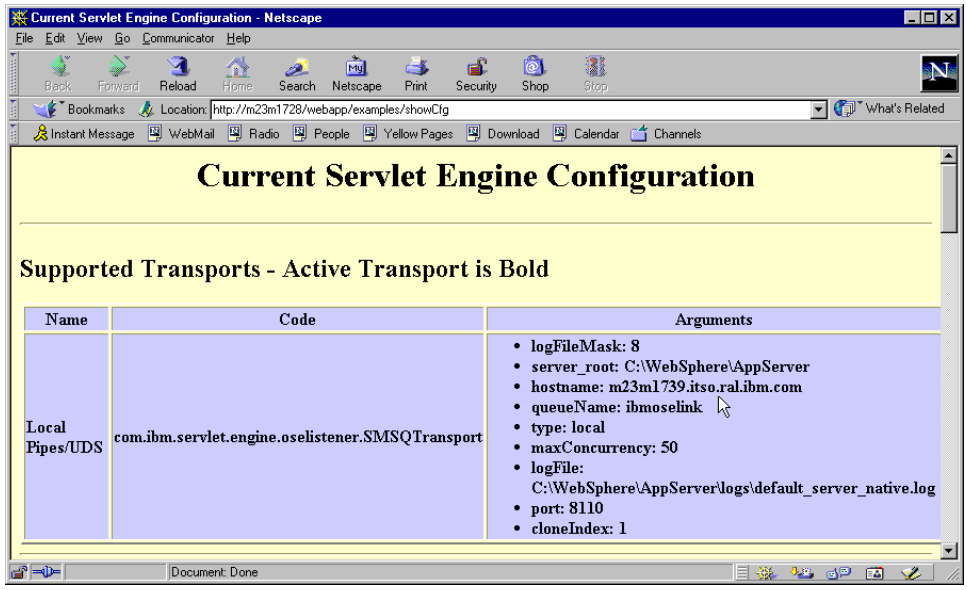

Figure 180. Output in browser

8.10 Related topology

WebSphere Version 3.021 and 3.5 support the use of clones and workload management with Servlet Redirector.

An HTTP server and a stand-alone Servlet Redirector will be installed on Machine A. Two application server clones will run on Machine B. Requests will be redirected from Machine A to the application server clones running on Machine B.

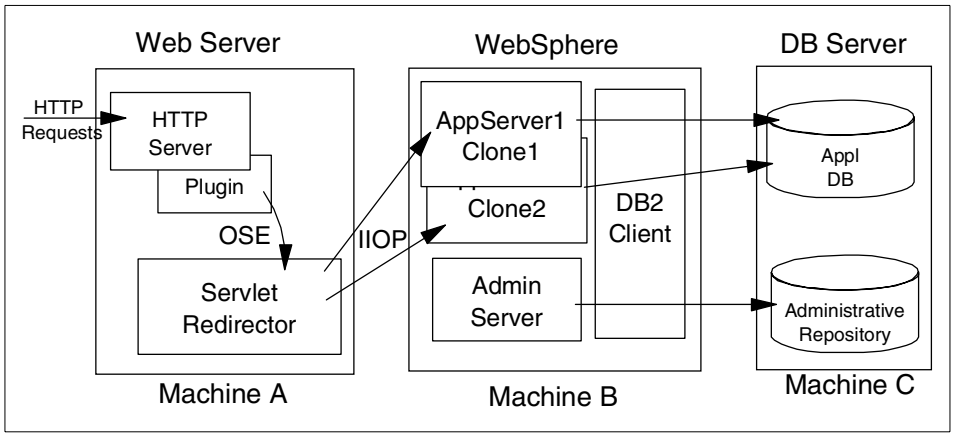

Figure 181. Standalone Servlet Redirector with clones

The clones are configured using the process described in 4.4, “Create the model” on page 80 through 4.6, “Create the clone” on page 84.

Note

With V3.5, you should convert your initial Web application into a clone. If not, you can create a model but you cannot create a clone.

With V3.021, you do not have to convert your initial Web application into a clone; however since we are using Servlet Redirector we need to select the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model’s clones to fail unless this server is made into a clone.

Before you start the model, you need to modify the CORBA listener port for each application server clone in the command line arguments. Note that you have to specify different port numbers for each clone on the same machine. If you specify the same port number for clones on the same machine, you cannot start them.

The configuration of the Servlet Redirector is the same as described previously in this chapter.

8.11 Maintenance/troubleshooting

The Servlet Redirector runs in a job independent of the HTTP server. Because of this, there are some considerations when doing maintenance on the application server.

8.11.1 Terminating an application server

When an application server aborts suddenly (we used the `kill -9 PID` command to simulate this situation), WebSphere automatically restarts the server. No action is required.

Servlet Redirector will automatically route the request to alternative servers.

If there are no clones of this server, no requests will be served and errors will be reported until the application server restarts.

We assume that you do not have clones and you specified the CORBA listener port for the application server and Administrative Server.

8.11.2 Stopping an application server

When an application server is stopped from the Administrative Console, the routing of the requests will automatically continue on to the other clones. No additional actions are required.

8.11.3 Restarting an application server

For maintenance or any other reason, you might stop and restart your application server. When the application server is restarted, browsers automatically get access to it. No actions are required.

8.11.4 Restarting a clone

When a clone is restarted, browsers automatically get access to it. No actions are required.

8.11.5 Stopping an administrative server

With multiple clone/multiple node environments, when one of the Administrative Servers, with which the Servlet Redirector does not use to communicate, is stopped, the routing of the requests will continue on to the other clones on the other node. No actions are required.

8.11.6 Restarting a machine

With multiple clone/multiple node environments, when a node is restarted, the Servlet Redirector will access it. If a quicker turn-around is needed, the refresh can be forced by restarting the Servlet Redirector

8.11.7 Modifying the application server

If any changes are made to the application server that affect the queues.properties, rules.properties file or vhosts.propeties file for the thin Servlet Redirector node, the plug-in files need to be regenerated as described in 8.7, “Generate the Web server plug-in configuration” on page 209.

Such changes include:

- Adding/removing a URL (Web resource) from the environment.
- Secure/unsecure a URI (when you add security to a URI, an “A” is added to it, changing its name.)
- Adding/removing a host alias.
- Adding/removing a server.
- Adding/removing a clone.

After the files are regenerated, the Servlet Redirector and HTTP server must be restarted.

Chapter 9. Three-tier topologies

Previous chapters have described how to physically separate the HTTP server from the WebSphere application server(s). It's also possible to further partition the application server into an application server for servlets and application server for EJBs (or multiple application servers for each of these). This partitioning can be desirable from a scalability perspective and from a security perspective as well. In terms of scalability, partitioning the application may be desirable when the application is comprised primarily of either servlets or EJBs. Partitioning the application in this manner allows more finite control of the server resources associated with the application server processes. From a security perspective partitioning the application in this manner allows placement of the EJB server inside a firewall (or a second firewall). Though not described here one could choose to configure an Administrative Server agent for the machine running the servlets (for further information on configuring an Administrative Server agent, refer to Chapter 7, "Thick Servlet Redirector with Admin Server agent" on page 179).

This chapter provides instructions on partitioning the application server processes so that one machine services HTTP requests, another machine serves servlet requests and a third machine is used to serve EJBs.

We will discuss the following steps for setting up this configuration:

1. Installation summary
2. Configure the EJB application server
3. Configure the servlet application server
4. Configure the OSE Remote
5. Start the servers
6. Test the servers
7. Maintenance/troubleshooting
8. Related topologies

9.1 Overview of the configuration

An HTTP server and WebSphere Application Server will be installed on Machine A (rs600010) and the HTTP server plug-in will be configured for Remote OSE. A WebSphere Administrative Server and application server will be installed in Machine B (rs600015) along with a DB2 client. We created an application server and its clone on Machine B. The same components will

also be installed on Machine C (rs600013) as well as an application server and its clone. The DB2 UDB server will be installed in Machine D (rs600012). This is where the WebSphere Administrative Repository will be housed as well as the application data and a database for HTTP session persistence.

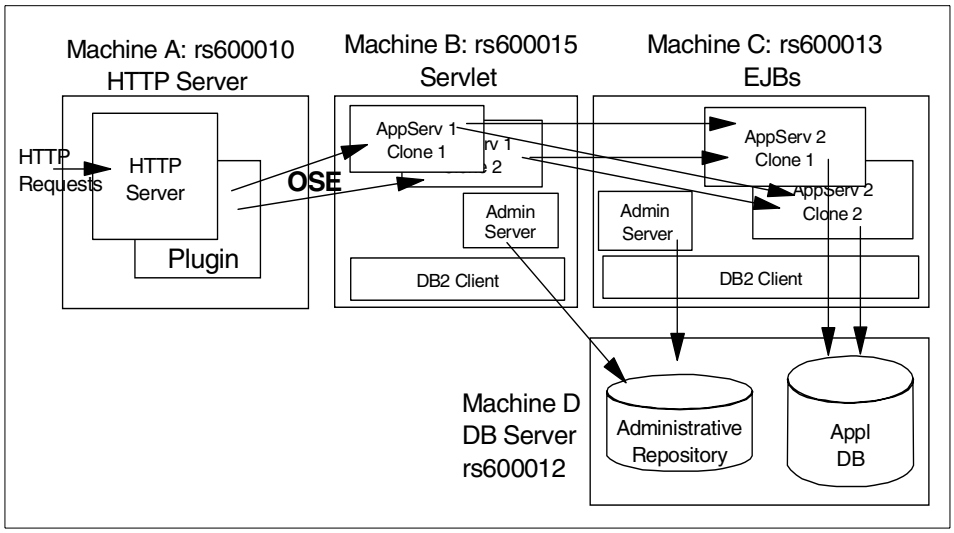

Figure 182. Three-Tier WebSphere applications

9.2 Installation summary

Table 8 is a summary of the software installation steps. We will not discuss this. Please refer to the *Getting Started* documentation which is part of the WebSphere product.

Table 8. Software installation steps

Step No.	Machine A (HTTP)	Machine B (Servlet)	Machine C (EJBs)	Machine D (DB)
1				Install DB2 UDB Apply FixPack
2				Create DB
3		Install DB2 Client Apply FixPack	Install DB2 Client Apply FixPack	
4		Catalog node Catalog db	Catalog node Catalog db	

Step No.	Machine A (HTTP)	Machine B (Servlet)	Machine C (EJBs)	Machine D (DB)
5	Install JDK (V3.02x)	Install JDK (V3.02x)	Install JDK (V3.02x)	
6	Install Web server			
7	Install WebSphere (Production Application Server: -Plugins, -Core Server)			
8		Install WebSphere (Production Application Server: -Core Server)	Install WebSphere (Production Application Server: -Core Server)	

9.3 Configuration flow summary

Table 9 is a summary of the configuration steps for our sample topology configuration.

Table 9. Configuration summary

Step No.	Machine A (HTTP)	Machine B (Servlet)	Machine C (EJBs)
1			Generate WLM enabled JAR
2			Place WLM enabled JAR
3			Start Admin Server
4			Start Admin Console
5			Create application server
6			Create container
7			Create enterprise bean

Step No.	Machine A (HTTP)	Machine B (Servlet)	Machine C (EJBs)
8			Create model
9			Configure WLM policy
10			Create clones
11		Place WLM enabled JAR	
12		Update bootstrap.properties for OSE Remote	
13		Start Admin Server	
14		Refresh console	
15		Add host aliases	
16		Create application server	
17		Create servlet engine	
18		Update transport type for OSE Remote	
19		Create Web application	
20		Create servlet	
21		Create model	
22		Create clones	
23			Start model
24		Start model	
25	Update bootstrap.properties for OSE Remote		
26	Update plug-in configuration properties files		
27	Start HTTP server		

9.4 BeenThere sample application

We used the BeenThere sample application for the three-tier WebSphere topology configuration. This sample is included in WebSphere V3.5. As a result we used WebSphere V3.5 in this chapter. You may notice that the V3.5 Administrative Console looks slightly different than V3.02x.

The BeenThere sample is used to demonstrate Workload Management (WLM) in the WebSphere Application Server V3.5 Advanced Edition. Since it's difficult to actually see WLM in action, this application displays execution information in order to allow a user to see that WLM functionality is being performed. The BeenThere servlet displays the following information:

- Hostname of the machine executing the BeenThere servlet.
- Application Server name that is executing the BeenThere servlet.
- Process ID of the Java Virtual Machine (JVM) process executing the BeenThere servlet.
- Hostname of the machine executing the BeenThere Session Enterprise Java Bean (EJB).
- Application Server name that is executing the BeenThere Session EJB.
- Process ID of the JVM process executing the BeenThere Session EJB.

Additionally, the BeenThere servlet can:

- Execute the BeenThere session EJB multiple times in one servlet execution instance.
- Display BeenThere session EJB execution statistics.
- Log the servlet results to a file.

9.5 EJB application server configuration

The instructions below describe the steps necessary to create the EJB application server and deploy the BeenThere EJBs for our application as well as how to create the model and clone.

9.5.1 Step 1: generate WLM-enabled JAR on Machine C

In order to utilize WebSphere WLM, your EJBs will have to be WLM enabled. Three options exist to WLM your EJBs:

- Use VisualAge for Java to create your EJB application.

- Select the option to make your application WLM enabled when you create on enterprise application with WebSphere V3.5 Administrative Console.
- Use wlmjar.sh (or wlmjar.cmd for Windows).

Depicted below is the sample output of the wlmjar.sh used to create a WLM-enabled application. See Appendix B, “WLM enabling EJBs” on page 487 for detailed information.

Note

Since the BeenThereEJB JAR file which comes with WebSphere V3.5 is already WLM-enabled, we don’t need to use `wlmjar` to make the EJB JAR file WLM-enabled. Therefore, we will use our own sample EJB JAR file called AddBean.jar to show you how to use the `wlmjar` command.

For example:

```
wlmjar -J AddBean.jar -IF Add -IF AddHome -P redbook
```

```
C:\WebSphere\AppServer\bin>dir *.jar
 Volume in drive C has no label.
 Volume Serial Number is 7867-2A7C

 Directory of C:\WebSphere\AppServer\bin

07/31/2000  11:27p             33,657 AddBean.jar
               1 File(s)         33,657 bytes
               0 Dir(s)     242,245,632 bytes free

C:\WebSphere\AppServer\bin>
C:\WebSphere\AppServer\bin>wlmjar -J AddBean.jar -IF Add -IF AddHome -P redbook
        1 file(s) copied.

C:\WebSphere\AppServer\bin>dir *.jar
 Volume in drive C has no label.
 Volume Serial Number is 7867-2A7C

 Directory of C:\WebSphere\AppServer\bin

07/31/2000  11:27p             33,657 AddBean.jar
07/31/2000  11:28p             43,594 _wlm_AddBean.jar
               2 File(s)         77,251 bytes
               0 Dir(s)     242,200,576 bytes free

C:\WebSphere\AppServer\bin>
```

9.5.2 Step 2: place EJB JAR file on Machine C

You have to place your EJB JAR file on Machine C (rs600013).

For example:

```
C:\WebSphere\AppServer\bin>copy _wlm_AddBeanjar ..\deployedEJBs\
```

Note

In our example in this chapter, we used the BeenThere sample as we mentioned earlier. The WLM-enabled jar (_wlm_BeenThere.jar) has been placed in the /usr/WebSphere/AppServer/deployedEJBs directory. Therefore, we actually skipped the steps 1 and 2.

The BeenThere sample (WLM-enabled) should be in the <was_dir>/deployedEJBs directory if you are using V3.5.

9.5.3 Steps 3 & 4: start Admin Server and Console on Machine C

You can start the WebSphere Administrative Server on Machine C (rs600013).

And then you can start the Administrative Console on Machine C or any other machine which must point to the Administrative Server on Machine C.

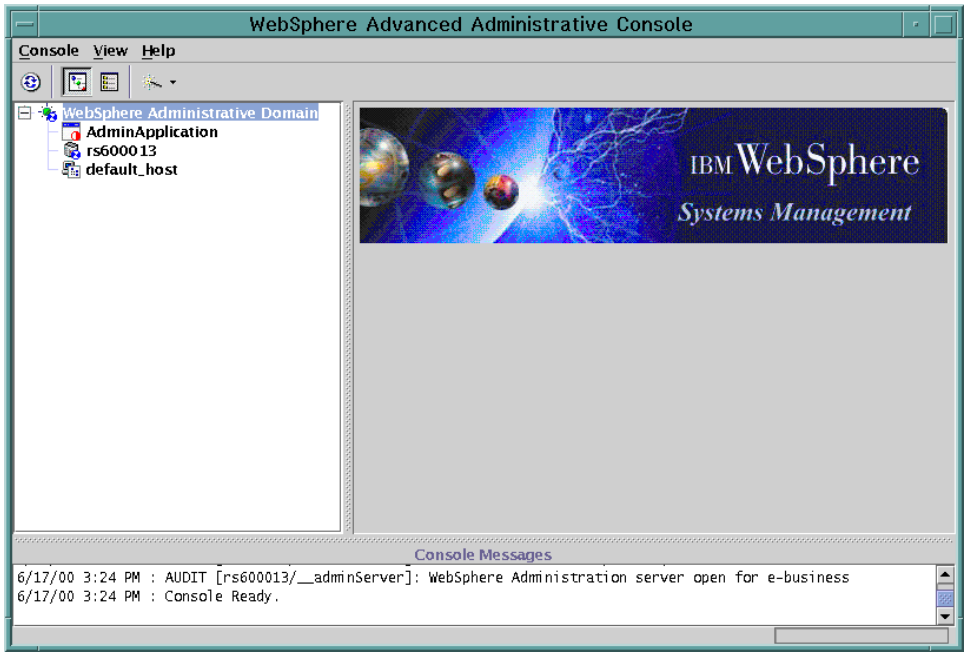

Figure 183. WebSphere 3.5 Administrative Console

If you are using WebSphere V3.02x, the GUI looks slightly different.

9.5.4 Step 5: create an application server on Machine C

In the console, highlight and right click Machine C (rs600013).

Select **Create** and **Application Server** as depicted in Figure 184 on page 233.

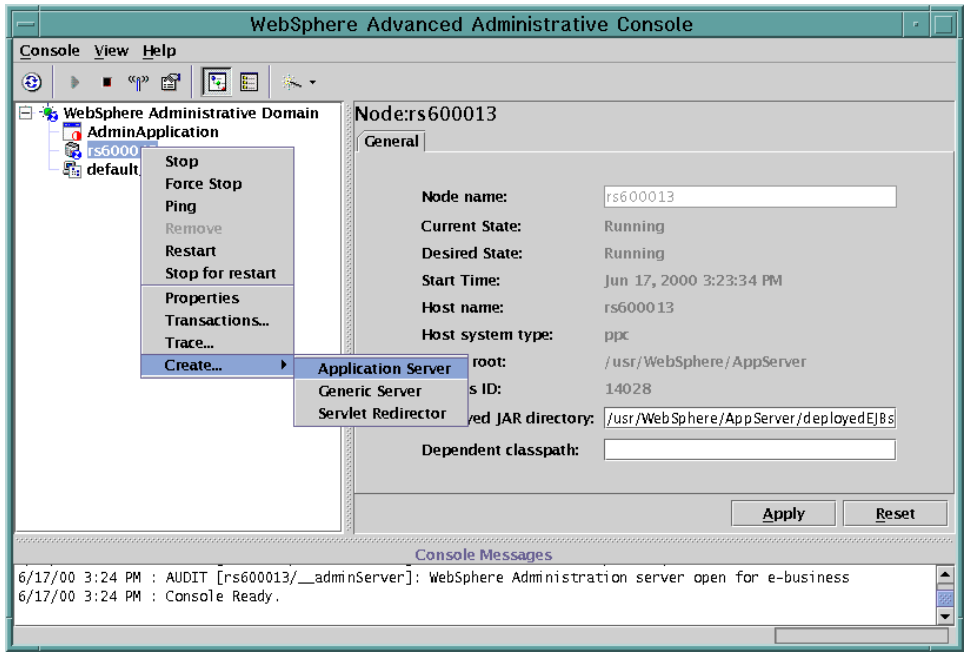

Figure 184. Create an application server

Then you will see the Create Application Server window.

We specified “BeenThere EJB Server” in the Application Server Name field as our application server’s name as shown in Figure 185.

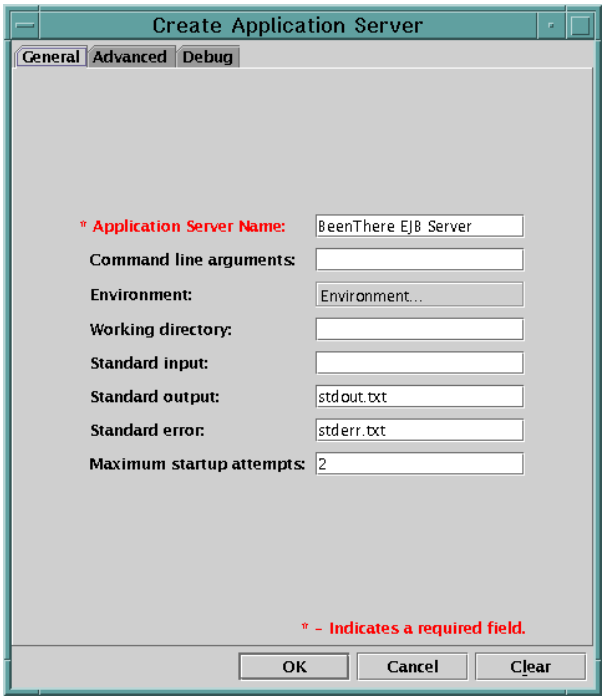

Figure 185. Create application server windows

Then click **OK**.

9.5.5 Step 6: create container on Machine C

Create an EJB container for your EJB application. To do so, highlight the application server that you created in the previous step and right click it.

Select **Create** and **EJBContainer** as depicted in Figure 186.

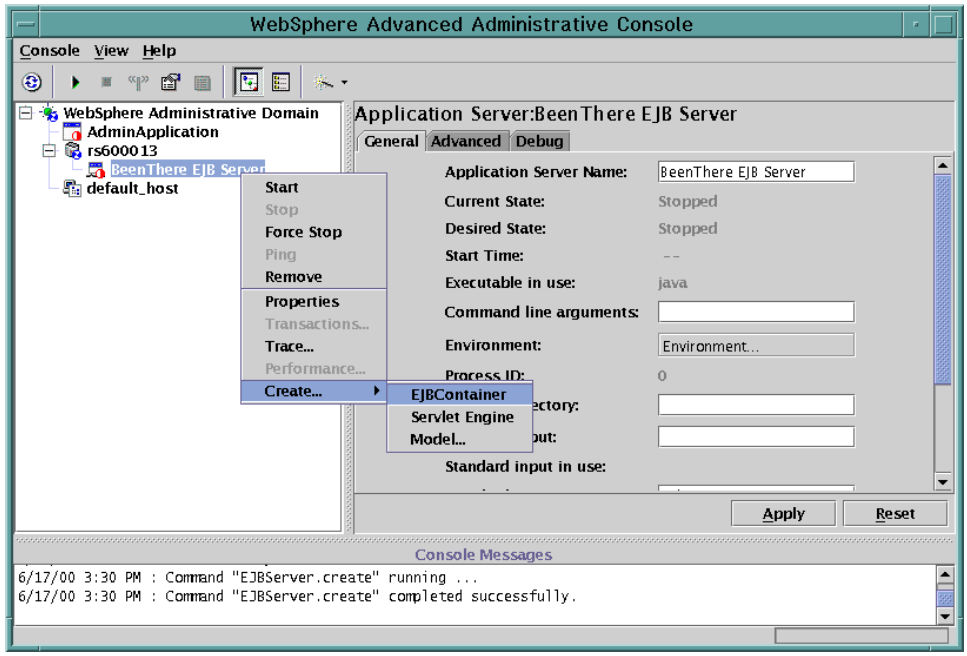

Figure 186. Create EJBContainer

Then you will get a Create EJBContainer window.

We specified “BeenThere EJB Container” as our container’s name as shown in Figure 187.

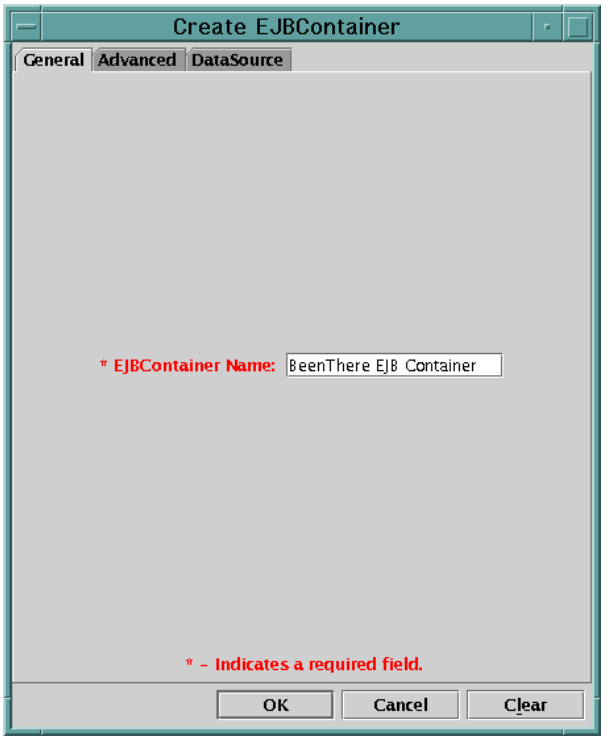

Figure 187. Create EJBContainer windows

Then click **OK**.

9.5.6 Step 7: create an enterprise bean on Machine C

Install/create an enterprise bean for the BeenThere Session EJB into the new EJB container just created.

Highlight the container which you created (in our case, BeenThere EJB container) and right click it.

Select **Create** and **EnterpriseBean** as depicted in Figure 188.

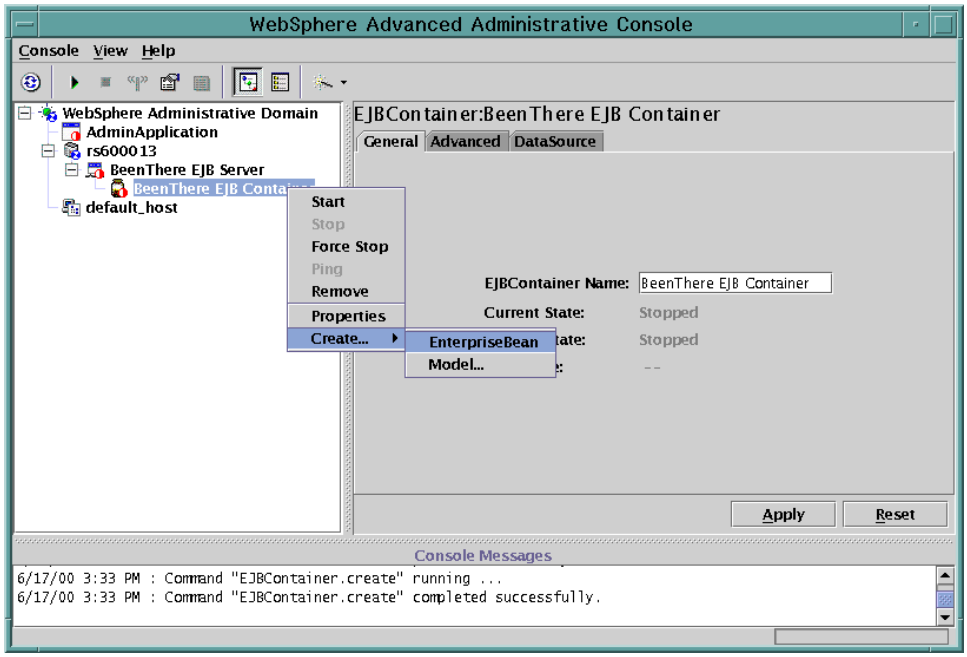

Figure 188. Create EnterpriseBean

Then you will get a Create EnterpriseBean window as shown in Figure 189.

Figure 189. Create EnterpriseBean window

Click the **Browse..** button and you will see the Open window as shown in Figure 190.

Figure 190. Selecting your EJB

Then select the file (EJB application) which you want to install.

For our example, we selected the <was_dir>/deployedEJBs/_wlm_DeployedBeenThere.jar file.

You may get the following Confirm dialog depending on your JAR file.

We clicked Yes for our sample application.

Figure 191. Confirm dialog

Then click **OK** to create the EnterpriseBean as shown in Figure 192.

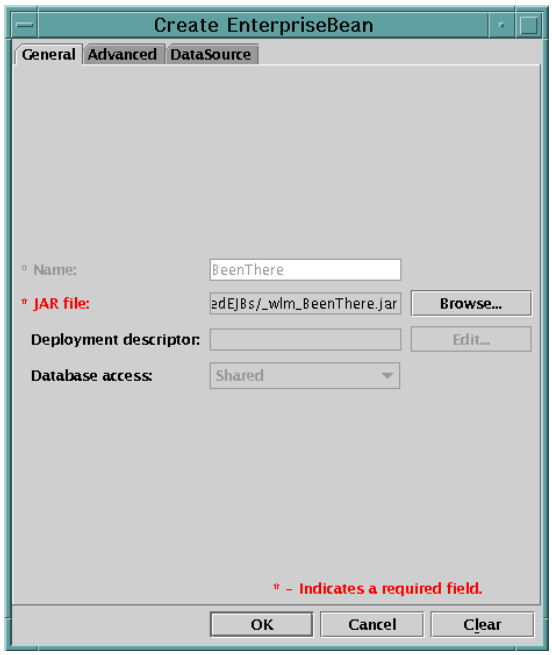

Figure 192. Create WLM enabled EnterpriseBean

Now, you can see the EnterpriseBean in the Administrative Console. In our case, we see the BeenThere EnterpriseBean as shown in Figure 193.

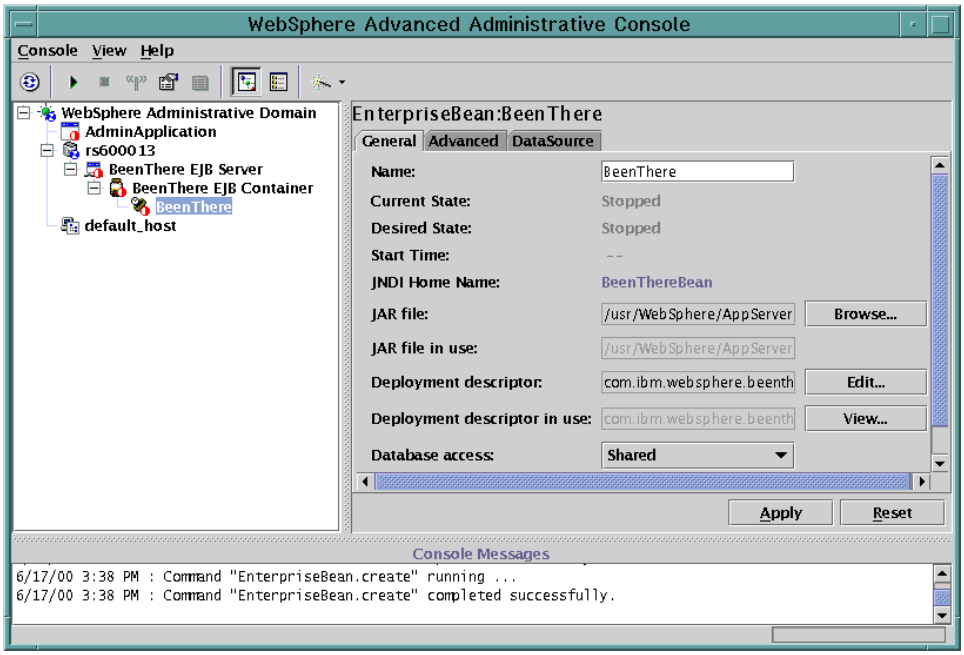

Figure 193. EnterpriseBean in the Administrative Console

9.5.7 Step 8: create model on Machine C

The steps to create a model and a clone are the same as discussed in Chapter 4, “Application server clones” on page 77. Therefore, we will not discuss these steps in detail.

Select your application server (in our case, BeenThere EJB Server) and right click it.

Select **Create** and **Model** as shown in Figure 194.

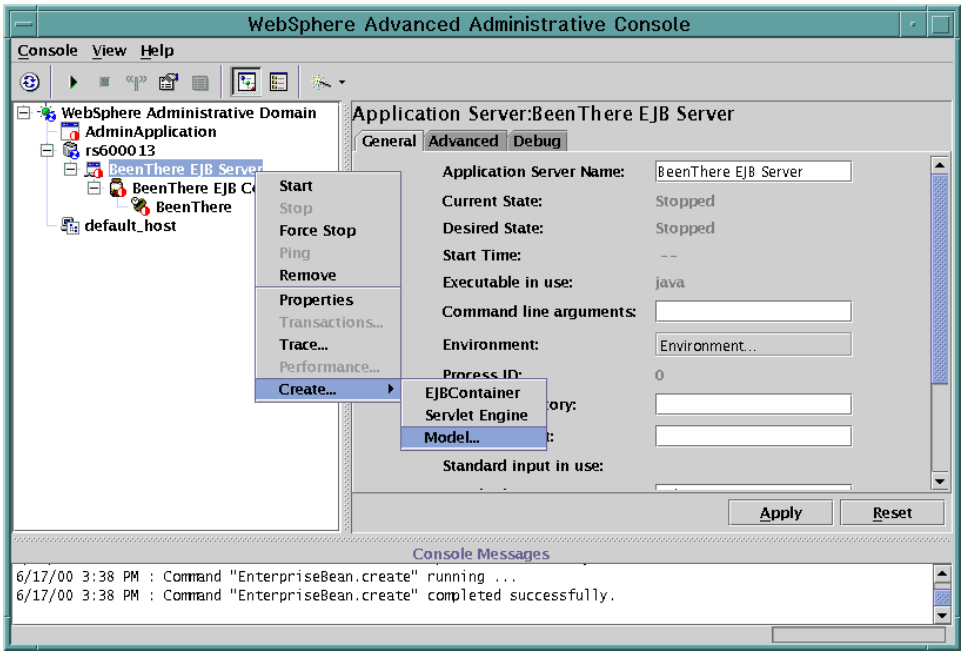

Figure 194. Create Model

Then you will get the Model Properties window.

Specify the name of the model which you are creating.

At the General tab, select the following options:

- Make <your application server> a Clone.
- Recursively Model all Instances under <your application server>.

In our example, we specified “BeenThere EJB Server Model” in the Model Name field as shown in Figure 195 on page 242.

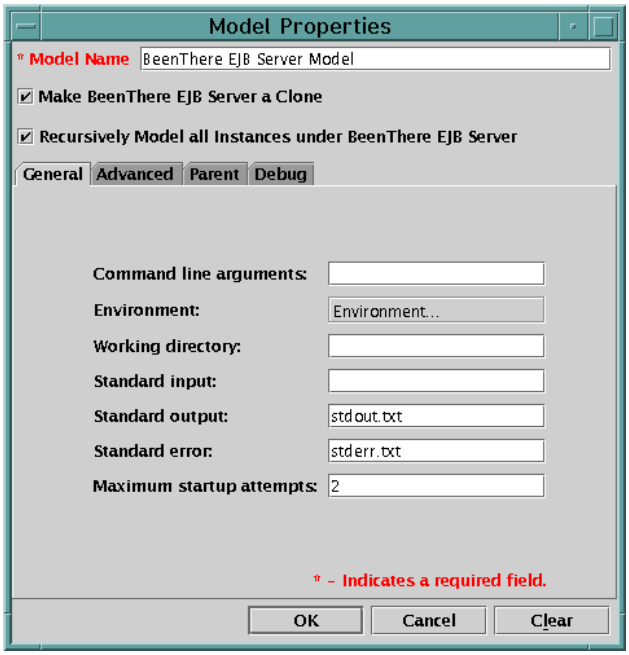

Figure 195. Model Properties window

Then click **OK**.

9.5.8 Step 9: configure WLM policy

After you create your application server model, you should configure the workload management selection policy. To do so, select the model which you just created.

At the Advanced tab, select **Workload management selection policy** and click **Apply**. In our example, we selected “roundrobin” as shown in Figure 196 on page 243.

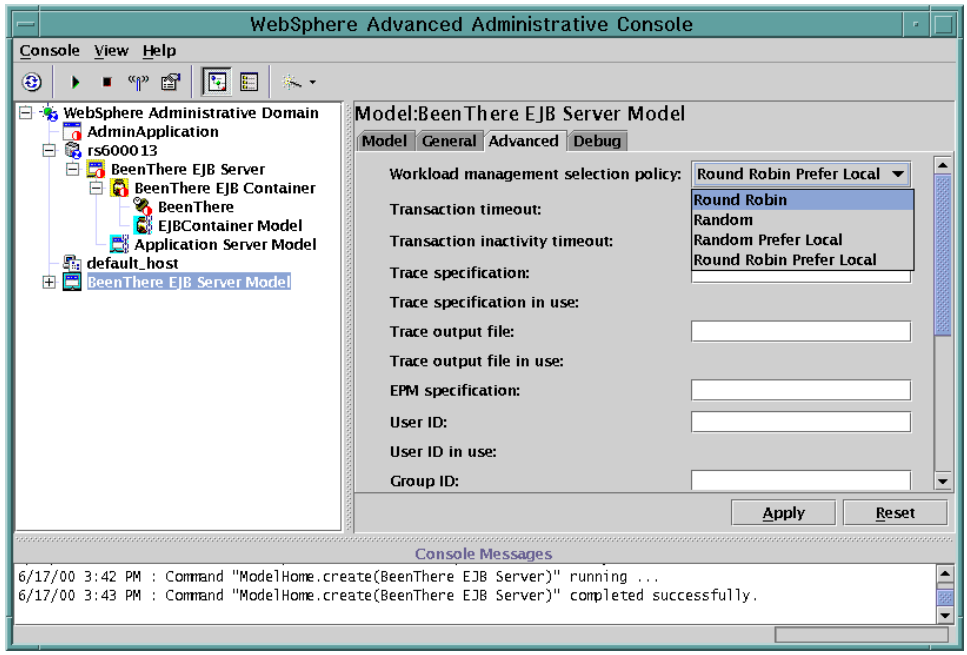

Figure 196. Workload management selection policy

9.5.9 Step 10: create clones

Create a second application server clone from the model just created.

Select the model (in our example, BeenThere EJB Server Model) and right click. Then select **Create** and **Clone** as shown in Figure 197.

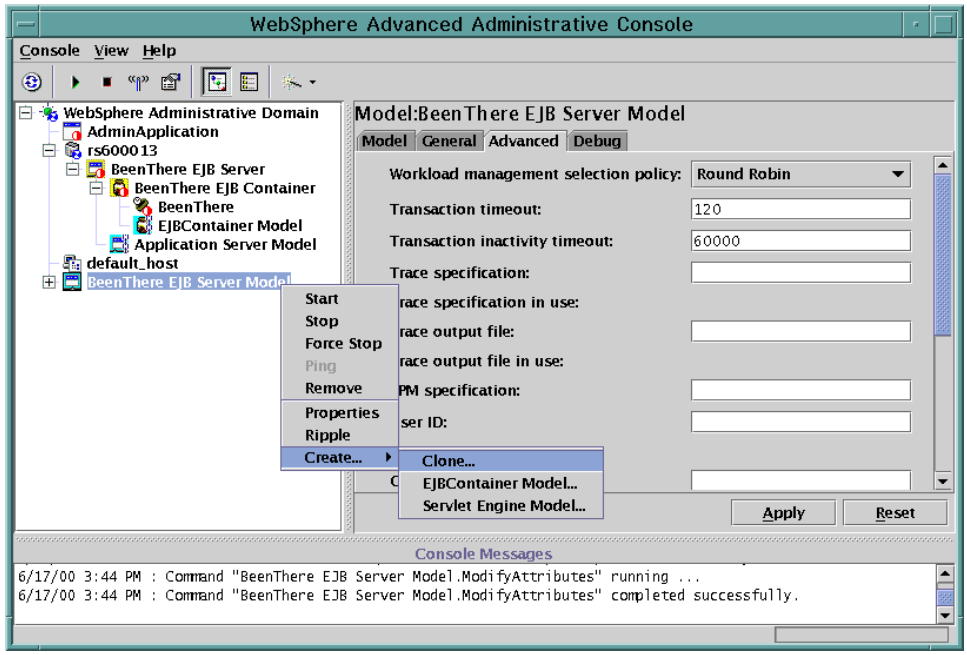

Figure 197. Create EJB application clone on Machine C

You will get the Clone Parent window.

We specified the name in the Name field for this example call it BeenThere EJB Server Clone1 and select the node on which you want to run the clone as shown in Figure 198.

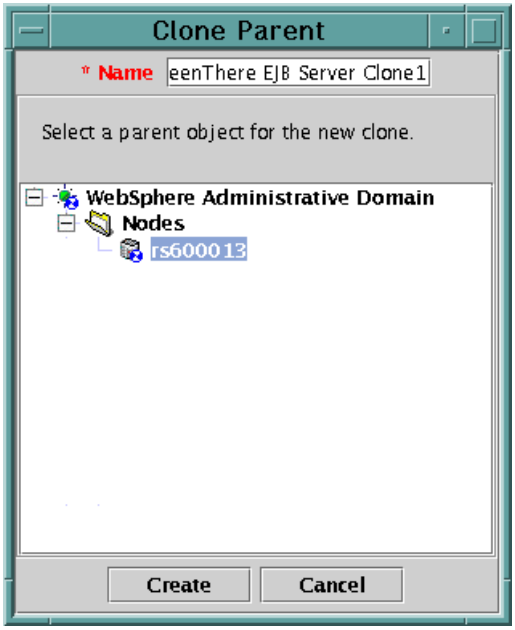

Figure 198. Clone parent

Then click **Create**.

Now, you have finished creating your EJB application server, EJB container, EnterpriseBean, model and clone.

9.6 The servlet application server configuration

The instructions below describe the steps necessary to create the servlet application server and deploy the BeenThere servlet for our application. It also shows how to create a model and clone.

9.6.1 Step 11: place WLM-enabled JAR file on Machine B

You have to place your WLM-enabled EJB application (at least EJB client application) on Machine B (rs600015).

For example:

```
copy /tmp/_wlm_BeenThere.jar /usr/WebSphere/AppServer/deployedEJBs/
```

In our example, we used the BeenThere sample as we mentioned earlier. The WLM-enabled JAR (_wlm_BeenThere.jar) has been placed in the

/usr/WebSphere/AppServer/deployedEJBs directory. Therefore, we actually skipped this step.

The BeenThere sample should be in the <was_dir>/deployedEJBs directory if you are using V3.5.

9.6.2 Step 12: update the bootstrap.properties file on Machine B

For OSE Remote with WebSphere security, update the bootstrap.properties file.

You need to modify the bootstrap.properties file on both the HTTP server and WebSphere node, if you will use WebSphere security for the resources which are on the Web server. Please refer to 5.6, “Configure bootstrap.properties for security” on page 109.

Edit the <was_dir>/properties/bootstrap.properties file on the application server machine (we chose rs600015). Set the property:

```
ose.srvgrp.ibmappserve.clone1.type=remote
```

Based on your requirements, change the clone port property as needed:

```
ose.srvgrp.ibmappserve.clone1.port=unused port number
```

If you specify or change the port, ensure that the port is the same in each bootstrap.properties file on the HTTP server machine. We will configure this later.

```
#############################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=remote
ose.srvgrp.ibmappserve.clone1.host=localhost
ose.mode=out
```

Figure 199. <was_dir>/properties/bootstrap.properties on Machine B

Note that this step is not required if you don’t use WebSphere security for the resources which are on the HTTP server machine.

9.6.3 Step 13: start Administrative Server on Machine B

Now you can start the Administrative Server on Machine B (rs600015).

9.6.4 Step 14: refresh Admin Console to see Machine B

After the Administrative Server on Machine B starts completely, you should refresh your Administrative Console to see it.

Select WebSphere Administrative Domain and click the refresh button.

Then you can see Machine B (rs600015) in the Administrative Console as shown in Figure 200.

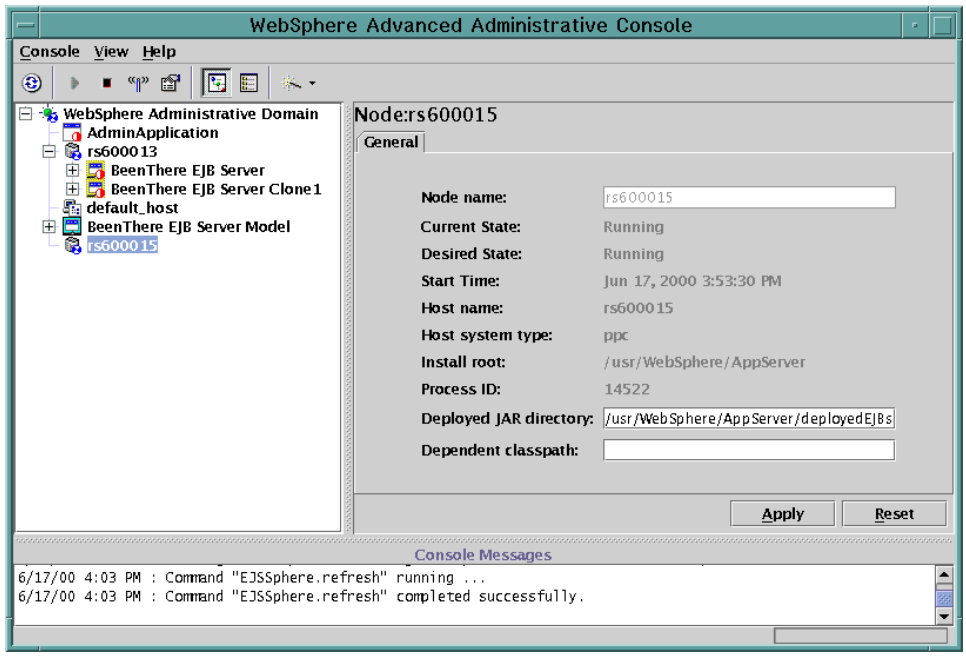

Figure 200. Started WebSphere node for servlet

9.6.5 Step 15: add host aliases

Add aliases for Machine A. These aliases are needed so that the virtual host will accept requests redirected from Machine A with Machine A's URI instead of Machine B's URI. You may add Machine A's hostname (rs600010), IP address, and fully qualified name.

To add the host aliases from the Administrative Console,

Click **default_host** and the **Advanced** tab.

Add aliases under **Host Aliases** and click **Apply** as shown in Figure 201.

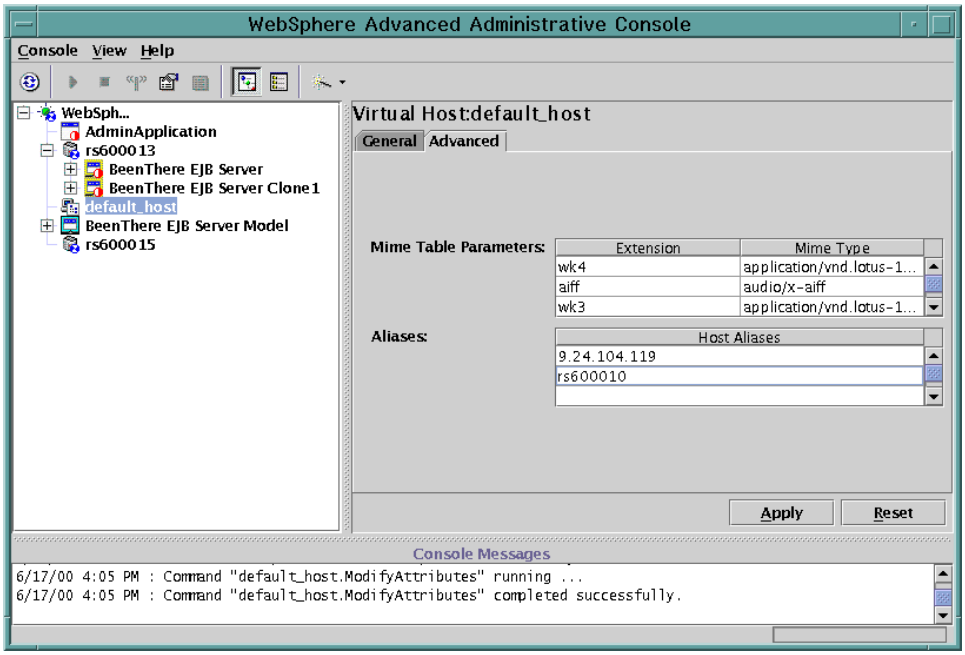

Figure 201. Add aliases

9.6.6 Step 16: create an application server on Machine B

Select Machine B (rs600015) and right click.

Select **Create** and **Application Server** as shown in Figure 202.

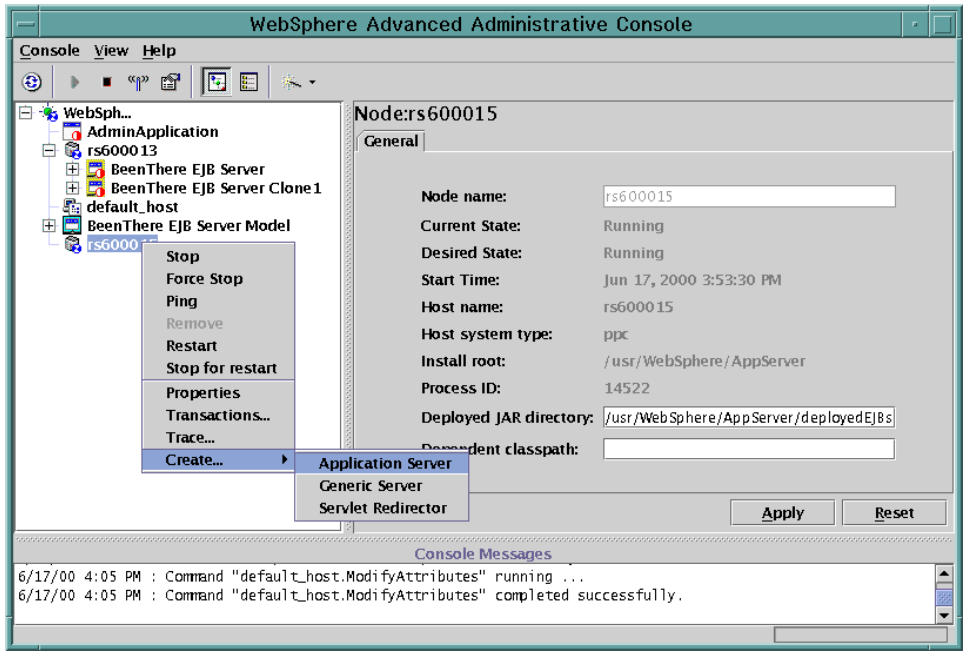

Figure 202. Create application server on Machine B

Then you will get the Create Application Server window.

We specified BeenThere Servlet Server as our application server name.

9.6.6.1 Adding a classpath

In the Command Line Arguments field, we specified the classpath with the WLM-enabled JAR file which contains a WLM-enabled EJB client stub.

For example:

```
-classpath /<was_dir>/deployedEJBs/_wlm_BeenThere.jar
```

as shown in Figure 203.

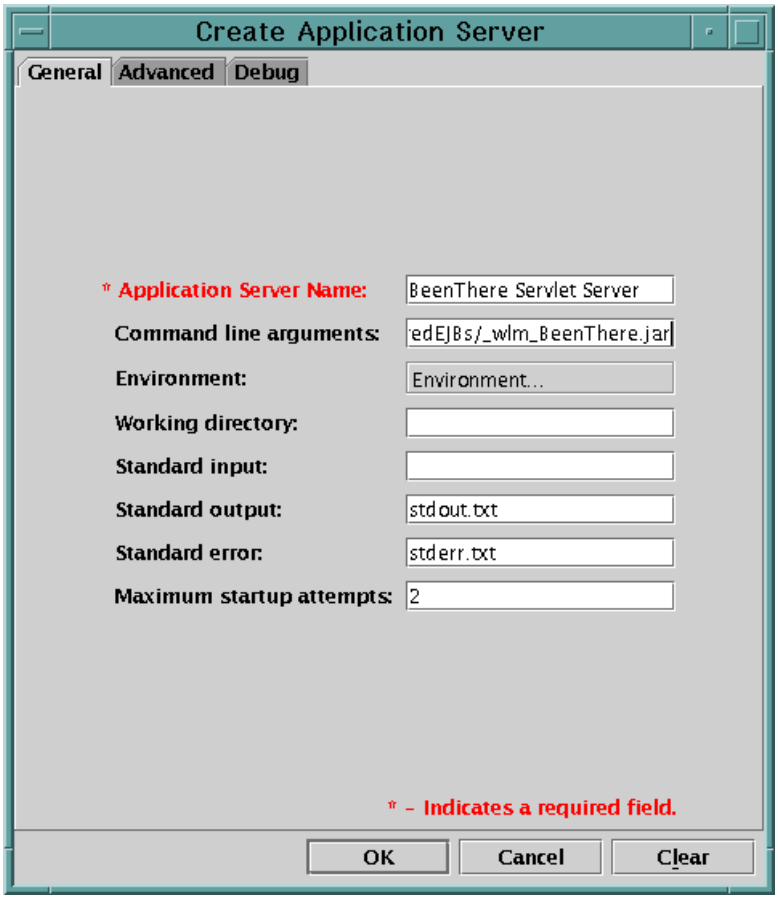

Figure 203. Create application server with classpath for EJB client application

Then click **OK**.

A Note On Classpaths

In WebSphere there are four distinct classloaders. The first of these is defined by the Node Dependent Classpath. As its name would imply this is set at the node level. This is a static classloader that specifies where to find classes that enterprise beans depend on, but that are not located in, the JAR files of the enterprise beans. For example, suppose an Account enterprise bean makes use of a utility class called GeneralLedger, but that class is not in the JAR file that contains the Account bean. The GeneralLedger class should be specified in the dependent classpath to ensure it can be loaded when the enterprise bean depending on it is loaded or deployed. When making a change to the dependent classpath you will need to bring down all application servers on that node.

The second classloader is defined by the Application Server Classpath. This is set as a command line argument for the application server with " -classpath" argument. As with the Node Dependent Classpath this is a static classloader and it provides much same the functionality but it's scope is limited to the application server. This classloader should be used for Frameworks, Utility classes, etc. that will not change frequently or are common to a set of Web Applications running under an application server.

The third classloader is defined by the Web Application Classpath. This is a reloadable classloader and is set for each Web Application. This classpath is used for servlet class files, client JAR files (stubs and I/F), and access beans.

The fourth classloader is for Deployed EJBs. This is a "pseudo static" classloader. New EJBs can be deployed without stopping the container, but replacement of existing EJBs requires a stop and restart. This classpath is used for Deployed EJB JAR files (EJB remote, Home I/F, Bean Implementation, stubs, ties, persistence classes).

In the example above (9.6.6.1, “Adding a classpath” on page 249), the entire JAR file was added to the application server classpath for the purpose of adding the client components to the classpath. Alternatively, this JAR file could have been added to the Web Application Classpath. If code for this example had been developed with VisualAge for Java, the client JAR file could have been placed on either of the noted classpaths instead of the entire JAR file.

9.6.7 Step 17: create a servlet engine on Machine B

Select Machine B (rs600015) and right click.

Select **Create** and **Servlet Engine** as shown in Figure 204.

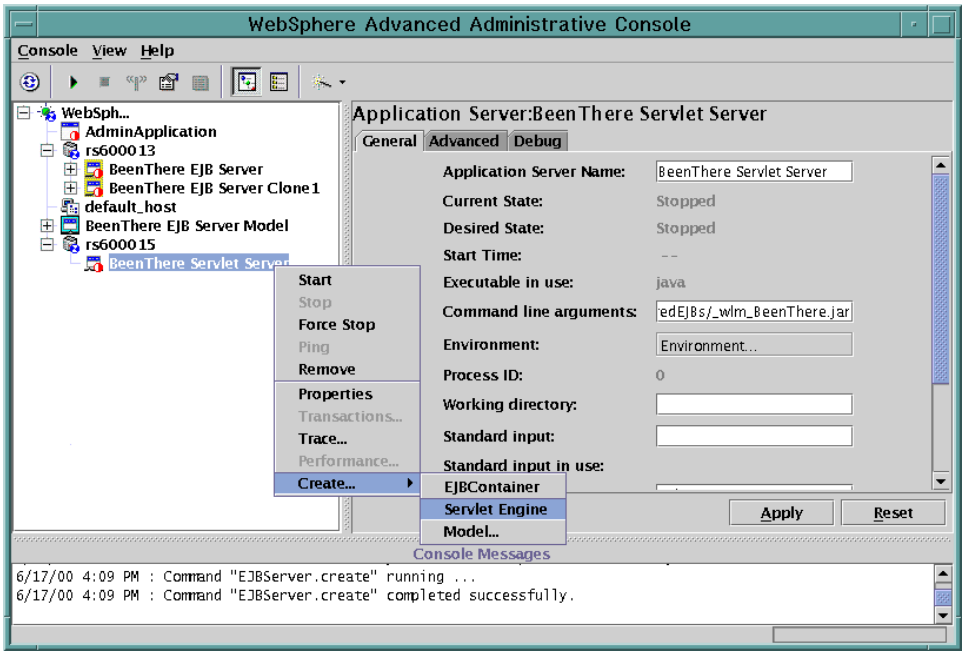

Figure 204. Create application server on Machine B

Then you will get the Create Servlet Engine window.

In our example, we specified BeenThere Servlet Engine as our servlet engine name as shown in Figure 205.

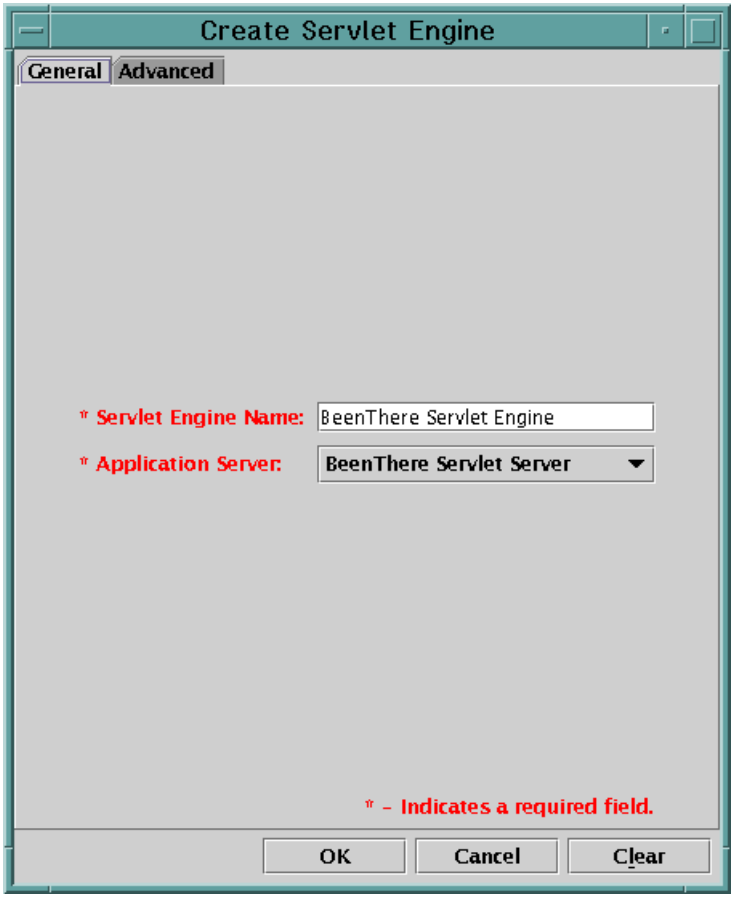

Figure 205. Create servlet engine on Machine B

Then click **OK**.

9.6.8 Step 18: update transport type for OSE Remote

The servlet engine needs to be configured to use sockets instead of local pipes. You need to specify INET sockets for the transport type of OSE queue. This is the default on Solaris machines.

Select the servlet engine (BeenThere Servlet Engine in our example) of the Application Server (BeenThere Servlet Server in our example) under Machine B (rs600015) as shown in Figure 206.

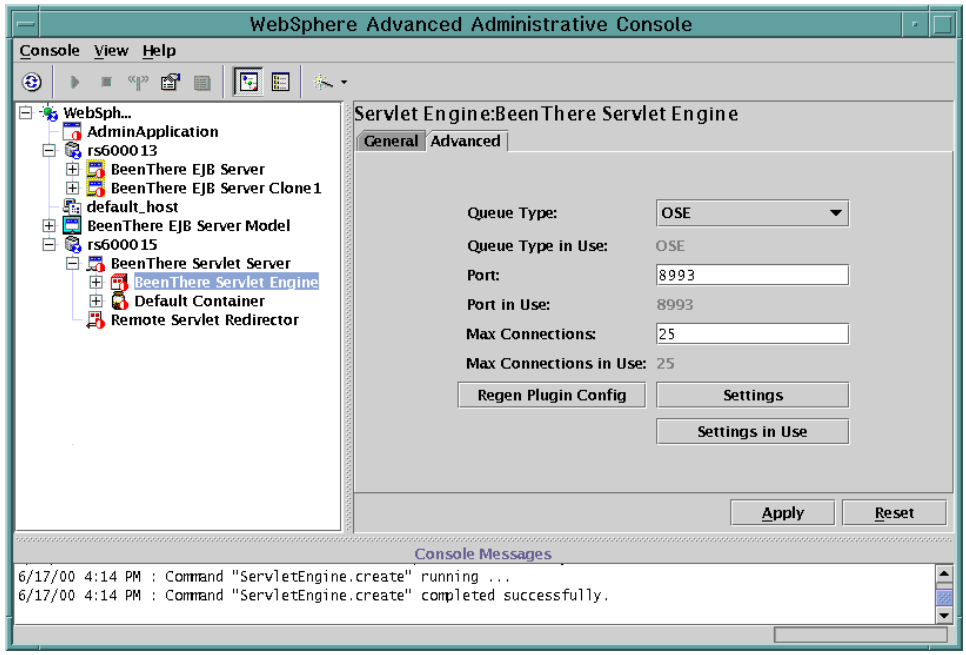

Figure 206. Servlet engine on Machine B

Select the **Advanced** tab inside the Servlet Engine panel, and choose **OSE** for the Queue Type and click **Settings**.

You will get the Edit Servlet Engine Transport window. Then choose **INET Sockets** for Transport Type as shown in Figure 207.

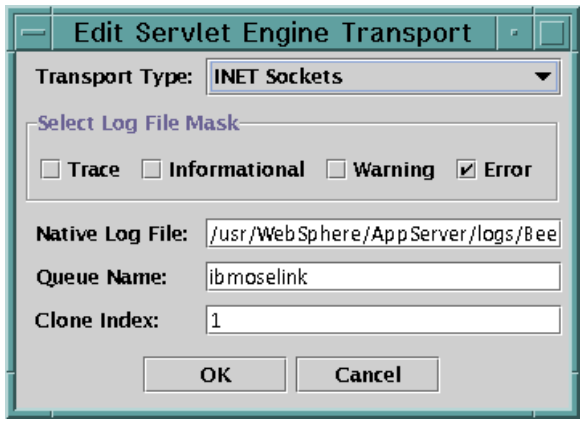

Figure 207. Edit servlet engine transport

Then click **OK**. You will be returned to the Administrative Console.

Click the **Apply** button. The servlet engine is now updated.

9.6.9 Step 19: create Web application on Machine B

Create the Web application in the servlet engine which you created in the previous step.

Select the servlet engine you just created (in our example, BeenThere Servlet Engine), and right click.

Select **Create** and **Web Application** as shown in Figure 208.

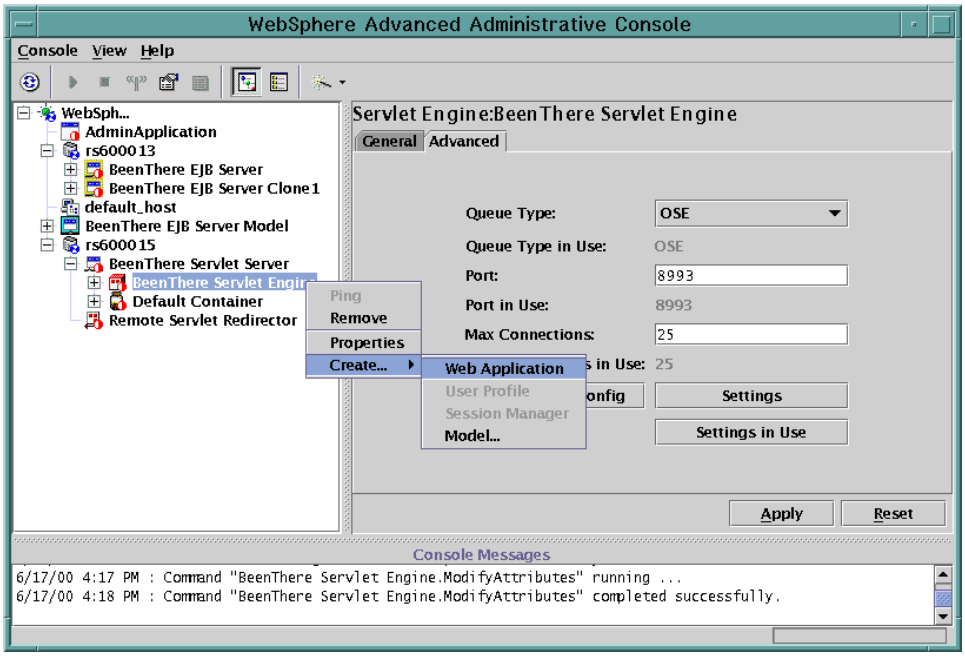

Figure 208. Create Web application on Machine B

Then you will get the Create Web Application window.

In our example, we specified “examples” as our Web application name as shown in Figure 209.

Click **OK**.

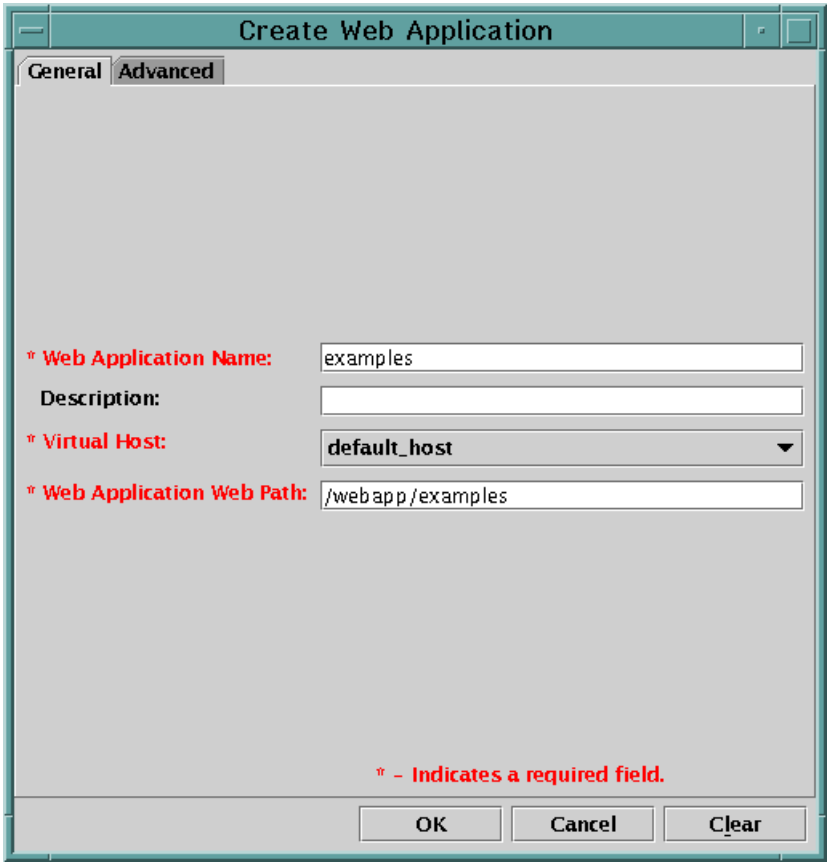

Figure 209. Create Web application on Machine B

9.6.10 Step 20: create servlet on Machine B

Now, you can create your servlet on Machine B (rs600015).

Select the Web application which you just created (in our example, examples) and right click.

Select **Create** and **Servlet** as shown in Figure 210.

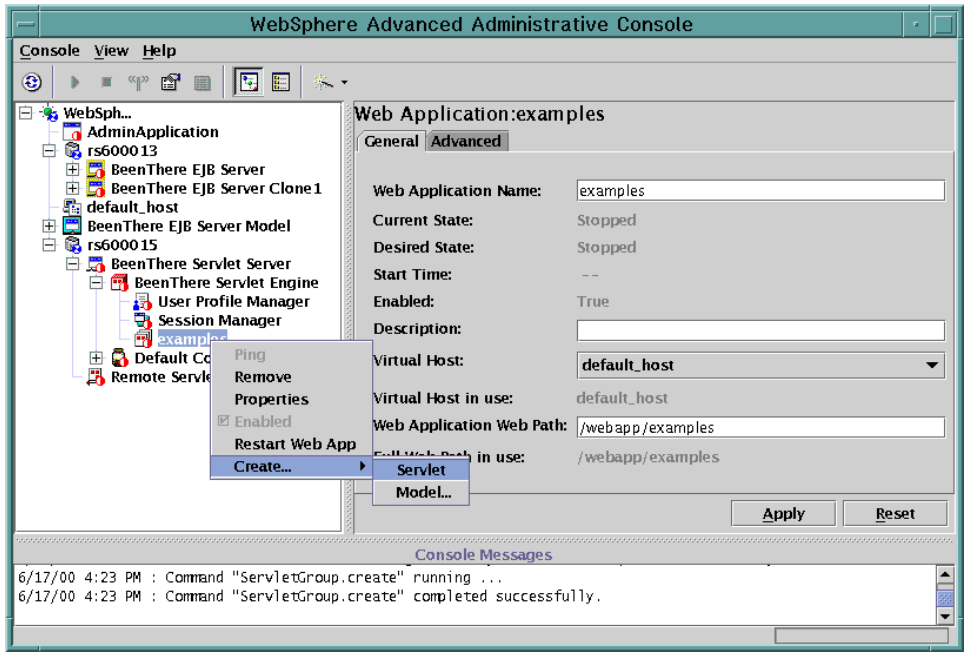

Figure 210. Create servlet on Machine B

Then you will get the Create Servlet window.

In our example, we specified “BeenThere” as Servlet Name and “BeenThereServlet” as Servlet Class Name as shown in Figure 211.

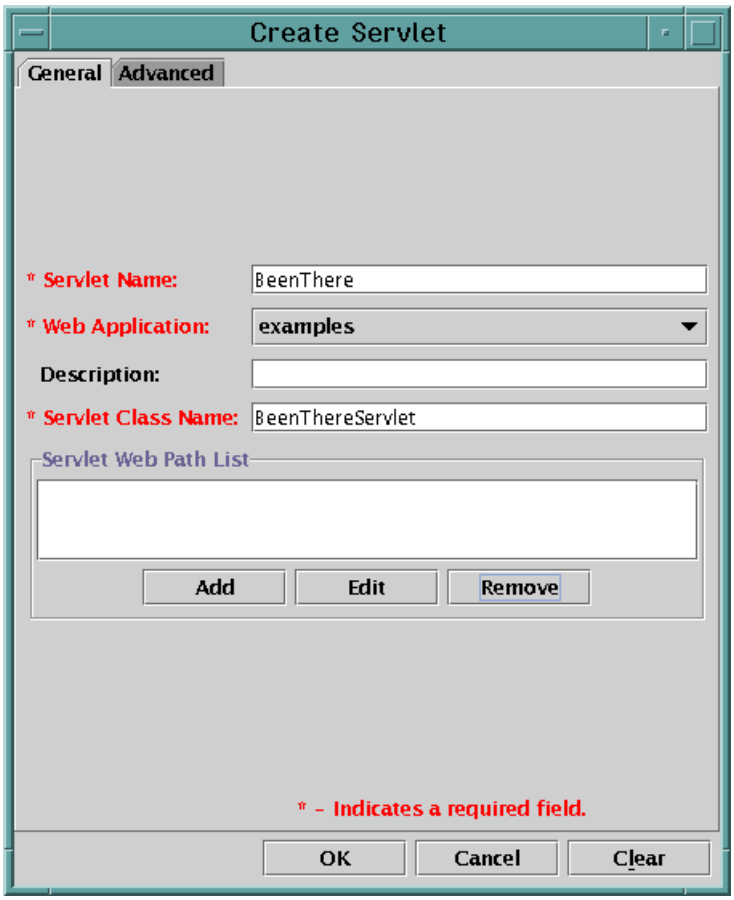

Figure 211. Specify servlet name and servlet class name

Then click **Add** to specify the Servlet Web Path.

You will get the Add Web Path to Servlet window.

In our example, we specified “/webapp/examples/BeenThere” as Servlet Path as shown in Figure 212. Click **OK**.

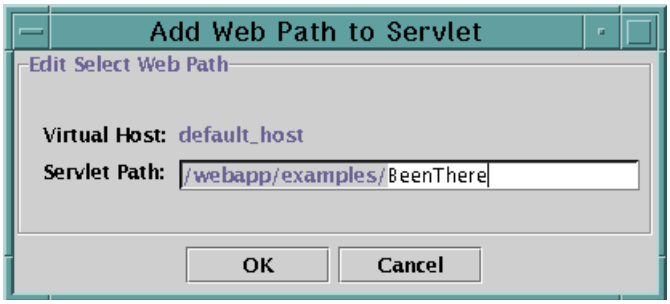

Figure 212. Add Web path to servlet

You will get back to the Create Servlet window.

Please verify that Servlet Name, Web Application, Servlet Class Name and Servlet Web Path List are specified as shown in Figure 213.

Click **OK**.

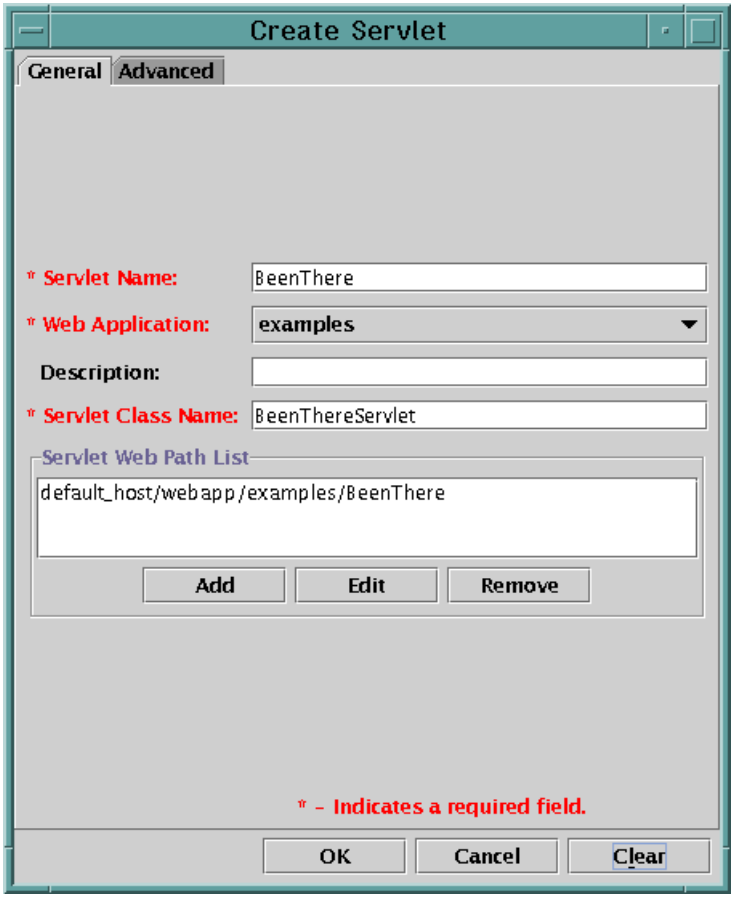

Figure 213. Verify the servlet information which you specified

Now you can see the servlet which you just created in the Administrative Console as shown in Figure 214.

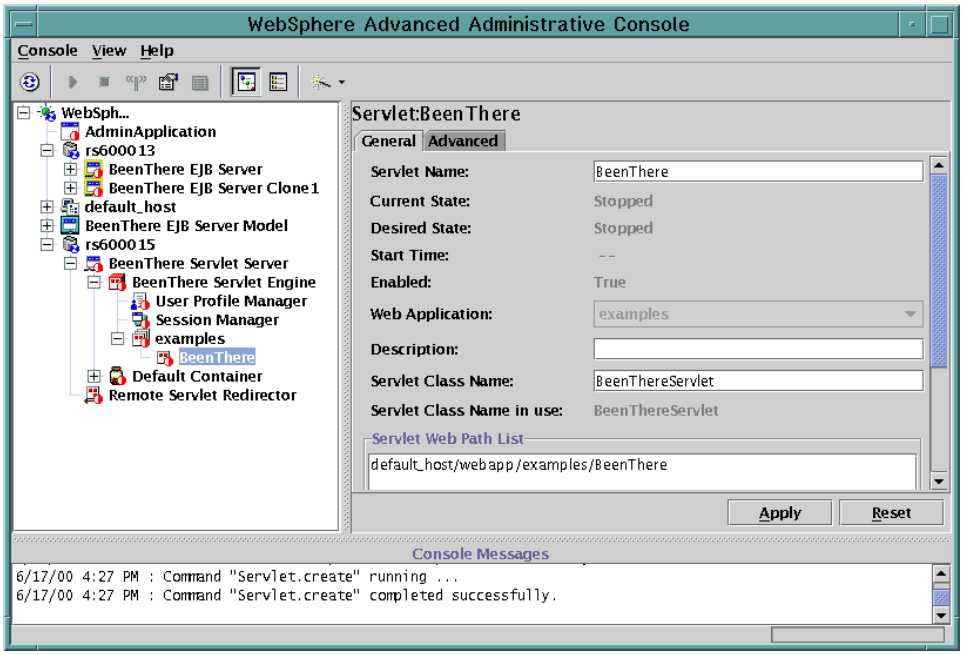

Figure 214. A new servlet is in the Administrative Console

9.6.11 Step 21: create model on Machine B

In the Administrative Console, expand the node (rs600015). Locate the application server in the list. For our sample configuration, we used BeenThere Servlet Server as our application server.

To create the model, right click on the application server to be cloned, and select **Create ->Model** as shown in Figure 215.

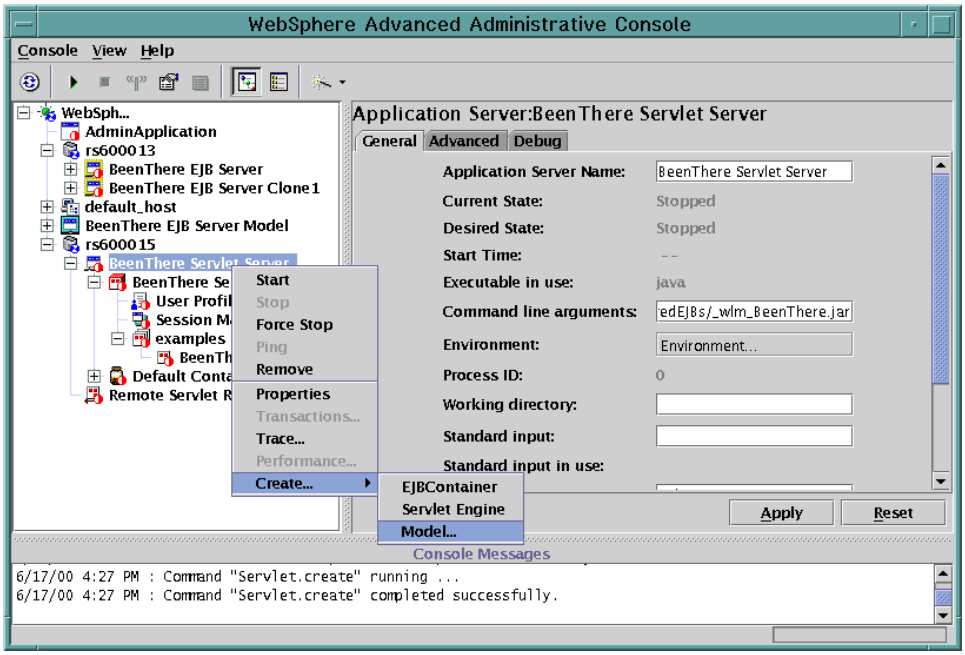

Figure 215. Create servlet application server model

The Model Properties window will appear. Enter a model name and check the box to both Make Server a Clone and Recursively Model all Instances under the Server as shown in Figure 216. Click **OK**.

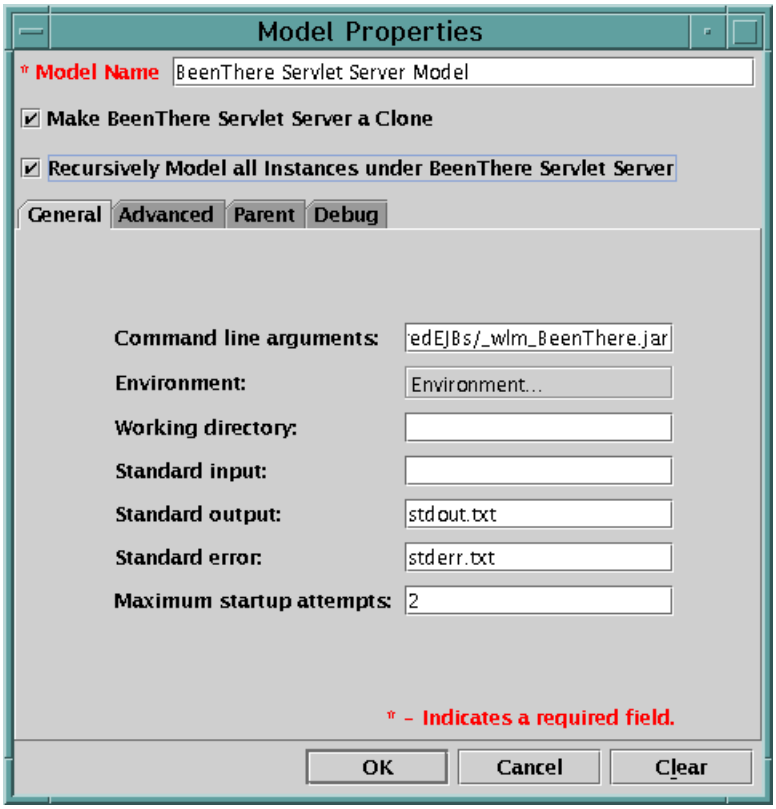

Figure 216. Model Properties for BeenThere Servlet Server Model

9.6.12 Step 22: create clones on Machine B

Using the model you created (in our case, BeenThere Servlet Server Model), you can now create a clone. To do this, right click on the model and select **Create->Clone** as shown in Figure 217.

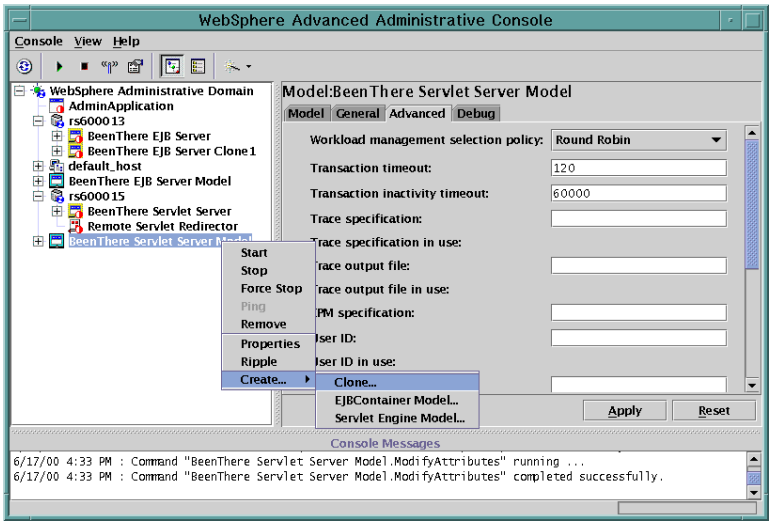

Figure 217. Create clone on Machine B

The Clone Parent window will appear. Enter a clone name (in our case, BeenThere Servlet Server Clone1) and select the node for the clone (in our case, rs600015) as shown in Figure 218. Then click the **Create** button.

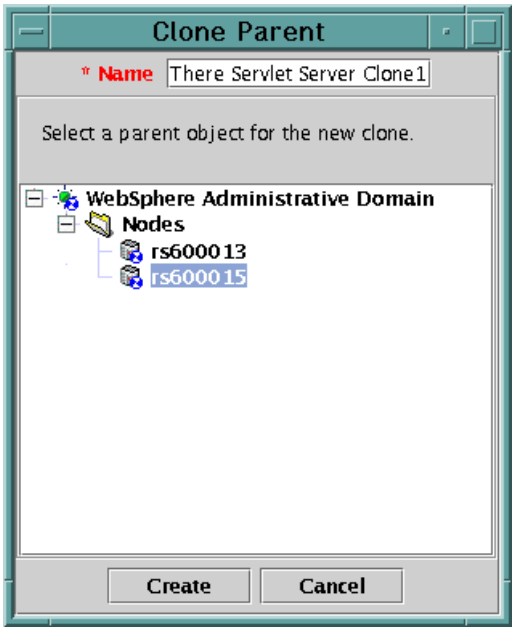

Figure 218. Clone parent

Now you can see the application server (BeenThere Servlet Server) and its clone (BeenThere Servlet Server Clone1) in the Administrative Console as shown in Figure 219.

Note

If you selected the option to make the server a clone when you generated the model, you will already have one clone associated with the model (the server which you cloned).

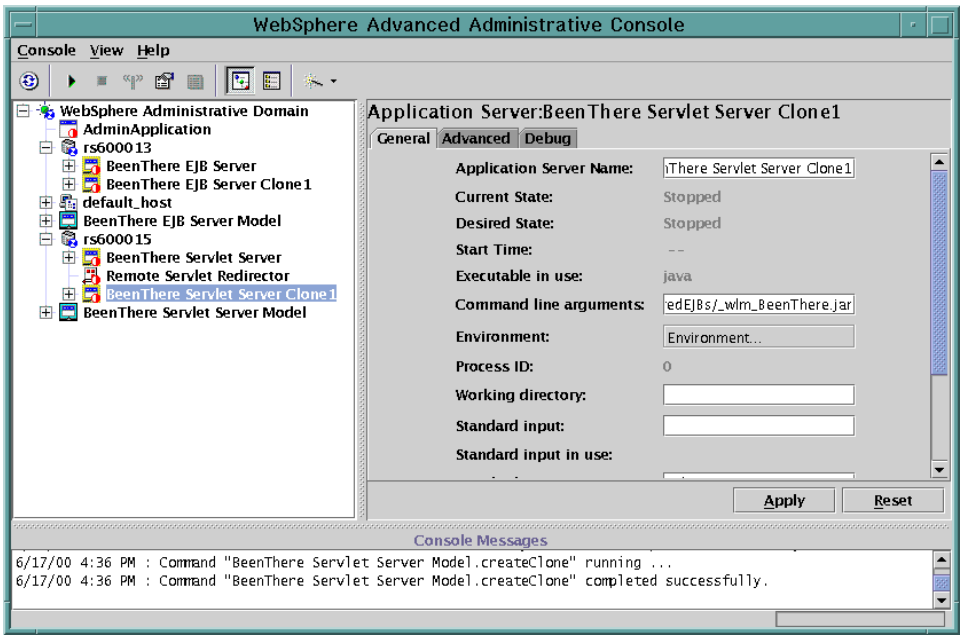

Figure 219. BeenThere Servlet Server and its clone in the Administrative Console

9.7 Starting models

9.7.1 Step 23: start EJB application server model on Machine C

Starting the model will start all the clones associated with the model.

Start the model (BeenThere EJB Server Model) by clicking the **Start** button  as shown in Figure 220.

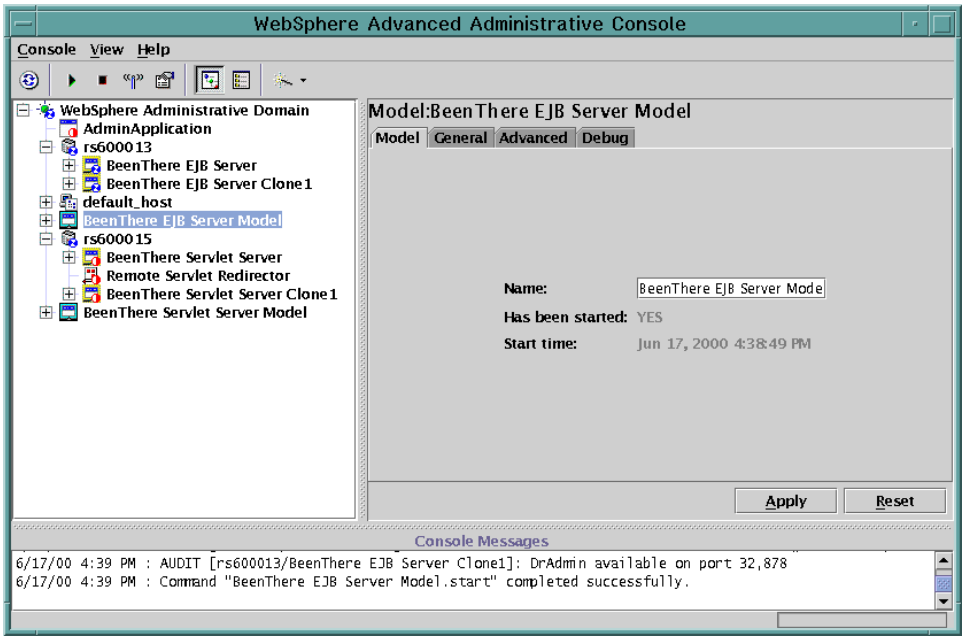

Figure 220. Start EJB model

9.7.2 Step 24: start servlet application server model on Machine B

Start the model (BeenThere Servlet Server Model) by clicking the **Start** button  as shown in Figure 221.

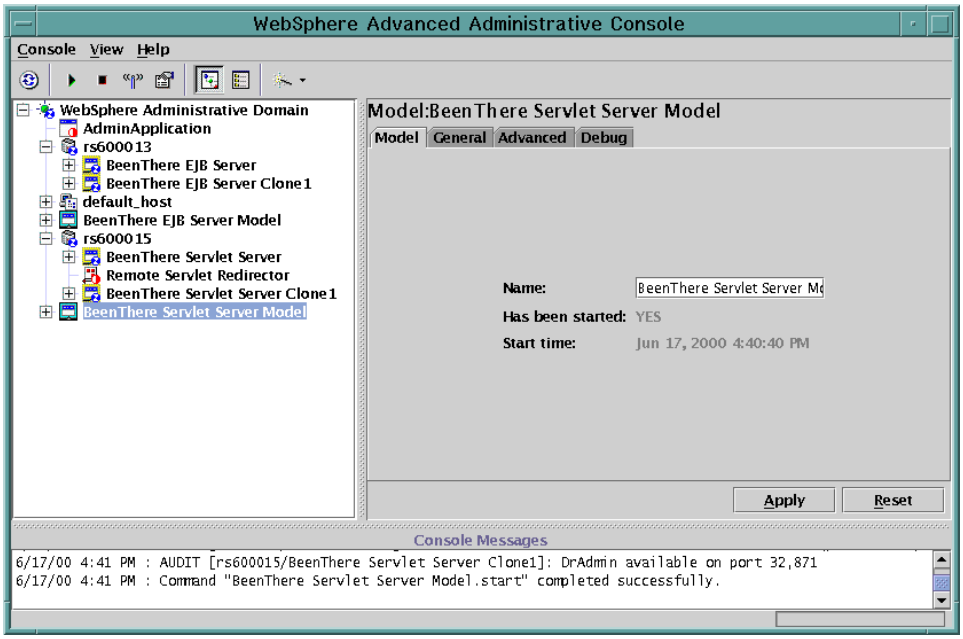

Figure 221. Start servlet model

9.8 OSE Remote configuration

Now, you are ready to configure OSE Remote for communication between the HTTP server on Machine A and servlet engines on Machine B.

9.8.1 Step 25: update the bootstrap.properties file on Machine A

You need to modify the bootstrap.properties file on both Machines A and B, if you use WebSphere security for the resources which are on the Web server.

Edit the <was_dir>/properties/bootstrap.properties file on the HTTP server machine (rs600010). Set the following properties:

```
ose.srvgrp.ibmappserve.clone1.port=8081
```

Make sure that the same entries are in the bootstrap.properties file on the WebSphere machine, if you need to change it.

```
ose.srvgrp.ibmappserve.clone1.type=remote

ose.srvgrp.ibmappserve.clone1.host=<admin server hostname>
```

```
##############################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.conncurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8081
ose.srvgrp.ibmappserve.clone1.type=remote
ose.srvgrp.ibmappserve.clone1.host=rs600015
ose.mode=out
```

Figure 222. <was_dir>/bootstrap.properties on Machine A

Please refer to 5.6, “Configure bootstrap.properties for security” on page 109.

9.8.2 Step 26: update plug-in configuration properties on Machine A

The WebSphere plug-in for the HTTP server uses three properties files generated by WebSphere to define its configuration:

1. queues.properties
2. rules.properties
3. vhosts.properties

These files need to be generated on the machine that will be used as the HTTP server.

Please refer to 5.5, “Configure the plug-in for OSE Remote” on page 103 for detailed information.

Here are the sample configurations for OSE Remote which we used in our example.

```
#IBM WebSphere Plugin Communication Queues
#Sat Jun 17 16:44:52 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.port=8993
ose.srvgrp.ibmoselink.clone2.port=8994
ose.srvgrp.ibmoselink.clone1.host=rs600015
ose.srvgrp.ibmoselink.clone2.host=rs600015
```

Figure 223. <was_dir>/temp/queues.properties on Machine A (rs600010)

```
#IBM WebSphere Plugin URL Mapping Rules
#Sat Jun 17 16:44:52 EDT 2000
default_host/webapp/examples/BeenThere=ibmoselink
```

Figure 224. <was_dir>/temp/rules.properties on Machine A (rs600010)

```
#IBM WebSphere Plugin Virtual Host Mappings
#Sat Jun 17 16:44:52 EDT 2000
9.24.104.119=default_host
rs600013=default_host
rs600010=default_host
localhost=default_host
127.0.0.1=default_host
```

Figure 225. <was_dir>/temp/vhosts.properties on Machine A (rs600010)

9.8.3 Step 27: start HTTP Server on Machine A

Finally, you need to start the HTTP Server on Machine A. If it's already started, stop and restart it to reload the latest configuration information.

9.9 Test the servers

Now that the HTTP server, WebSphere servlet model, and WebSphere EJBs model are running, we need to test the servers to make sure that everything is configured properly.

Using a Web browser, access the following URI (rs600010 is the HTTP server and /webapp/examples/BeenThere is the servlet):

```
http://rs600010/webapp/examples/BeenThere
```

Figure 226 shows the result of the request.

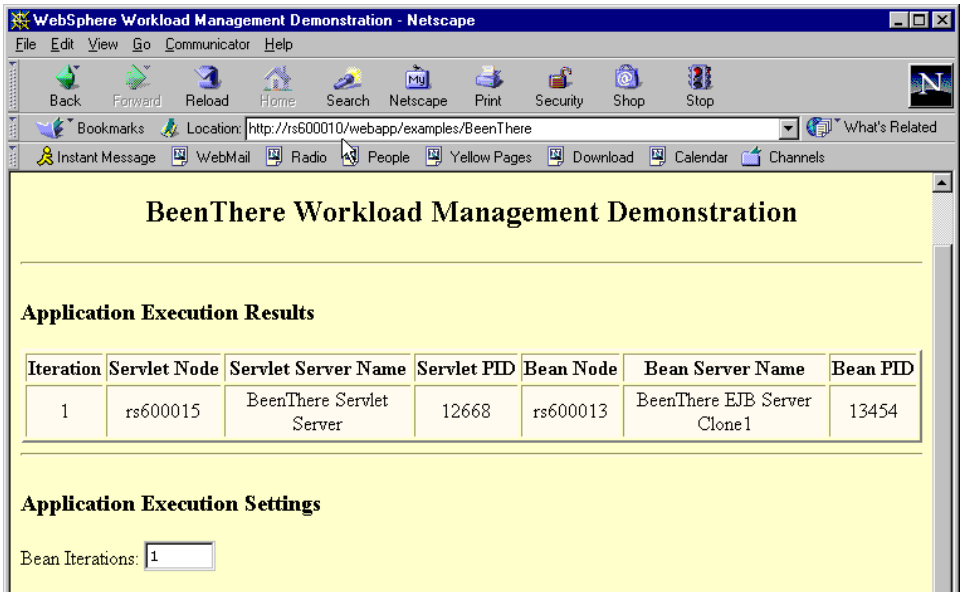

Figure 226. Hit BeenThere Servlet Server and BeenThere EJB Server Clone1

This output tells you that:

1. The browser sent a request to Machine A (rs600010).

   We specified the URL (`http://rs600010/webapp/examples/BeenThere`).

2. The servlet in the BeenThere Servlet Server on Machine B (rs600015) received the request.
3. The EnterpriseBean in the BeenThere EJB Server Clone1 on Machine C (rs600013) received the request.

Figure 227 shows the result of the next request.

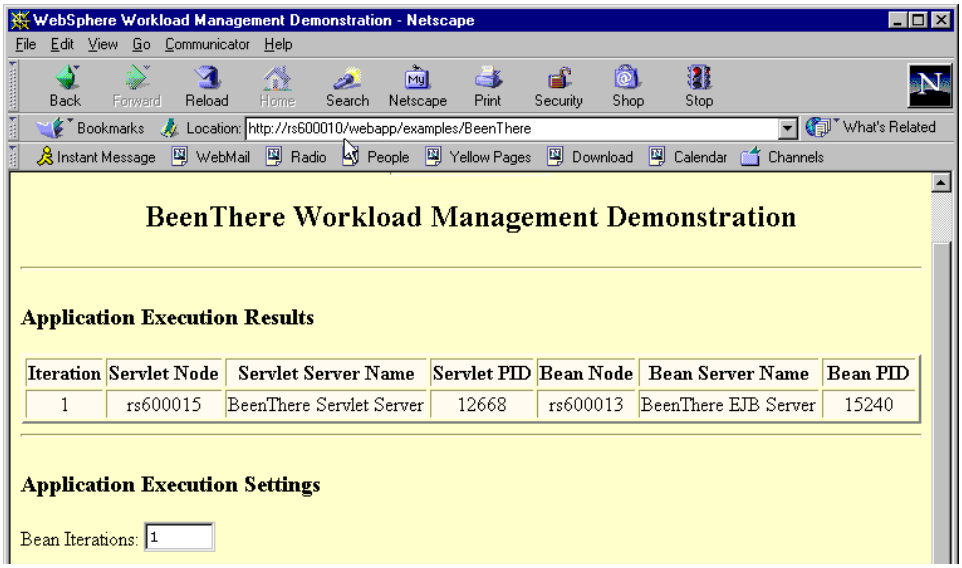

Figure 227. Hit BeenThere Servlet Server and BeenThere EJB Server

This output tells you that:

1. Browser sent a request to Machine A (rs600010).

   We specified the URL (`http://rs600010/webapp/examples/BeenThere`).

2. The servlet in the BeenThere Servlet Server on Machine B (rs600015) received the request.

3. The EnterpriseBean in the BeenThere EJB Server on Machine C (rs600013) received the request.

Figure 228 shows the result of the next request.

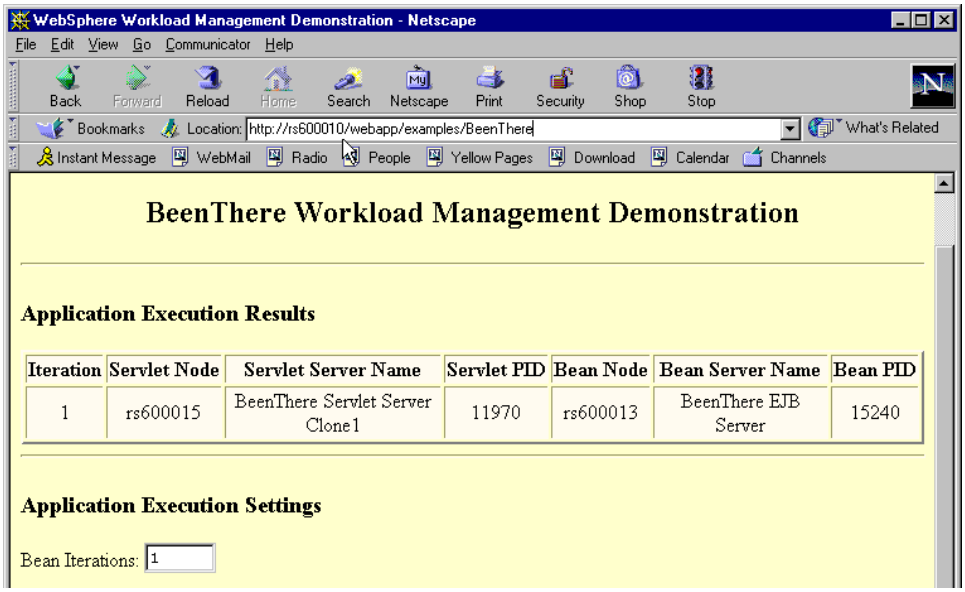

Figure 228. Hit BeenThere ServletServer Clone1 and BeenThere EJB Server

This output tells you that:

1. Browser sent a request to Machine A (rs600010).

   We specified the URL (`http://rs600010/webapp/examples/BeenThere`).

2. The servlet in the BeenThere Servlet Server Clone1 on Machine B (rs600015) received the request.

3. The EnterpriseBean in the BeenThere EJB Server on Machine C (rs600013) received the request.

Figure 229 shows the result of the next request.

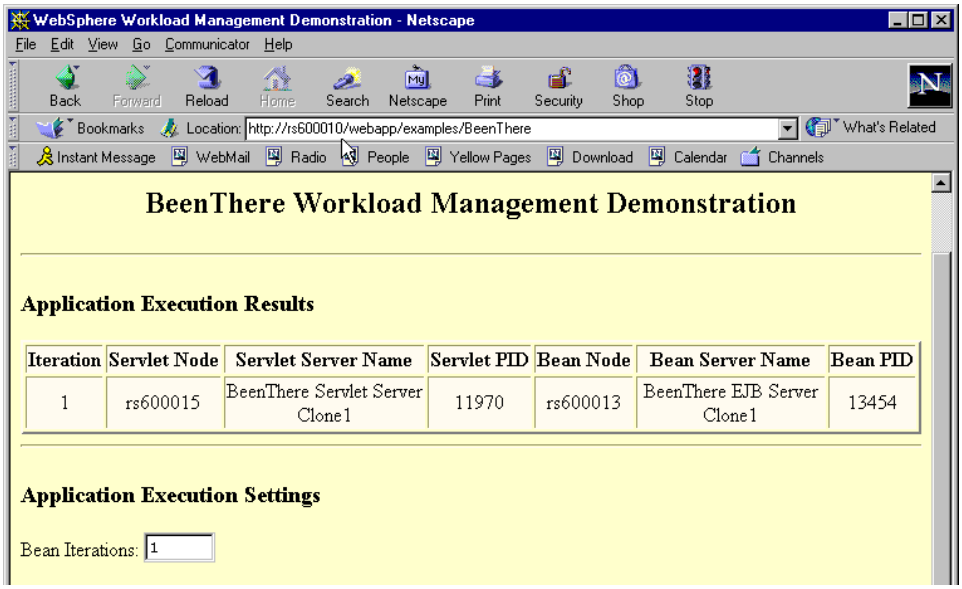

Figure 229. Hit BeenThere Servlet Server Clone1 and BeenThere EJB Server Clone1

This output tells you that:

1. Browser sent a request to Machine A (rs600010).

   We specified the URL (`http://rs600010/webapp/examples/BeenThere`).

2. The servlet in the BeenThere Servlet Server Clone1 on Machine B (rs600015) received the request.

3. The EnterpriseBean in the BeenThere EJB Server Clone1 on Machine C (rs600013) received the request.

9.9.1 Related topology: multiple EJB application server nodes

You can configure this topology to extend it to the previous topology. You will need to create a clone for the EJB application server on the remote node.

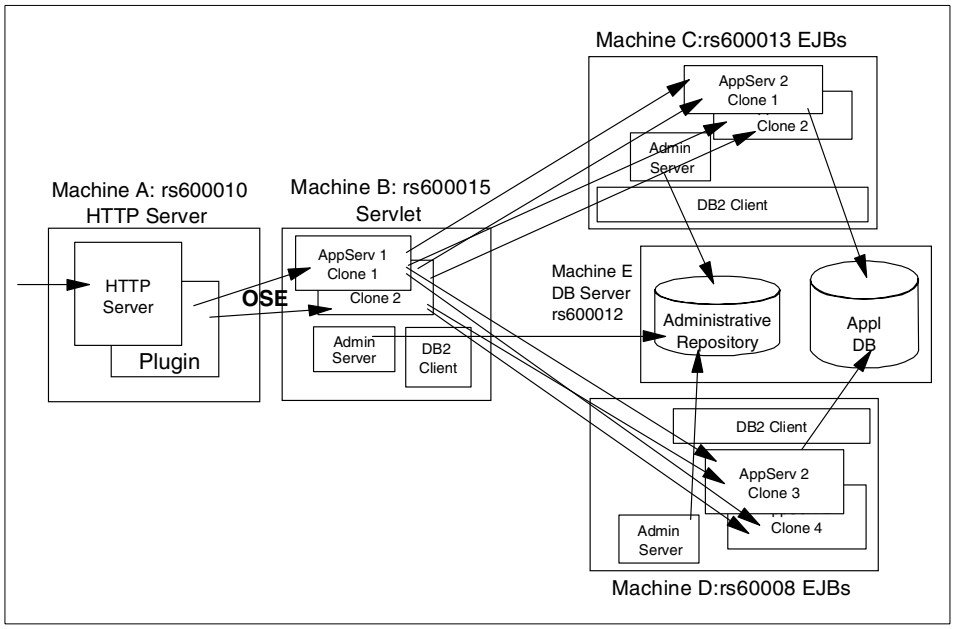

Figure 230. Three-tier WebSphere application topology with multiple EJB application servers

9.10 Maintenance

There are some considerations when performing maintenance on an EJB application server.

9.10.1 Terminating an EJB application server

When an EJB application server aborts suddenly (we used the `kill -9 PID` command to simulate this situation), WebSphere automatically restarts the server. No action is required.

WLM-enabled EJB clients will automatically detect this failure and route new requests to alternative serves. However, some Web clients may experience brief transient errors.

If there are no clones of this server, no requests will be served and errors will be reported until the application server restarts.

9.10.2 Stopping an EJB application server

When an EJB application server is stopped for administrative reasons (via the Administrative Console or from the command line), WLM-enabled EJB clients will automatically route new requests to alternative servers.

On rare occasions, while a server is terminating, it is possible that some number of WLM-enabled clients may experience transient errors before the client requests are dispatched to an alternate server.

Of course WLM can only dispatch requests to alternate servers for application servers that are part of a model. If an the application server has no clones, no client requests will succeed.

9.10.3 Restart an EJB application server

For maintenance or any other reason, you might stop and restart your application server. When the application server is restarted, browsers automatically get access to it. No additonal actions are required.

9.10.4 Starting a new EJB application server clone

When you need to create additional clones, start them after you start the application server. No additional action is required.

9.10.5 Restart the Admin Server on an EJB application server node

When you restart the Administrative Server on an EJB application server node, the EJB client will be notified as discussed in Chapter 2, “Techniques for WebSphere WLM and clustering” on page 15. For quicker notification, you can restart the EJB client (in our example, the servlets engine).

9.10.6 Restarting an EJB machine

With multiple clone/multiple node environments, when a node is restarted, the servlets will access it. If a quicker turn-around is needed, the refresh can be forced by restarting the servlets.

9.10.7 Ripple Mode

In V3.5 when you administer a model (highlight a model, right mouse button), one of the options presented is “ripple”. This option will “ripple” through all the clones in the model, stopping and starting them. This allows one to make a change to a model (change a class of jar file for an existing app, or add a new component), then “ripple” the model, so that all running instances of the application server are stopped and restarted so that the new code is used.

Chapter 10. Horizontal scaling with IBM Network Dispatcher

Unlike the previous chapters, this environment utilizes an external load balancing mechanism, the *IBM Network Dispatcher* (ND). This technology improves the performance of servers by distributing the incoming TCP/IP requests, in our case HTTP requests, amongst a group of servers typically known as a cluster. The *IBM Network Dispatcher* product utilizes intelligent load balancing, taking into account the availability, capability, workload and user definable criteria when determining which server the TCP/IP request is sent to.

This chapter provides instructions for the administrative tasks required to configure a WebSphere Administrative Server to support multiple cloned application servers, accessing a remote Database server for the Administrative Repository and application database.

We will discuss the following steps for setting up this configuration:

1. Configuration overview
2. Installation summary
3. Creating the ND environment
4. Starting the Administrative Servers and console
5. Updating host aliases
6. Creation of the models
7. Changing the WLM
8. Creating clones
9. Starting the servers
10. Testing the configuration
11. Related topologies

10.1 Overview of the configuration

A Web server and an Administrative Server are installed on Machine A (rs60008) and Machine B (rs600013). The WebSphere Administrative Servers on Machine A and Machine B need access to the shared Administrative Repository on Machine D (rs600012). Network Dispatcher is installed on Machine C (rs60007) as shown in Figure 231.

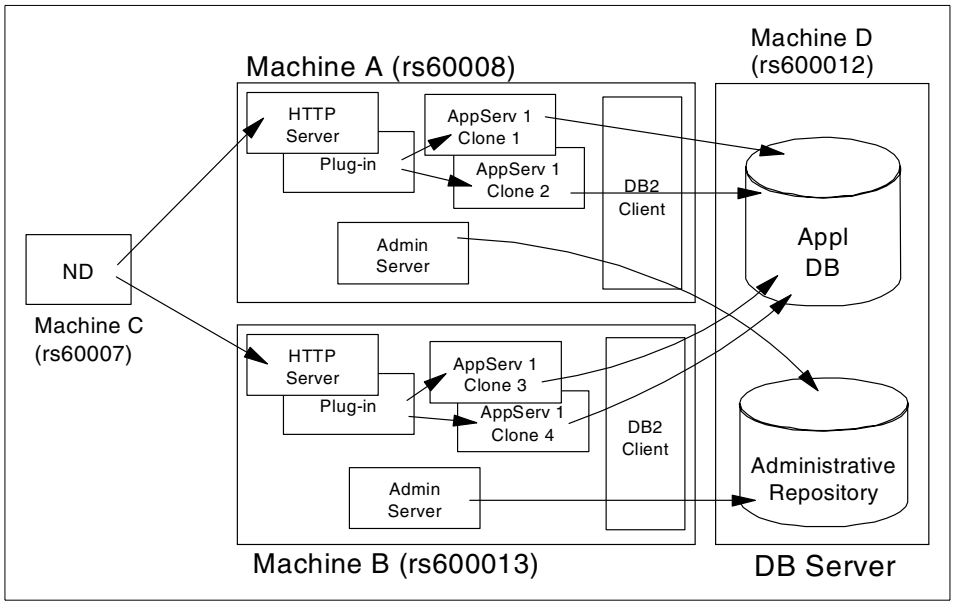

Figure 231. Network Dispatcher with two WebSphere nodes in a single WebSphere domain

10.2 Installation summary

Table 10 summarizes the installation steps to configure this topology.

Table 10. Installation steps

Step No.	Machine C (ND)	Machine A & B (WAS)	Machine D (DB)
1			Install DB2 UDB Apply appropriate FixPack
2			Create DB
3		Install DB2 client Apply appropriate FixPack	
4		Catalog node Catalog db	
5	Install JDK	Install JDK (WebSphere 3.02x)	
6		Install Web server	

Step No.	Machine C (ND)	Machine A & B (WAS)	Machine D (DB)
7		Install WebSphere	
8	Install ND		

10.3 Creating the ND environment

This technology improves the performance of servers by distributing the incoming TCP/IP requests, in our case HTTP requests, amongst a group of servers typically known as a cluster. Network Dispatcher utilizes intelligent load balancing, taking into account the availability, capability, workload and user definable criteria when determining which server the TCP/IP request is sent to.

10.3.1 Configuring Network Dispatcher

The configuration of *IBM Network Dispatcher* is a complex process beyond the scope of this redbook. For information on using the *IBM Network Dispatcher* to load balance in your environment, refer to the following redbook:

IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher, SG24-5858

With our scenarios, the IBM Network Dispatcher runs on an RS/6000 AIX system. We have included our sample scripts in Appendix G, “Sample Network Dispatcher configuration script” on page 545.

10.4 Starting WebSphere application servers

1. Start the DB2 instance for the Administrative Repository on Machine D if it’s not started.
2. Start WebSphere Administrative Server on Machine A.
3. Start WebSphere Administrative Server on Machine B.
4. Start WebSphere Administrative Console on Machine A.

Note: You can use any machine for your Administrative Console.

You will see both Machines A and B in the Topology pane as shown in Figure 232.

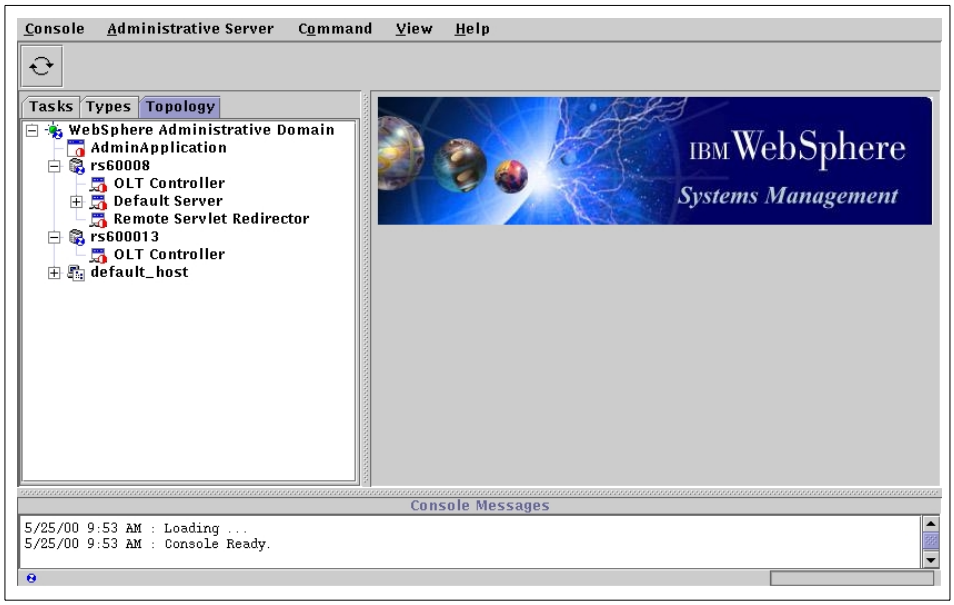

Figure 232. Administrative Console

10.5 Adding the host aliases

When utilizing *IBM Network Dispatcher* a common base URI is used to access the resources of all nodes in the cluster. In this scenario we used `www.itso.ibm.com`. We updated the host alias information to reflect this as shown in Figure 233 on page 281.

Note

Unless a requirement exists to utilize a distinct HTTP server outside of the IBM Network Dispatcher environment, for example testing scenarios, it is recommended you remove the host alias information for the individual nodes.

However, if you want to run a WAS custom advisor, you must have your HTTP servers' information in the WebSphere host alias information.

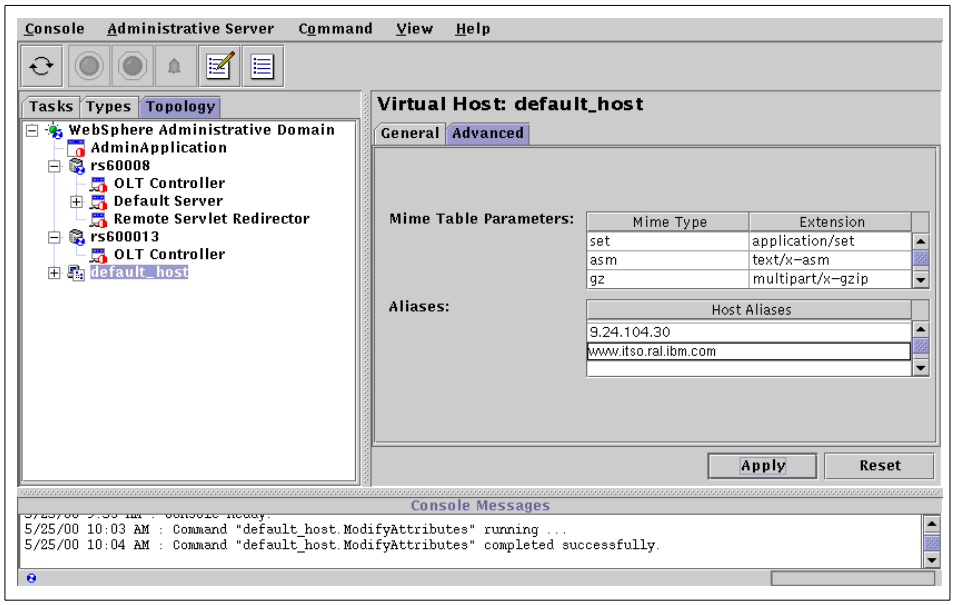

Figure 233. Adding the host aliases

10.6 Creating the model

From the Topology pane of the WebSphere Administrative Console, select the Web application you wish to clone (our case, Default Server), and right click then select **Create->Model** as shown in Figure 234.

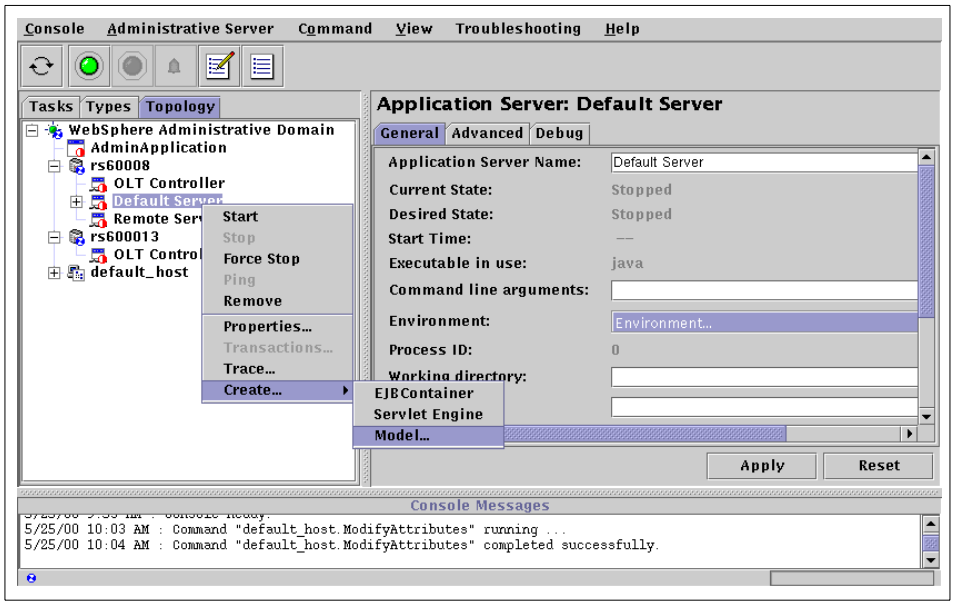

Figure 234. Create model

In the Clone properties dialog box that is displayed, enter your chosen model name (our case, Default Server Model) and check the box to Recursively Model all Instance under Default Server as shown in Figure 235.

Note

Typically, you would not want to covert your initial Web application into a clone; however if you plan to enhance this topology to utilize the Servlet Redirector then you should tick the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model's clones to fail unless this server is made into a clone.

If you are using V3.5, you should tick the option to make this server a clone. If you don't, you cannot create a clone.

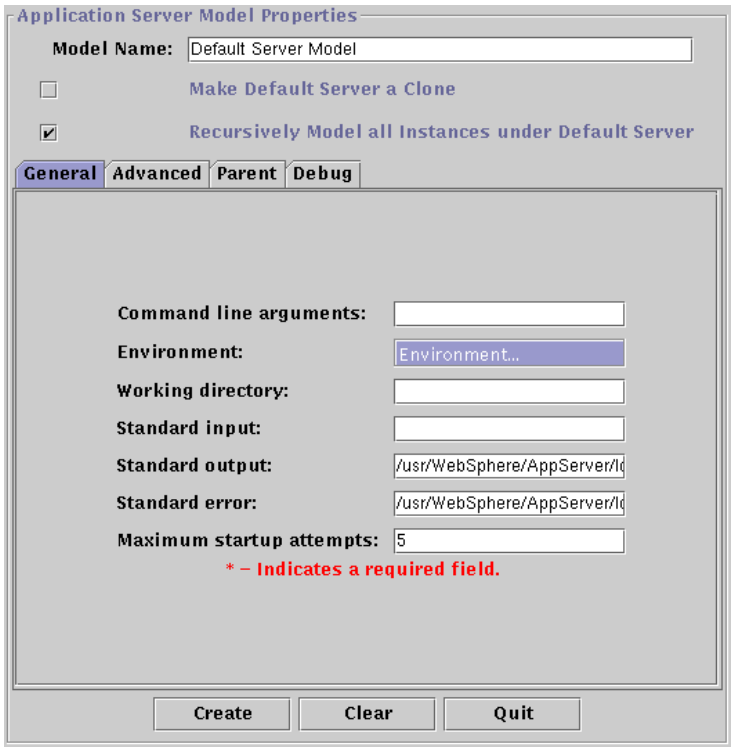

Figure 235. Clone properties: general

At this time you may wish to change the workload management selection policy. To do this, click the **Advanced** tab and you will be presented with this option, as shown in Figure 236 on page 284.

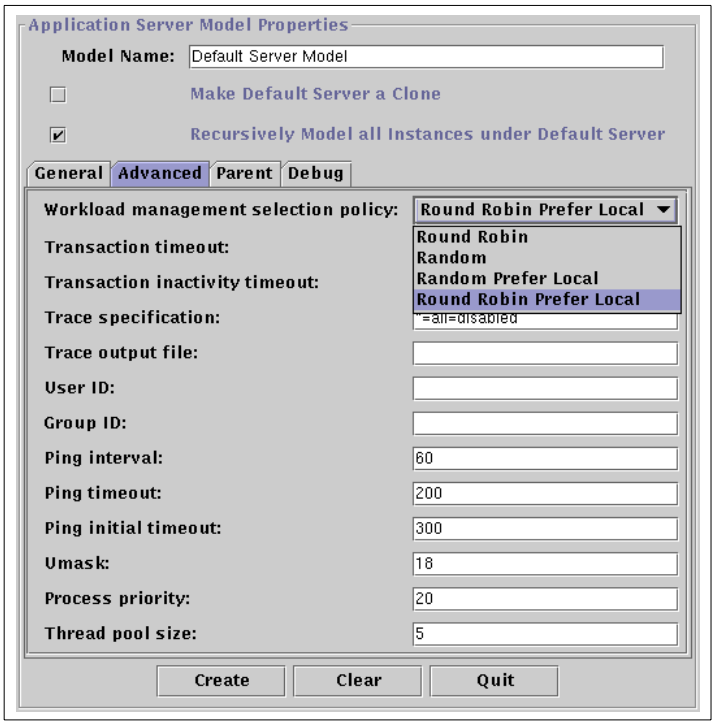

Figure 236. Clone properties: advanced

Note

Currently, with V3.021 any changes you make to the workload management selection policy from this tab during model definition are not reflected in the generated model.

To change the workload management selection policy, you should change the properties of the resultant model.

After completing any changes you wish to make, click the **Create** button to generate the model.

When the model has been created, you will see the following message.

Figure 237. Model created message

After completing your configuration, you should see your model in the Topology tree surrounded by a cyan square, used to identify a model.

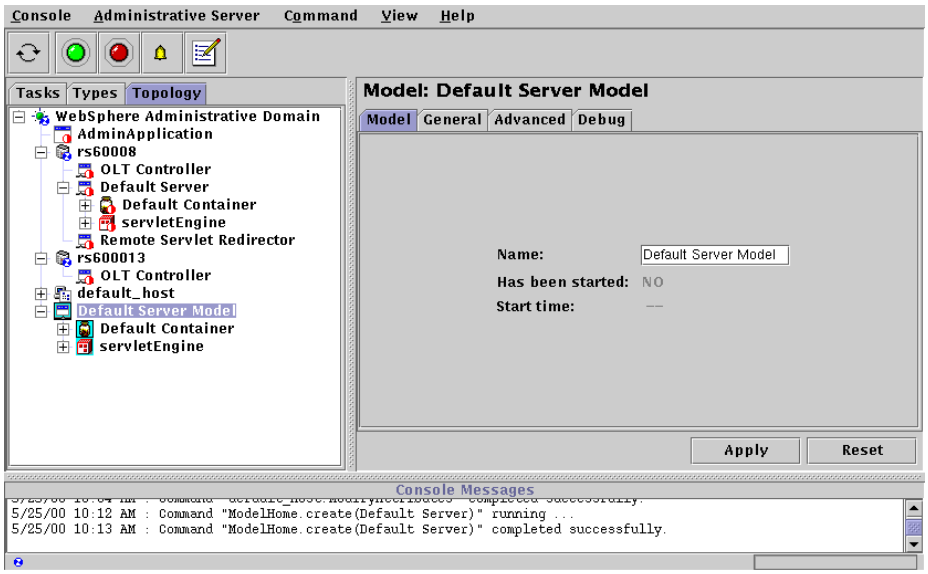

Figure 238. WebSphere topology showing model and clones

10.7 Changing the WLM configuration

Once the Web application model has been created, you can change the workload management selection policy to suit your requirements. Select the model in the Topology pane of the Administrative Console and then click the **Advanced** tab. The default is Round Robin Prefer Local which is the preferred policy for this solution as all clones exist on a single machine. If you wish to change the policy, select one from the drop down list and then click **Apply** as shown in Figure 239 on page 286.

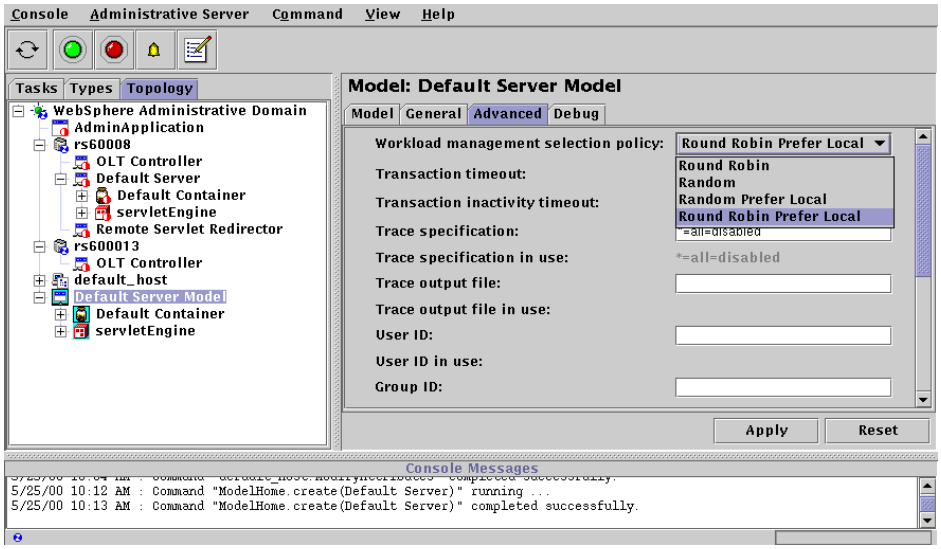

Figure 239. Changing the workload management selection policy

10.8 Creating the clones

Within this scenario, we are creating 2 clones of the model we generated above on each node. For brevity, we have shown this process for the first clone only on each node.

Note

If you clicked the option to make the parent application server a clone of the model when you generated the model, you will only need to create a single clone for model, as the first clone will be your original server.

10.8.1 Creating the clones for Machine A

From the Topology pane of the WebSphere Administrative Console, select the model (in our case, Default Server Model) you wish to clone, right click then select **Create->Clone** as shown in Figure 240 on page 287.

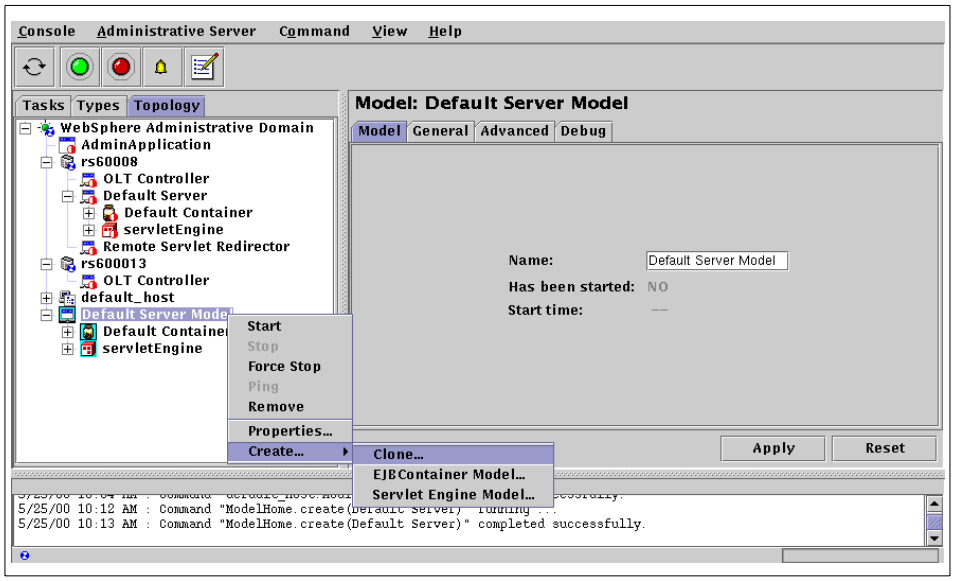

Figure 240. Creating a clone of the model

In the Clone Parent dialog box that is displayed, enter your chosen clone name (our case, Default Server Clone1). Select the node on which you require the clone to be installed, by expanding the Nodes tree and right clicking the relevant node (our case, rs60008) as shown in Figure 241.

Click the **Create** button to generate the clone.

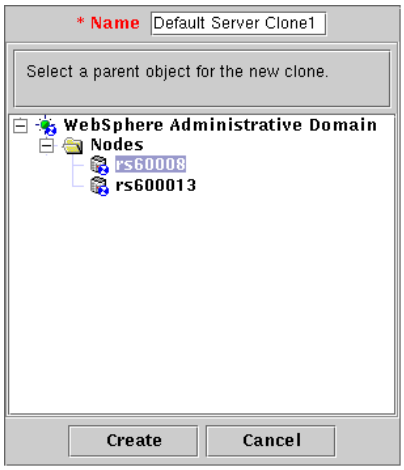

Figure 241. Clone parent

When the clone has been created, you will see the following message.

Figure 242. Clone created message

You should now repeat this process for the remaining clones for Machine A.

10.8.2 Creating the clones for Machine B

From the Topology pane of the WebSphere Administrative Console, select the model (in our case, Default Server Model) you wish to clone, right click and select **Create->Clone** as shown in Figure 243.

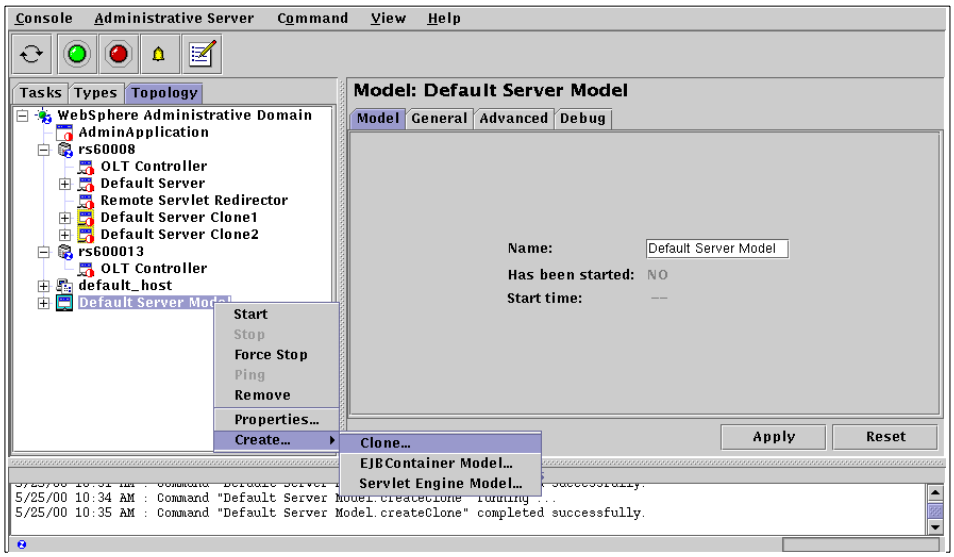

Figure 243. Creating a clone of the model

In the Clone Parent dialog box that is displayed, enter your chosen clone name (in our case, Default Server Clone3). Select the node on which you require the clone to be installed, by expanding the Nodes tree and clicking on the relevant node as shown in Figure 244 on page 289.

Click the **Create** button to generate the clone.

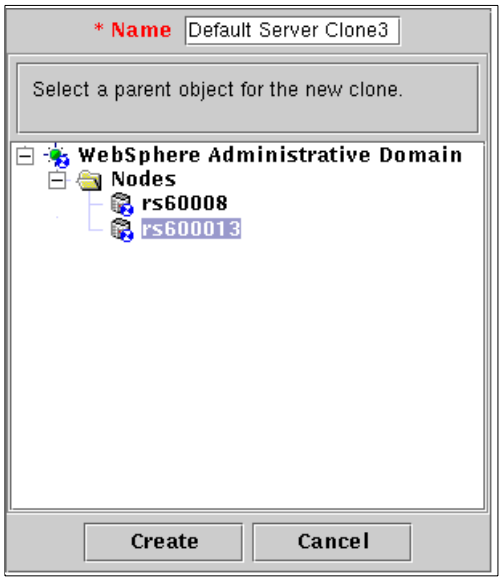

Figure 244. Clone parent

When the clone has been created, you will see the following message.

Figure 245. Clone created message

You should now repeat this process for the remaining clones for Machine B.

After completing your configuration, you should see your clones in the Topology tree surrounded by a yellow square, used to identify clones as shown in Figure 246 on page 290.

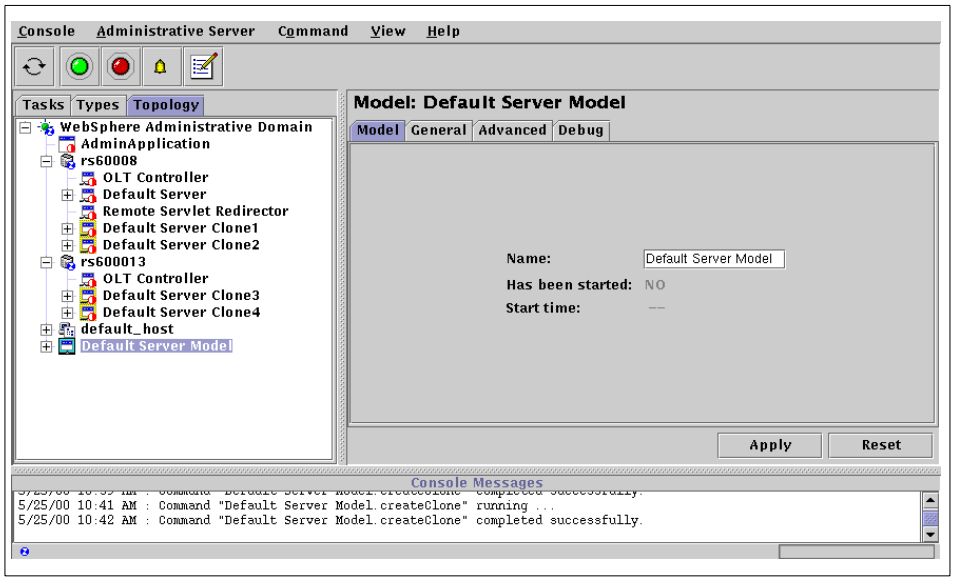

Figure 246. WebSphere topology showing clones

Clones of the Default Server on separated machines

For V3.5, the default configuration automatically creates and installs the JDBC Driver (Admin DB Driver).

When creating a clone of the Default Server to a remote node, it requires the Admin DB Driver to be installed on the remote node prior to creating the clone.

To install the Admin DB Driver, right click the Admin DB Driver and select **Install**. The Install Driver window will appear. Then select the remote node and specify the JAR file which contains the JDBC Driver. Then click **Install**. After you install the JDBC Driver for the Admin DB Driver on the remote node, you can create clones of the Default Server.

10.9 Starting the servers

Now you are ready to start the model, Web servers, and Network Dispatcher.

10.9.1 Starting the model

Start the model by right clicking the model name (in our case, Default Server Model), in the Topology Tree and selecting **Start** as shown in Figure 247.

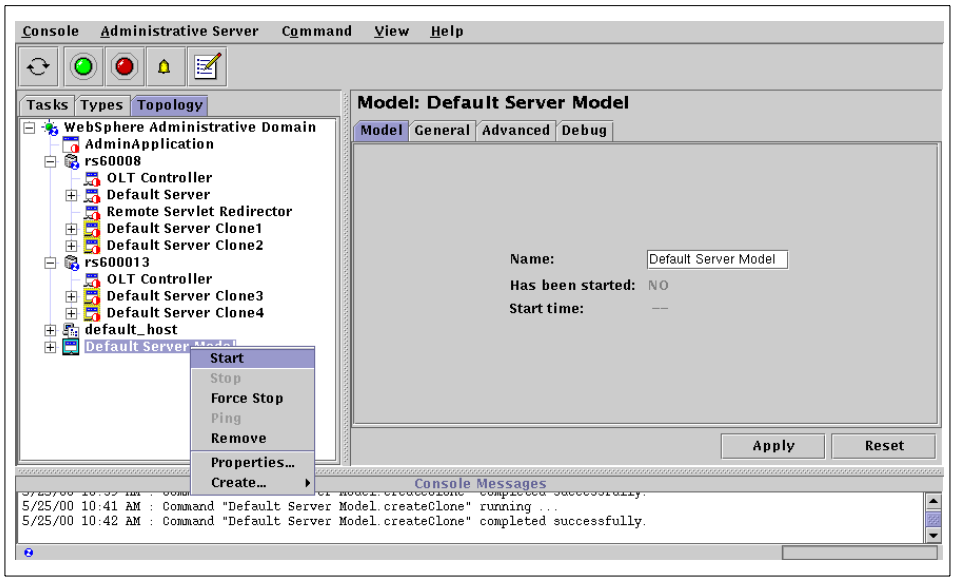

Figure 247. Starting the model

Once the model has started, the clones for the model you started, should now show as started in the Topology pane as shown in Figure 248 on page 292.

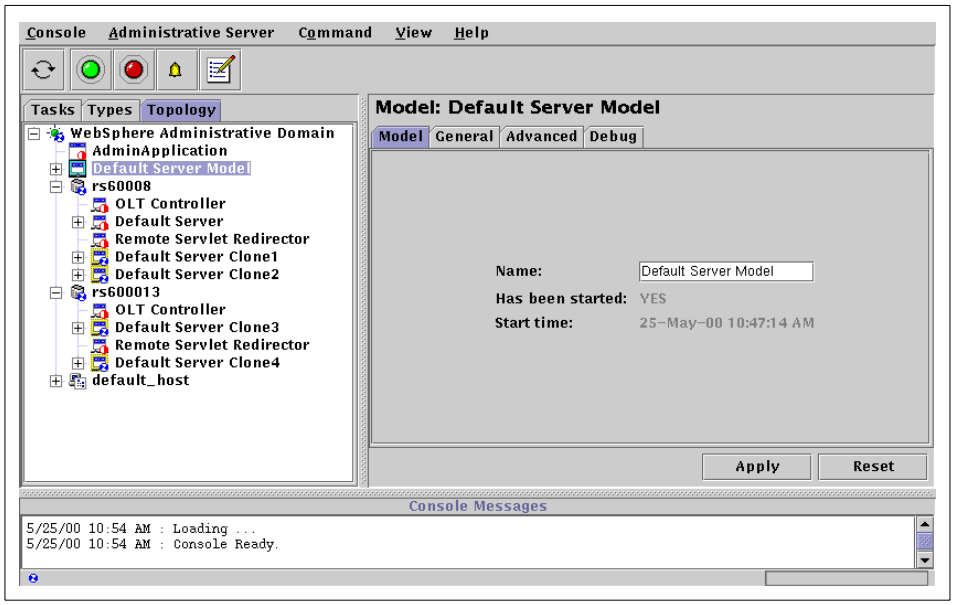

Figure 248. Topology pane, showing started clones

Note

At the time of writing, an issue with the Administrative Console causes the clone icons to not always display as started. We have found that refreshing the node subtree will only occasionally correct the icons. Stop and restart the Administrative Console to display the correct information.

10.9.2 Starting the HTTP servers

Now, you need to start the HTTP servers on both Machines A and B. If they are already started, stop and restart them.

10.9.3 Starting the Network Dispatcher

After you start the WebSphere nodes and the HTTP servers, you need to start the Network Dispatcher.

10.10 Test the servers

Now that the Network Dispatcher, the HTTP servers, and WebSphere servers are running, we need to test the servers to make sure that everything is configured properly.

Start up an instance of your preferred browser, in our examples we use Netscape, enter the following URI (`www.itso.ral.ibm.com` is the cluster address of ND) and press Enter:

```
http://www.itso.ral.ibm.com/webapp/examples/showCfg
```

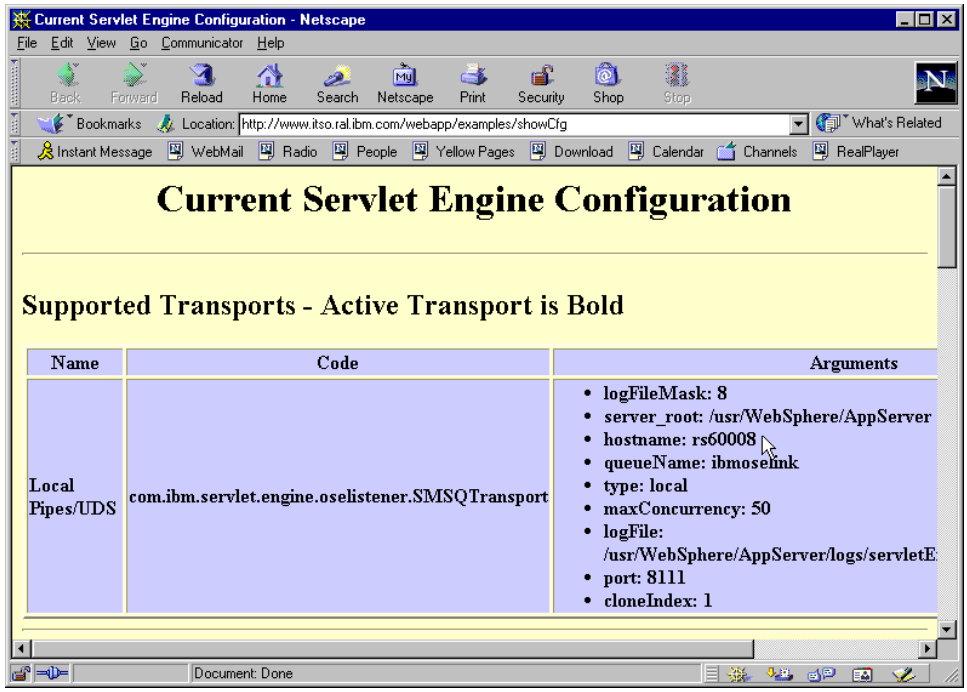

Figure 249. Result in the browser: rs60008, Clone1

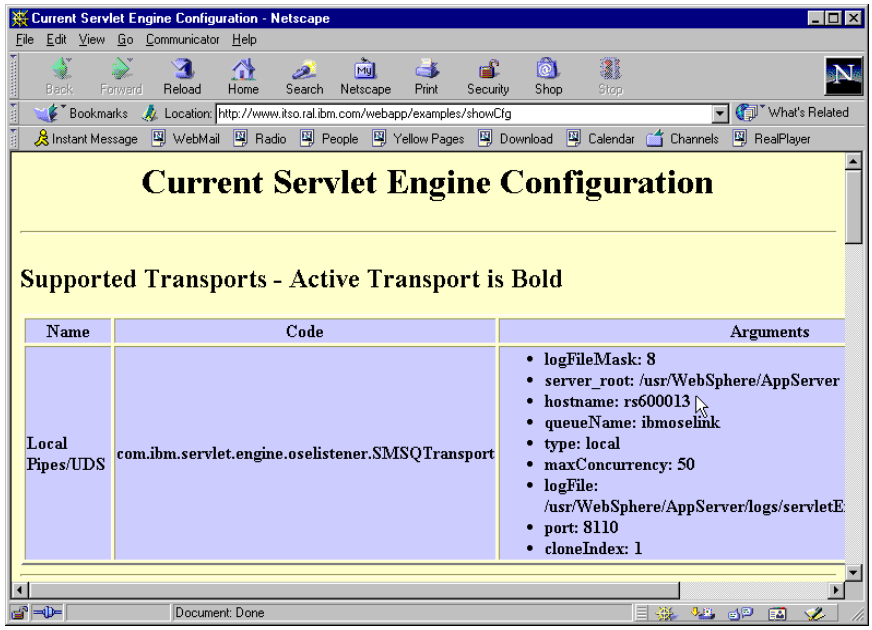

Figure 250. rs600013, Clone3

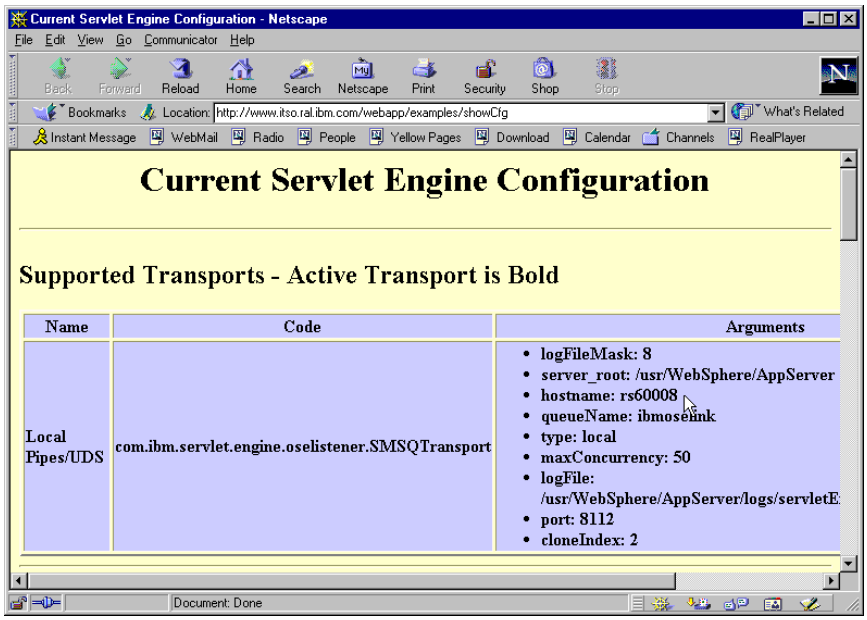

Figure 251. rs60008, Clone2

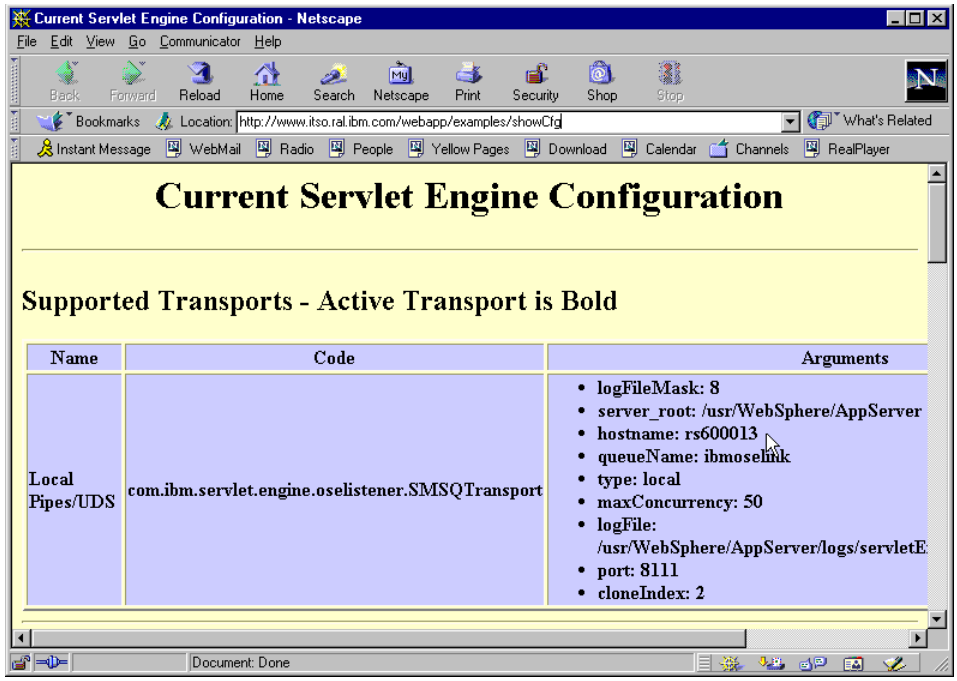

Figure 252. rs600013, Clone4

Notice that the result in Figure 249 on page 293 tells you that the hostname is the name of the node running the application server (rs60008) and not the cluster name (www). And also, the port number of the servlet engine is 8111 and cloneIndex is 1. The result in Figure 251 on page 294 tells you that the hostname is also rs60008 but the port number of the servlet engine is 8112 and cloneIndex is 2.

As the initial load balancing is performed by the *IBM Network Dispatcher*, you cannot guarantee which HTTP server instance the request will be forwarded to. It is possible to predict how requests are processed when they reach a particular WebSphere instance based on the workload selection policy. For our scenario, the policy was round robin prefer local and our testing confirmed this.

10.11 Related topologies

Using the *IBM Network Dispatcher* allows us to perform intelligent load blancing and horizontal scaling for HTTP servers in all of our scenarios. We included some below.

10.11.1 Variation 1: ND with multiple WebSphere domains

This section describes how we can utilize distinct WebSphere domains and the *IBM Network Dispatcher* to create highly available environments. As each domain uses a separate repository on a different machine, we can create a cluster of WebSphere domains to support our environment. Although this type of scenario provides a greater level of availability, it does so with an increased level of complexity for maintaining and synchronizing the environments. In this topology, each WebSphere node has to have its own Administrative Repository.

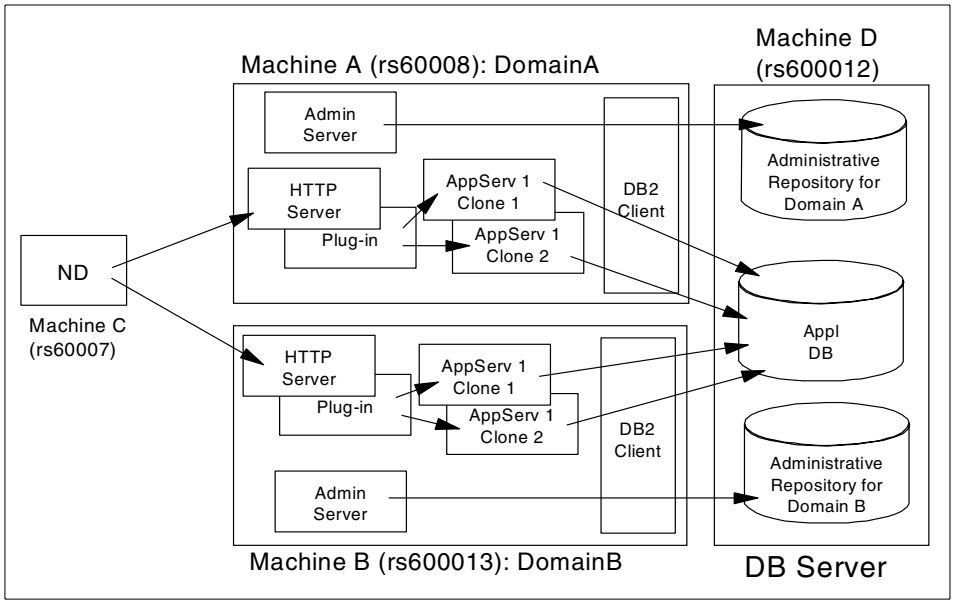

Figure 253. Multiple domains

Therefore, on each Administrative Console, you can see only your node as shown in Figure 254 on page 297 and Figure 255 on page 297.

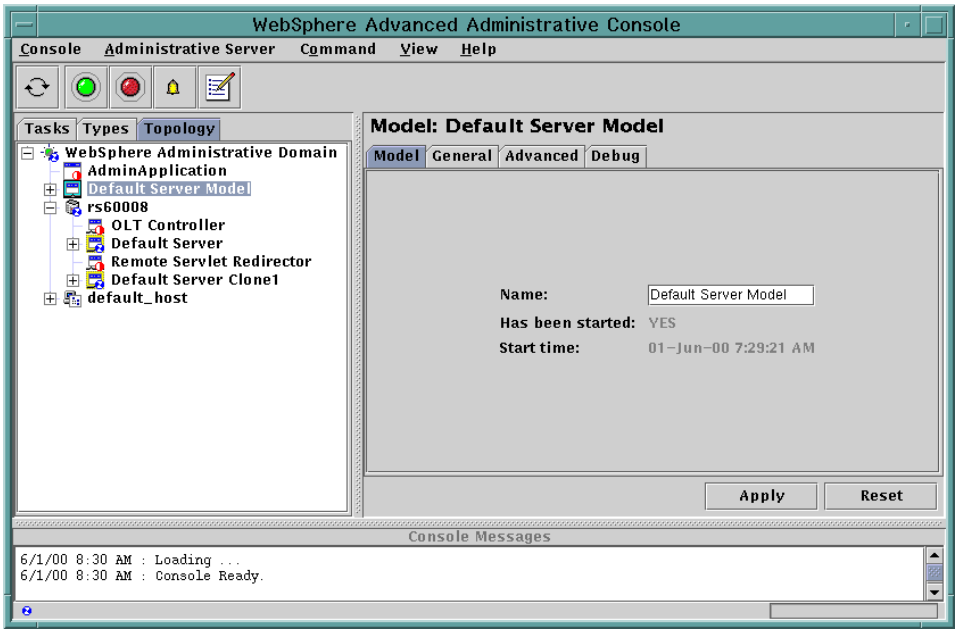

Figure 254. Administrative Console on Machine A (Domain A)

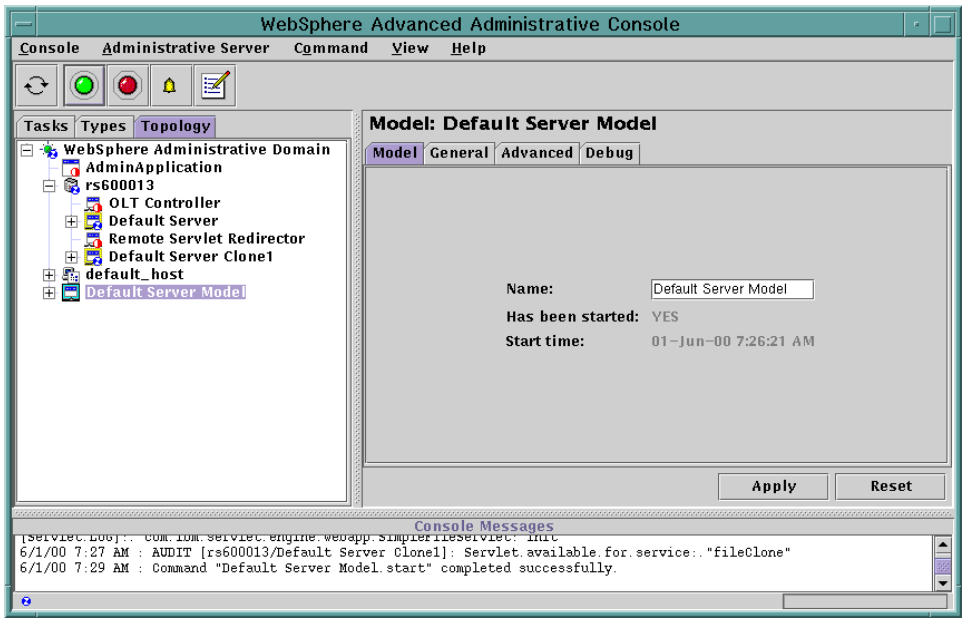

Figure 255. Administrative Console on Machine B (Domain B)

10.11.2 Variation 2: ND with OSE Remote

Two machines acting as the HTTP servers (Machines A and D) will use OSE Remote to route requests to nodes on Machines B and E respectively. Each node will be running a clone or clones of one or more application servers. The nodes will share a repository stored on Machine C. For OSE Remote configuration, see Chapter 5, “OSE Remote” on page 95.

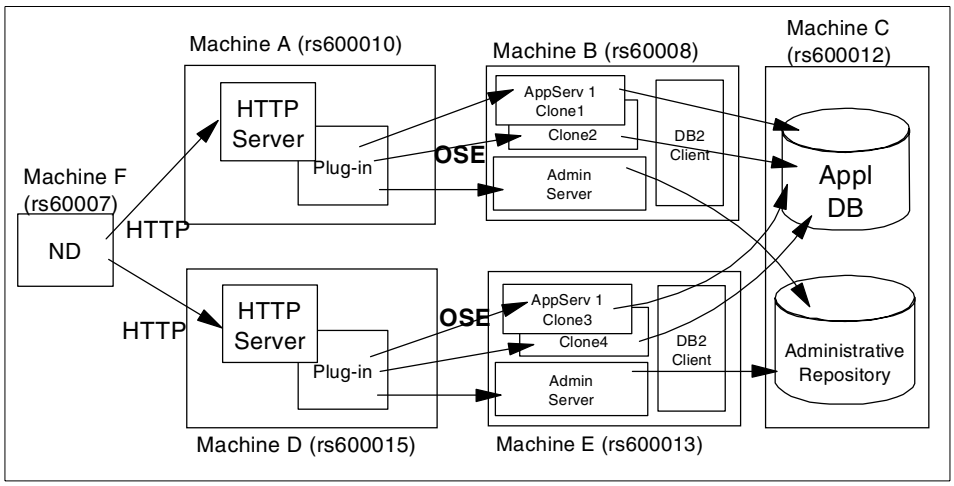

Figure 256. Network Dispatcher with OSE Remote

With this configuration, Machine A can forward requests to Machine B (rs60008) only.

```
#IBM WebSphere Plugin Communication Queues
#Thu May 25 11:00:09 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone2.port=8112
ose.srvgrp.ibmoselink.clone1.port=8111
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone2.host=rs60008
ose.srvgrp.ibmoselink.clone1.host=rs60008
```

Figure 257. <was_dir>/temp/queues.properties on Machine A

```
#IBM WebSphere Plugin Communication Queues
#Thu May 25 10:56:28 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone2.port=8111
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone2.host=rs600013
ose.srvgrp.ibmoselink.clone1.host=rs600013
```

Figure 258. <was_dir>/temp/queues.properties on Machine D

In this environment, if Machine B has been stopped, browsers will get the following error message even if running the Network Dispatcher http advisor.

Figure 259. Error message on the browser

Using the ND WebSphere application server custom advisor

To avoid this problem, IBM Network Dispatcher 3.0 provides a custom advisor for WebSphere application server as a sample. In our environment, we ran it instead of the default http advisor. When the WAS custom advisor is running, we can continue to access clone 3 and clone 4 on Machine E without the above error message even when Machine B is stopped.

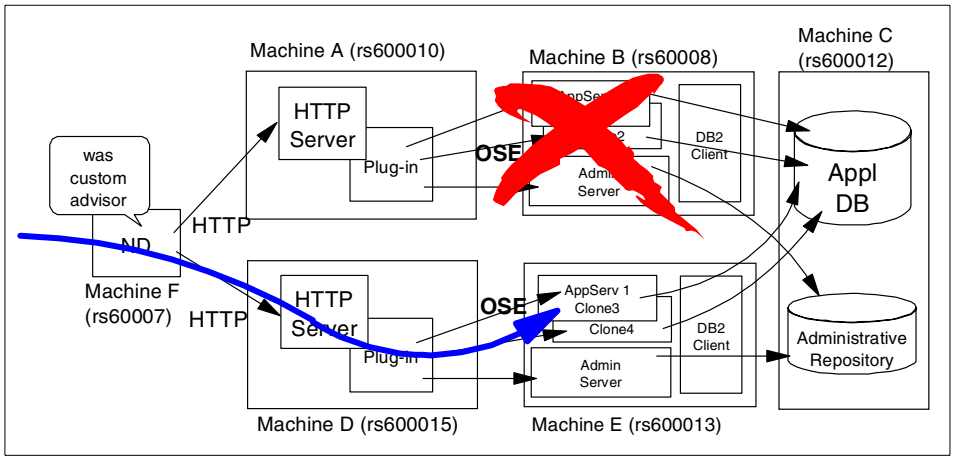

Figure 260. Network Dispatcher 3.0 provides WebSphere custom advisor

To run the WAS custom advisor, you need to modify the host aliases information. All HTTP server machines have to have their own hostname information in their <was_dir>/temp/vhosts.properties file. Also, the WebSphere machines have to have the host alias information of all HTTP servers as well.

```
#IBM WebSphere Plugin Virtual Host Mappings
#Tue May 30 09:05:39 EDT 2000
localhost=default_host
127.0.0.1=default_host
www.itso.ral.ibm.com=default_host
rs600010.itso.ral.ibm.com=default_host
```

Figure 261. <was_dir>/temp/vhosts.properties on Machine A for WAS custom advisor support

```
#IBM WebSphere Plugin Virtual Host Mappings
#Tue May 30 09:06:30 EDT 2000
localhost=default_host
127.0.0.1=default_host
www.itso.ral.ibm.com=default_host
rs600015.itso.ral.ibm.com=default_host
```

Figure 262. <was_dir>/temp/vhosts.properties on Machine D for WAS custom advisor support

```
#IBM WebSphere Plugin Virtual Host Mappings
#Tue May 30 11:17:14 EDT 2000
localhost=default_host
127.0.0.1=default_host
www.itso.ral.ibm.com=default_host
rs600015.itso.ral.ibm.com=default_host
rs600010.itso.ral.ibm.com=default_host
```

Figure 263. <was_dir>/temp/vhosts.properties on Machine B and E for WAS custom advisor

Note

You cannot run the HTTP adviser and the WAS custom advisor at the same time if you specify the same port number for both advisors.

10.11.3 Variation 3: OSE Remote with high availability

This is a variation of the model discussed in 10.11.2, “Variation 2: ND with OSE Remote” on page 298. Two machines acting as the HTTP server (Machines A and D) will use OSE Remote to route requests to nodes on Machines B and E. Each node will be running a clone or clones of one of more application servers. The nodes will share a repository stored on Machine C.

The topology has the advantage over the previous topology in that we have failover should either WebSphere Administrative Server on Machine B or Machine E fail or if they are taken down for service without the WAS custom advisor.

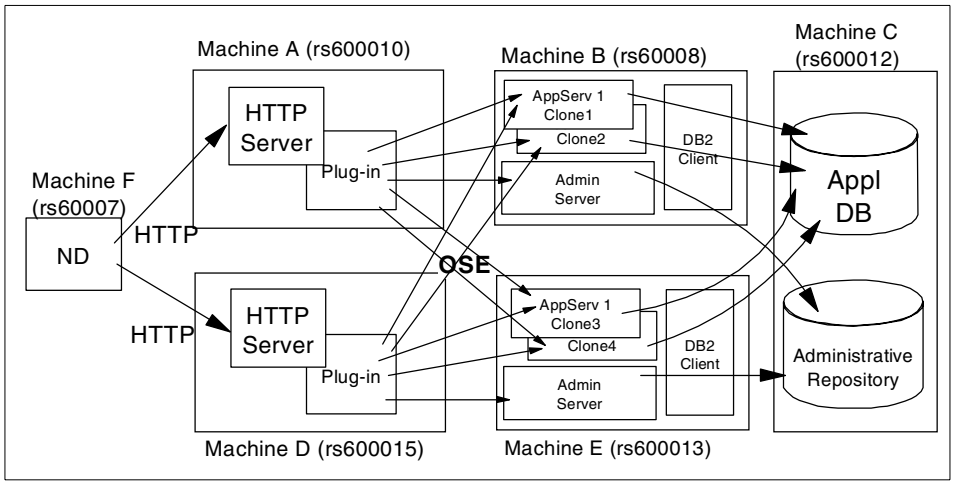

Figure 264. Network Dispatcher with OSE Remote high availability

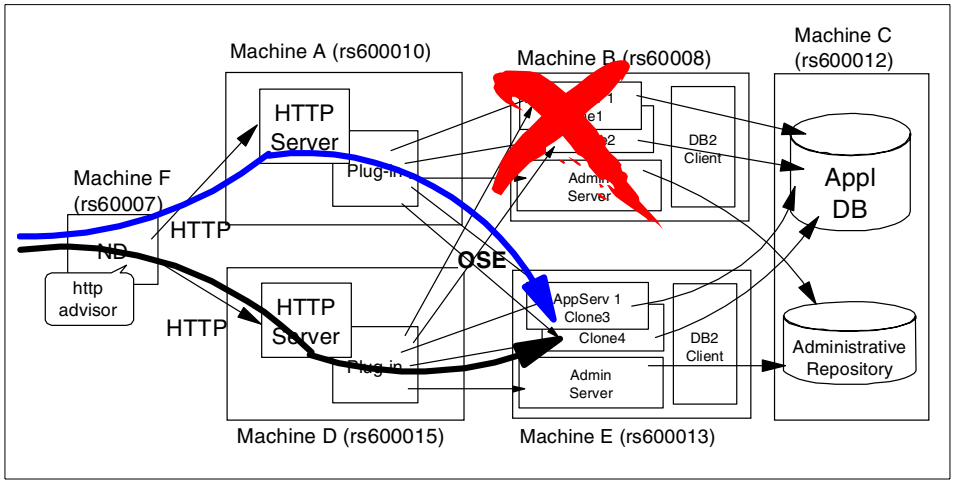

Figure 265. Network Dispatcher and OSE Remote high availability with http advisor

To access Machine E via both Machines A and D when Machine B fails, you need to run the http advisor instead of the WAS custom advisor as depicted in Figure 265. You cannot run both the http advisor and the WAS custom advisor at the same time because of port number contention.

With the WAS custom advisor, when Machine B fails, you can only access Machine E via Machine D as shown in Figure 266 on page 303.

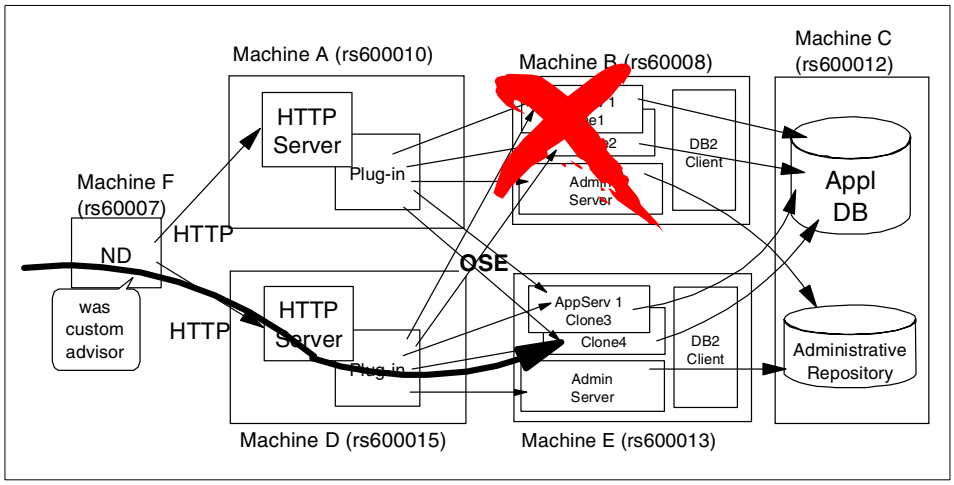

Figure 266. Network Dispatcher and OSE Remote high availability with WAS advisor

You can continue accessing both Machines B and E when either the HTTP server on Machine A or D fails or is taken down for service with either the http advisor or the WAS custom advisor as shown in Figure 267.

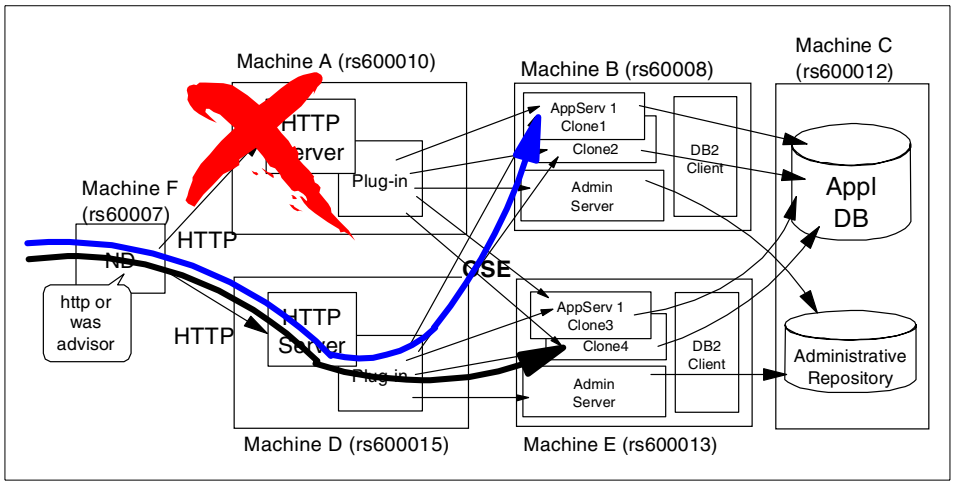

Figure 267. Network Dispatcher and OSE Remote high availability with advisor

Both Machines A and D have to know all clones on both Machines B and E as follows.

```
#IBM WebSphere Plugin Communication Queues
#Thu May 25 11:00:09 EDT 2000
ose.srvgrp.ibmoselink.clonescount=4
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone4.port=8111
ose.srvgrp.ibmoselink.clone3.port=8110
ose.srvgrp.ibmoselink.clone2.port=8112
ose.srvgrp.ibmoselink.clone1.port=8111
ose.srvgrp.ibmoselink.clone4.type=remote
ose.srvgrp.ibmoselink.clone3.type=remote
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone4.host=rs600013
ose.srvgrp.ibmoselink.clone3.host=rs600013
ose.srvgrp.ibmoselink.clone2.host=rs60008
ose.srvgrp.ibmoselink.clone1.host=rs60008
```

Figure 268. <was_dir>/temp/queues.properties on Machine A

```
#IBM WebSphere Plugin Communication Queues
#Thu May 25 10:56:28 EDT 2000
ose.srvgrp.ibmoselink.clonescount=4
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone4.port=8112
ose.srvgrp.ibmoselink.clone3.port=8111
ose.srvgrp.ibmoselink.clone2.port=8111
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone4.type=remote
ose.srvgrp.ibmoselink.clone3.type=remote
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone4.host=rs60008
ose.srvgrp.ibmoselink.clone3.host=rs60008
ose.srvgrp.ibmoselink.clone2.host=rs600013
ose.srvgrp.ibmoselink.clone1.host=rs600013
```

Figure 269. <was_dir>/temp/queues.properties on Machine D

10.11.4 Variation 4: OSE Remote and local

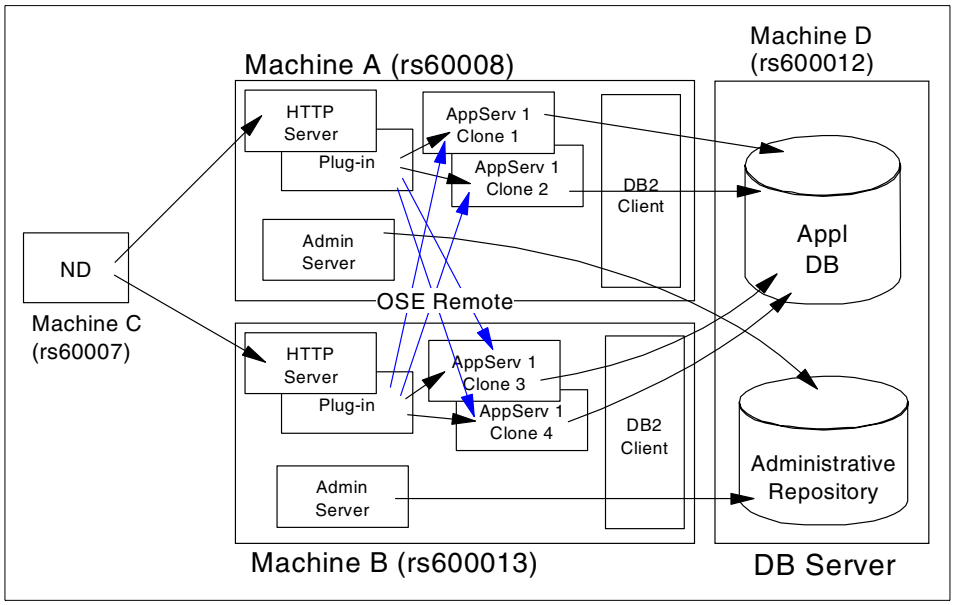

Figure 270. Network Dispatcher with OSE Remote and local

This configuration is not as simple as the other topologies. For this topology, after you configure the topology as shown in Figure 231 on page 278, update the transport type of servlet engines from Local pipe to INET Socket to support OSE Remote and modify queues.properties for clones on the remote machine. There are several steps you then have to do manually.

WebSphere periodically updates the <was_dir>/temp/queues.properties file, which will overwrite the manual (or with script) changes you made to enable the Remote OSE link.

```
#IBM WebSphere Plugin Communication Queues
#Tue May 30 11:17:14 EDT 2000
ose.srvgrp.ibmoselink.clonescount=4
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone4.port=8111
ose.srvgrp.ibmoselink.clone3.port=8110
ose.srvgrp.ibmoselink.clone2.port=8112
ose.srvgrp.ibmoselink.clone1.port=8111
ose.srvgrp.ibmoselink.clone4.type=remote
ose.srvgrp.ibmoselink.clone3.type=remote
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone4.host=rs600013
ose.srvgrp.ibmoselink.clone3.host=rs600013
ose.srvgrp.ibmoselink.clone2.host=rs60008
ose.srvgrp.ibmoselink.clone1.host=rs60008
```

Figure 271. queues.properties for OSE Remote with clones

The HTTP server periodically reads this file as shown in Figure 272 on page 307. If the servlet engine and the HTTP server are on the same machine then both are using the same file so the HTTP server will pick up the updates which will break workload management and all requests will go only to the local machine's clones. If you make a copy of the working queues.properties file then you can copy it back whenever required. This is cumbersome and impractical.

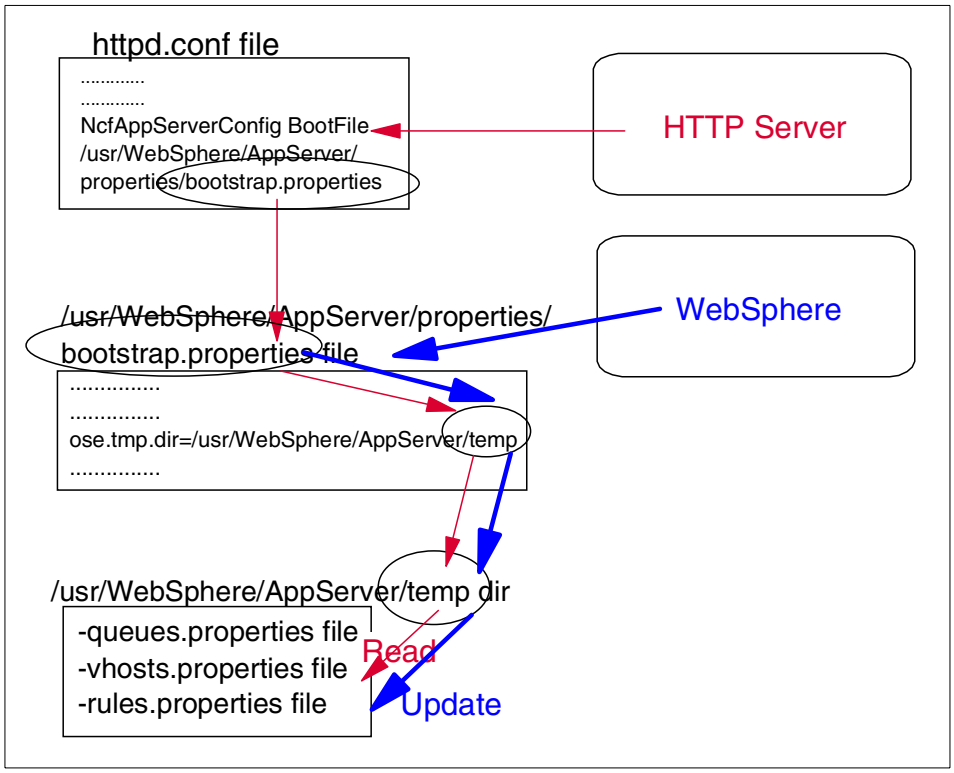

Figure 272. Both HTTP Server and WebSphere need access to the <was_dir>/temp directory

The solution is to separate the files that WebSphere updates and those that the HTTP server reads.

The following steps will avoid the problem with the queues.properties file being overwritten:

1. Create a new directory under <was_dir>, for example, we created:

   `/usr/WebSphere/AppServer/temp4was`

```
# pwd
/usr/WebSphere/AppServer
#
# mkdir temp4was
#
# ls -l |grep temp
drwxr-xr-x   3 root     system       512 May 30 10:04 temp
drwxr-xr-x   2 root     system       512 May 31 09:05 temp4was
#
```

2. Copy the properties files from the <was_dir>/temp directory (the queues.properties file in its working form) to the new directory which you created in step 1 above.

```
# pwd
/usr/WebSphere/AppServer
#
# cp -p temp/queues.properties temp4was/
# cp -p temp/vhosts.properties temp4was/
# cp -p temp/rules.properties temp4was/
#
# cd temp4was
# ls -l
total 24
-rw-rw-r--   1 root     system       657 May 31 09:00 queues.properties
-rw-rw-r--   1 root     system      1078 May 30 11:17 rules.properties
-rw-rw-r--   1 root     system       232 May 30 11:17 vhosts.properties
#
```

3. Make a copy of the <was_dir>/properties/bootstrap.properties file (for example, HTTPbootstrap.properties)

```
# pwd
/usr/WebSphere/AppServer/properties
#
# cp -p bootstrap.properties HTTPbootstrap.properties
#
# ls -l |grep HTTP
-rwxr-xr-x   1 root     system      4567 May 30 10:12 HTTPbootstrap.properties
#
```

4. Change the `ose.tmp.dir` entry in the original bootstrap.properties file to point to the new directory. For example:

   `/usr/WebSphere/AppServer/temp4was`

```
##
# Temp ose directory.
# To optimize performance insure that ose.tmp.dir points to a local
drive
#
# ose.tmp.dir=/usr/WebSphere/AppServer/temp
ose.tmp.dir=/usr/WebSphere/AppServer/temp4was
```

Figure 273. Update the ose.tmp.dire entry in the <was_dir>/properties/bootstrap.properties

5. Stop and restart WebSphere Administrative Servers.

6. Start Administrative Console.
7. Start model (clones).

> **Note**
>
> Do not restart the model or clones before you restart the Administrative Servers (step 5). If you do, the queues.properties file will get overwritten in the <was_dir>/temp directory.

The queues.properties file now only gets overwritten in the new directory (in our case, <was_dir>/temp4was), not the original <was_dir>/temp directory.

```
#IBM WebSphere Plugin Communication Queues
#Wed May 31 11:45:12 EDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone2.port=8112
ose.srvgrp.ibmoselink.clone1.port=8111
ose.srvgrp.ibmoselink.clone2.type=remote
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 274. <was_dir>/temp4was/queues.properties was updated after model (clones) restarted

8. Change /usr/HTTPServer/conf/httpd.conf so that the BootFile points to the copied version, for example:

   ```
   <was_dir>/properties/HTTPbootstrap.properties
   ```

```
Alias /IBMWebAS/samples/  /usr/WebSphere/AppServer/samples/
Alias /IBMWebAS/  /usr/WebSphere/AppServer/web/
# NcfAppServerConfig BootFile
/usr/WebSphere/AppServer/properties/bootstrap.properties
NcfAppServerConfig BootFile
/usr/WebSphere/AppServer/properties/HTTPbootstrap.properties
```

Figure 275. Update the BootFile entry in the /usr/HTTPServer/conf/httpd.conf

9. Stop and restart the HTTP server.

Now WebSphere will not destroy the changes to the <was_dir>/temp/queues.properties file which the HTTP server is using

because it updates the new directory <was_dir>/temp4was, which is no longer used.

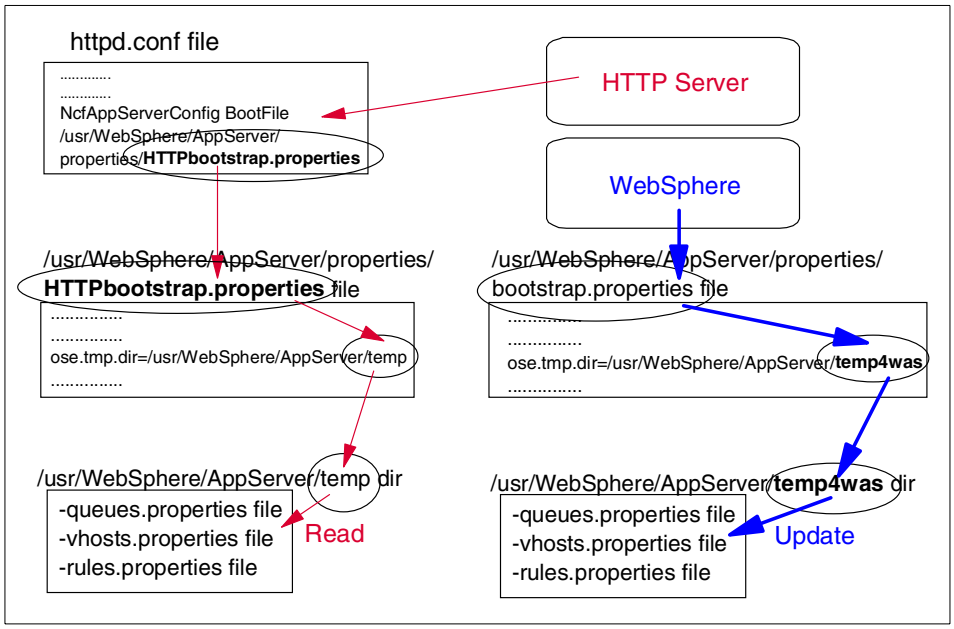

Figure 276. Configuration to avoid the problem with the queues.properties file being overwritten

Part 3. Setting up your topology for OS/400

Chapter 11. AS/400 considerations

The AS/400 offers many different advantages over other platforms for running WebSphere. Things from ease of operations to administration. The tools that you need to create your Web applications and debug them are all available to you. The total cost of ownership is proven lower for the AS/400. The AS/400 will also grow with your business, scaling to meet your demands. As a multiuser and multi-application platform, you won't need separate machines to run each piece of your Web application.

In this chapter we will discuss the AS/400 specific considerations for WebSphere V3.02. The items we will be discussing are:

AS/400 advantages:

1. Integrated DB2 database
2. Integrated JVM
3. Access to legacy applications
4. Coexistence Standard Edition and Advanced Edition
5. Multiple instance support

Configuration customization:

6. Creating a server without the Default Application Server
7. Knowing when the Administrative Server is ready
8. Administrative agent versus Administrative Server
9. Configuring a remote Administrative Repository
10. Creating a WebSphere cluster
11. Configuring an Administrative agent
12. Persistence session
13. Configure an HTTP server
14. Using a Network Dispatcher
15. Implementing a heterogeneous environment
16. Using JDBC drivers

11.1 Integrated DB2 database

As the AS/400 has an integrated database, this gives rise to slightly different topology designs to those described in the Windows/UNIX topology sections. Using a local database for an Administrative Repository provides faster accesses and more security than a remote database. As standard the OS/400 operating system and licensed products provide extensive tools that help manage, track, and maintain the database.

We can also utilize the UNIX/Windows topologies described in the earlier chapters. For information on how to set up the WebSphere Administrative Server Repository on a separate machine see 11.9, “Using a separate AS/400 for the admin instance repository” on page 325.

11.2 Integrated JVM

Just as a typical Java Virtual Machine insulates the Java byte codes from the nuances of the operating system and hardware platform, the AS/400 has always insulated users and developers from the underlying hardware characteristics through the use of a layered machine architecture. This layered architecture, named Technology Independent Machine Interface (TIMI), raises the level of the machine interface, creating a high level machine instruction set that is independent of the underlying hardware implementation.

The Java Virtual Machine takes the Java bytecodes and converts these into the specific operating system functions and calls. Similarly, when an AS/400 program is executed, the instructions undergo a further process of translation, carried out by the System Licensed Internal Code (SLIC), before they are understood by the hardware.

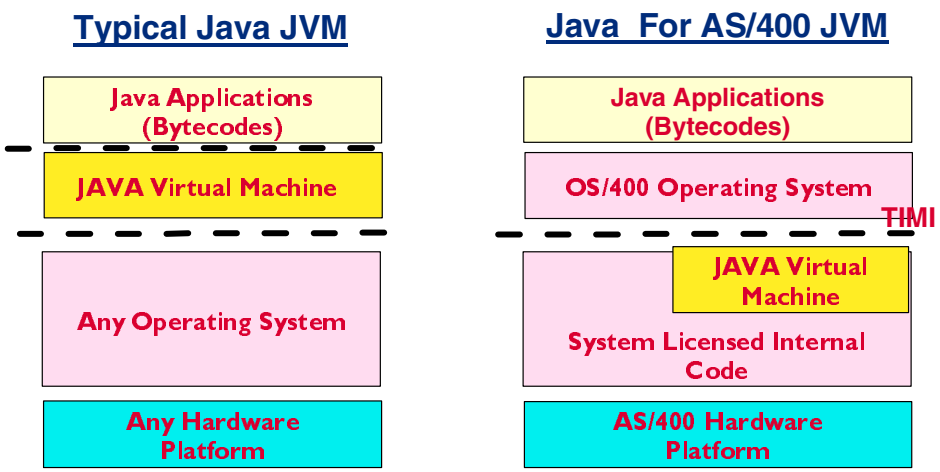

Figure 277. Differences between a typical JVM implementation and the integrated AS/400 JVM

As the AS/400 implements the concept of a technology independent machine interface it was possible to build the Java Virtual Machine into the OS/400 SLIC to provide a highly tuned, efficient, stable and scalable implementation.

11.3 Access to legacy programs

OS/400 allows access to legacy programs and applications through the use of the Java Tool Box for the 400. This is a collection of Java classes designed to help you interface with other programs.

11.4 Co-existence of WebSphere Standard and Advanced Editions

11.4.1 V2 Standard and V3 Advanced

The AS/400 implementation of WebSphere Application Server Advanced V3.0x is designed to co-exist with WebSphere V2 Standard Edition. This provides AS/400 developers with the ability to parallel run their application under V2 Standard and V3 Advanced without the requirement for an additional AS/400.

The co-existence of the WebSphere Application Server software does not mean that a single HTTP server can utilize both versions of WebSphere concurrently, so you must either create a new HTTP server instance or remove and V2 Standard Edition configuration information from the HTTP

server configuration information before configuring the HTTP server for V3.0 Advanced support.

11.4.2 V2 Standard and V3 Standard

The AS/400 implementation of WebSphere Application Server Standard V3.0x is designed to co-exist with WebSphere V2 Standard Edition. This provides AS/400 developers with the ability to parallel run their application under V2 Standard and V3 Standard without the requirement for an additional AS/400.

The co-existence of the WebSphere application server software does not mean that a single HTTP server can utilize both versions of WebSphere concurrently, so you must either create a new HTTP server instance or remove and V2 Standard Edition configuration information from the HTTP server configuration information before configuring the HTTP server for V3.0 Standard support.

11.4.3 V3 Standard and V3 Advanced

Unfortunately, V3 Standard and V3 Advanced Editions can not co-exist on the same AS/400.

11.5 Multi-instance support for WebSphere V3 Advanced

The AS/400 version of the WebSphere Application Server Advanced Edition V3.0x officially supports multiple-instances of the WebSphere Administrative Server on a single AS/400.

Each WebSphere Administrative Server shares a single copy of the WebSphere Advanced Edition code and requires a unique IFS directory for the instance setup, such as the property files, servlets, etc. and a unique AS/400 database collection for the instance repository.

The number of support instances is limited by the size and capacity of the AS/400 utilized and the type of work being undertaken within the WebSphere Application Server Advanced instances.

Note

Although not shown or described in the succeeding sections, the majority of WebSphere scenarios were configured as multiple instances to allow us to test these scenarios in parallel.

11.5.1 Creating another WebSphere Administrative Server instance

Configuring another WebSphere Administrative Server instance is a simple process involving the following steps:

1. Creating a new, discrete instance of the WebSphere Configuration files
2. Modifying the instance properties files to ensure there are no conflicts
3. Starting the new instance
4. Conecting the Administrative Console
5. Other considerations

Note

Refer to your product documentation for a full description of this configuration process for multiple instance support.

The latest online version of this information can be found at:

`www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html`

11.5.1.1 Creating a new instance

Since the AS/400 WebSphere Application Server environment offically supports multiple instances, a script has been provided to create the required directories and copy the necessary files. The script named crtnewinst exists in the /QIBM/ProdData/WebAsAdv/bin IFS directory.

To run this script, you must enter the Qshell interpreter and specify the fully qualified script name followed by the fully qualified IFS directory name you require for this new instance, separated by a single space. It is recommended, but not enforced, that this directory exists under the /QIBM/UserData/WebAsAdv directory as shown in Figure 278 on page 318.

```
QSH Command Entry

  $



===> /QIBM/ProdData/WebASAdv/bin/crtnewinst
                           /QIBM/UserData/WebASAdv/new


F3=Exit   F6=Print F9=Retrieve F12=Disconnect
F13=Clear F17=Top  F18=Bottom  F21=CL command entry
```

Figure 278. Create a new WebSphere Administrative Server instance

Note

The new directory information can be placed after the script name with a single space in between. It is shown in this format for clarity.

The fully qualified IFS directory name you specified is known as the WebSphere instance root or instance root. It can be found specified as <was_instance_root> in the WebSphere documentation. In the above example our <was_instance_root> is /QIBM/UserData/WEbAsAdv/new.

Typically, the default WebSphere Administrative Server instance root is known as <was_default_instance_root> and has the fully qualified IFS directory path of /QIBM/UserData/WebASAdv/default.

11.5.2 Modifying the instance properties files

The WebSphere property files created by running the script are generic. They will have the TCP/IP port and <was_default_instance_root> directory information settings of the initial default instance created when the WebSphere product was installed. Three property files: bootstrap.properties, admin.properties, and sas.server.props must be modified, before you start this new instance. The files exist in the <was_instance_root>/properties directory for the new instance. In our example this would be /QIBM/UserData/WebAsAdv/new/properties.

Note

These files have an ASCII code page and must have an ASCII code for the WebSphere Administrative Server instance to work.

Several of the changes require you to specify new TCP/IP port information. You must ensure that the TCP/IP port numbers you specified are unique amongst all your instances and are not used by any existing TCP/IP application.

11.5.2.1 bootstrap.properties file

The bootstrap.properties file contains the information pertinent to the HTTP plug-in. The following four properties must be changed.

Table 11. bootstrap.properties file changes

Property to edit	Description
server.root	The location of the WAS instance root directory. This should be your chosen <was_instance_root>.
ose.tmp.dir	The location of the OSE temporary files use by the HTTP plug-in. This is the temp subdirectory of the <was_instance_root>.
ose.logs.dir	The location to generate any OSE log files. This is the logs subdirectory of the <was_instance_root>.
ose.srvgrp.ibmappserve.clone1.port	This is used internally by the HTTP plug-in to communicate with the Administrative Server instance. This must be a unique unused TCP/IP port number.

We edited the bootstrap.properties file and replaced the default values for our example instance as shown.

Table 12. bootstrap.properties file changes

Property to edit	New value
server.root	/QIBM/UserData/WEbAsAdv/new
ose.tmp.dir	/QIBM/UserData/WEbAsAdv/new/temp

Property to edit	New value
ose.logs.dir	/QIBM/UserData/WEbAsAdv/new/logs
ose.srvgrp.ibmappserve.clone1.port	8899

11.5.2.2 admin.properties

The admin.properties file contains the information pertinent to the WebSphere Administrative Server instance. The following properties must be changed.

Table 13. admin.properties file changes

Property to edit	Description
mntr.admin.name	A name to identify the WebSphere Administrative Server job for this instance. The name should preferably be unique across all instances.
install.initial.config	This setting controls whether the default application server is created when the instance next starts. This parameter should have a value of true to install the default application server or false not to install the server.
admin.dbSchema	The is the name of the AS/400 database collection used as this instance repository. The database collection will be created or re-created when the instances starts. This should be a maximum of 10 characters long.
admin.bootstrapPort	The Administrative Console uses this port number when connecting to a WebSphere Administrative Server. This must be a unique unused TCP/IP port number.
admin.lsdPort	A unique unused TCP/IP port number.
admin.classpath	This is the classpath used by the WebSphere Administrative Server when it starts. All references to the <was_default_instance_root> must be substituted with the new <was_instance_root>.
admin.instance.root	<was_instance_root>

Property to edit	Description
java.properties	These are the properties specified when starting the JVM. All references to the <was_default_instance_root> must be substituted with the new <was_instance_root>.

Note

We recommend that you adopt a naming standard for your repository database collections. When testing all our scenarios, we prefixed the repository database collection name with EJS.

We edited the admin.properties file and replaced the default values as shown.

Table 14. admin.properties file changes

Property to edit	New value
mntr.admin.name	NEWADMIN
install.initial.config	true
admin.dbSchema	EJSNew
admin.bootstrapPort	7799
admin.lsdPort	9999
admin.classpath	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root>.
admin.instance.root	<was_instance_root>
java.properties	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root>.

11.5.2.3 sas.server.props

The sas.server.props properties file is primarily used for WebSphere Security. We edited the file and replaced the default values as shown.

Table 15. sas.server.props file changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root>.
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root>.

11.5.3 Starting the new instance and connecting an Admin Console

11.5.3.1 Starting the new instance

The new instance will not start automatically, when the QEJBSBS subsystem is started, unless you automate this by utilizing an autostart job to do so.

To manually start an additional WebSphere Admistrative Server instance, you need to enter the following command from a 5250 command line as shown in Figure 279 substituting your <was_instance_root> in place of the example.

```
Command Entry                          AS4B
                                                    Request level:   4
Previous commands and messages:

  (No previous commands or messages)




                                                                    Bottom
Type command, press Enter.
===> SBMJOB CMD(CALL PGM(QEJB/QEJBMNTR)
 PARM('-P' '/QIBM/UserData/WebAsAdv/new/properties/admin.properties'))
 JOB(MNTRNEW) JOBD(QEJB/QEJBJOBD) JOBQ(QEJB/QEJBJOBQ) USER(QEJB)

F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
F11=Display full      F12=Cancel     F13=Information Assistant   F24=More keys
```

Figure 279. Starting the WebSphere Administrative Server instance

Note

The line spacing has been altered for clarity.

11.5.3.2 Starting the Administrative Console

The admin.bootstrapPort property in the admin.properties file (see Table 14 on page 321) specifies the port information required for connecting the Administrative Console. If you attempt to start the Administrative Console using the defaults, it will not connect to the new instance. You must specify the hostname of your Administrative Server and the port number you specified in admin.properties when you start the Administrative Console as shown in Figure 280.

```
C:\>adminclient RALYAS4B 7722
Remote AdminServer RALYAS4B will be accessed on port 7722.
```

Figure 280. Starting the Administrative Console

11.5.4 Other considerations

11.5.4.1 Horizontal scaling considerations

If this instance is being created for use in a horizontal scaling topology, then you should use the same <was_instance_root> for all instances in this topology. This is recommended, because when you create models and clones the <was_instance_root> directory used to qualify the standard out and error log files is that of the model's parent.

If you use a different <was_instance_root> for instances in your horizontal scaling topology then the log files will not be created due to an invalid directory structure.

11.5.4.2 Other considerations

When you have multiple instances of the WebSphere Application Server Advanced Edition configured on a single AS/400 system, you should ensure that the port information specified for the servlet engine does not conflict with other servlet engines from additional WebSphere Administrative Server instances. This is especially important if you are utilizing models and clones within your environment, as each successive clone increments this port number by one, starting from the port number originally specified.

If you are running multiple instances of the WebSphere Application Server or you have multiple HTTP server instances accessing a single WebSphere Application Server instance, then you must ensure that you update the host alias information of your virtual hosts to reflect this.

11.5.5 Multiple instance template

We have provided a template in order to help you gather the information required when creating a new WebSphere Administrative Server instance. The Appendix E, “Multi-instance template for AS/400” on page 513 also provides a before and after view of the property files using our example configuration.

11.6 Configuring Admin Server instance to not install the default application server

Under certain circumstances you may not want to have the default application server created when you first start your WebSphere Administrative Server instance. To stop the Administrative Server instance from creating the default application server, you need to change a property in the admin.properties file in your <was_instance_root>/properties directory. The admin.properties configuration change required to disable the creation of the default application server is shown in Table 16.

Table 16. admin.properties file changes

Property to edit	New value
install.initial.config	false

If you do not install the default application server when you first start your Administrative Server instance, you can change this property value to true and stop and restart the instance. This will create the default application server during the restart process of your instance.

11.7 Knowing when the Administrative Server instance is ready

The amount of time taken to fully start a WebSphere Administrative Server instance varies greatly, depending upon factors such as the workload of the system and how you are accessing the repository. When a WebSphere Administrative Server instance is fully started you will see message number EJB0106 in the joblog of your WebSphere Administrative Server instance. The message text for message EJB0106 is:

Table 17. Message ID and text

Message ID	Message text
EJB0106	WebSphere administration server QEJBTEST ready.

Note

It is important to allow the WebSphere Administrative Server to completely start before attempting to connect an Administrative Console. Failure to do so, may cause irregular behavior within both the Administrative Server and Console.

11.8 Running Administrative agent versus full Administrative Servers

At the time of writing, it is recommended that you configure additional nodes in a domain to run as Administrative agents. This is due to locking issues that can occur when using the toolbox JDBC driver to access the remote repository.

When you run a node as an Administrative agent, you must remember the following:

1. You cannot connect an Administrative Console.
2. An Administrative agent does not provide a Name Service Daemon (NSD) or the LTPA security services. These requests are routed to the full Administrative Server instance via the Administrative agent.

11.8.1 High availability considerations when using an Admin agent

As the Administrative agent proxies requests through to the full Administrative Server, you cannot shut down your full Administrative Server. Should the full Administrative Server fail or be shut down, then you lose all of your administrative functionality and potentially lose servlet and EJB capabilities.

Note

Please refer to the online documentation, to ensure that this recommendation is still valid.

11.9 Using a separate AS/400 for the admin instance repository

In this section we discuss the setup requirements when utilizing a separate AS/400 to hold an Administrative Repository. This AS/400 does not have any WebSphere V3 licensed products installed on it.

The QEJB user profile is the owning and only user profile with access to an Administrative Repository. You should review the following sections in conjunction with your company's security policies.

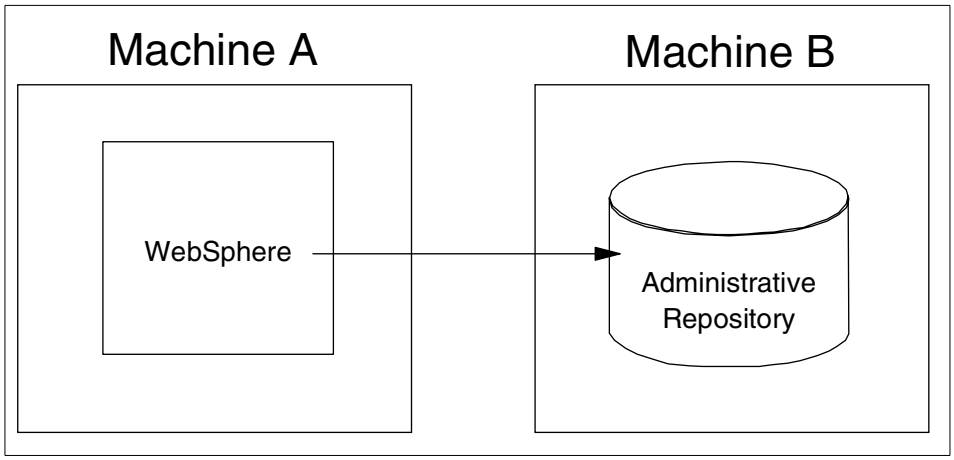

Figure 281. Separate AS/400 for Administrative repository

11.9.1 Setting up remote AS/400 to hold Administrative Repository

You need to create a QEJB user profile on the remote AS/400. This profile setup should be set up to have the same characteristics as the QEJB profile on the AS/400 with the WebSphere products, such as user class and special authorities, with the following differences:

- The user profile must have a password.
- The user profile should be set up so that it cannot sign-on.
- Optionally, the user profile should be disabled from using any Operations Navigator functionality. To do this you must set up Customized Access for each Operations Navigator application and client application functions. For further information please refer to the following redbooks
  - *Managing AS/400 V4R4 with Operations Navigator,* SG24-5646
  - *Client Access for Windows: Implementing V4R4M0,* SG24-5194

Figure 282 on page 327 through Figure 285 on page 328 show you the CRTUSRPRF parameters we specified when creating this profile. You should review these with your existing security policies before creating the user profile.

```
Create User Profile (CRTUSRPRF)

Type choices, press Enter.

User profile . . . . . > QEJB          Name
User password  . . . . . >  NEWPWD       Name, *USRPRF, *NONE
Set password to expired  . . . . > *NO          *NO, *YES
Status . . . . . >  *ENABLED     *ENABLED, *DISABLED
User class . . . . . > *USER         *USER, *SYSOPR, *PGMR...
Assistance level . . . . . > *ADVANCED     *SYSVAL, *BASIC, *INTERMED...
Current library  . . . . . > *CRTDFT       Name, *CRTDFT
Initial program to call  . . . . > *NONE         Name, *NONE
  Library  . . . . . >                   Name, *LIBL, *CURLIB
Initial menu . . . . . > *SIGNOFF      Name, *SIGNOFF
  Library  . . . . . >                   Name, *LIBL, *CURLIB
Limit capabilities . . . . . > *YES          *NO, *PARTIAL, *YES
Text 'description' . . . . . > 'IBM-supplied User Profile'



                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

Figure 282. Creating the QEJB user profile - part 1

```
Create User Profile (CRTUSRPRF)

Type choices, press Enter.


                          Additional Parameters

Special authority  . . . . . > *NONE         *USRCLS, *NONE, *ALLOBJ...
             + for more values
Special environment  . . . . . > *NONE         *SYSVAL, *NONE, *S36
Display sign-on information  . . > *YES          *SYSVAL, *NO, *YES
Password expiration interval . . > *NOMAX        1-366, *SYSVAL, *NOMAX
Limit device sessions  . . . . . > *YES          *SYSVAL, *YES, *NO
Highest schedule priority  . . . > 0             0-9
Job description  . . . . . > QDFTJOBD      Name
  Library  . . . . . >   *LIBL       Name, *LIBL, *CURLIB
Group profile  . . . . . > *NONE         Name, *NONE
Owner  . . . . . > *USRPRF       *USRPRF, *GRPPRF
Group authority  . . . . . > *NONE         *NONE, *ALL, *CHANGE, *USE...
Group authority type . . . . . > *PRIVATE      *PRIVATE, *PGP
                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display       F24=More keys
```

Figure 283. Creating the QEJB user profile - part 2

```
Create User Profile (CRTUSRPRF)

Type choices, press Enter.

Supplemental groups  . . . . . > *NONE         Name, *NONE
            + for more values
Accounting code  . . . . . . . > *BLANK
Message queue  . . . . . . . . > *USRPRF       Name, *USRPRF
  Library  . . . . . . . . . .                 Name, *LIBL, *CURLIB
Delivery . . . . . . . . . . . > *NOTIFY       *NOTIFY, *BREAK, *HOLD, *DFT
Severity code filter . . . . . > 0             0-99
Print device . . . . . . . . . > *WRKSTN       Name, *WRKSTN, *SYSVAL
Output queue . . . . . . . . . > *WRKSTN       Name, *WRKSTN, *DEV
  Library  . . . . . . . . . .                 Name, *LIBL, *CURLIB
Attention program  . . . . . . > *NONE         Name, *NONE, *SYSVAL, *ASSIST
  Library  . . . . . . . . . .                 Name, *LIBL, *CURLIB
Sort sequence  . . . . . . . . > *SYSVAL       Name, *SYSVAL, *HEX...
  Library  . . . . . . . . . .                 Name, *LIBL, *CURLIB
Language ID  . . . . . . . . . > *SYSVAL       *SYSVAL...
Country ID . . . . . . . . . . > *SYSVAL       *SYSVAL...
                                                                    More...
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure 284. Creating the QEJB user profile - part 3

```
Create User Profile (CRTUSRPRF)

Type choices, press Enter.

Coded character set ID . . . . . > *SYSVAL       *SYSVAL, *HEX...
Character identifier control . . > *SYSVAL       *SYSVAL, *DEVD, *JOBCCSID
Locale job attributes  . . . . . > *SYSVAL       *SYSVAL, *NONE, *CCSID...
            + for more values
Locale . . . . . . . . . . . . . > *sysval

User options . . . . . . . . . . > *EXPERT       *NONE, *CLKWD, *EXPERT...
            + for more values
User ID number . . . . . . . . . > 4294770716    1-4294967294, *GEN
Group ID number  . . . . . . . . > *NONE         1-4294967294, *NONE, *GEN
Home directory . . . . . . . . . > *usrprf




                                                                    Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

Figure 285. Creating the QEJB user profile - part 4

11.9.2 Configuration changes for WebSphere Administrative Server

You now have two configuration options for the WebSphere Application Server instance. Taking into account your existing security policies, you can either change the QEJB profile to have a common password or you can specify a user and password in the property files.

11.9.2.1 User profile options for connecting to the remote repository

You have two options for setting up the user profile information for connecting to the remote repository. You should review these options with your companies security policies.

Using a common password for the QEJB profiles

You can specify a common password for the QEJB profile on both systems and change the password expiry interval to *NOMAX. This helps to ensure that your passwords are always synchronized and prevents a possible failure to connect to the remote repository due to unexpected password changes. You should also change the user profile to ensure that it cannot be used to sign-on to your system via 5250, as shown in Figure 286. Optionally disable this profile from any client access functionality.

```
Command Entry                          BDFTMSS7
                                                      Request level:   4
Previous commands and messages:

  (No previous commands or messages)




                                                                      Bottom
Type command, press Enter.
===> CHGUSRPRF USRPRF(QEJB)  INLPGM(*NONE) INLMNU(*SIGNOFF)
       LMTCPB(*YES) DSPSGNINF(*YES) PWDEXPITV(*NOMAX) LMTDEVSSN(*YES)


F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
F11=Display full      F12=Cancel    F13=Information Assistant   F24=More keys
```

Figure 286. Changing the QEJB profile

Setting a user ID and password in the admin.properties file

Alternatively, you can specify the user and password information in the admin.properties file in the <was_instance_root>/properties directory. To do this you must adjust two properties shown below:

Table 18. User ID and password in the admin.properties file

Property to edit	New value
admin.dbUser	QEJB
admin.dbPassword	password for QEJB profile on remote AS/400

11.9.2.2 JDBC driver options for connecting to the repository

Currently, the only thoroughly tested JDBC driver that can be utilized to connect to a remote repository is the Java Toolbox JDBC driver. Additional admin.properties file changes are required to add support for this JDBC driver as shown in Table 19.

Table 19. Additional admin.property changes

Property to edit	New value
admin.dbUrl	jdbc:as400://RALYAS4C;lob threshold=1050000;prompt=false
admin.dbDriver	com.ibm.as400.access.AS400JDBCDriver

These changes are related to the setup differences between the Java Toolbox driver and the Native JDBC driver. We must specify the Java Toolbox style JDBC URL and the JDBC driver information. It is essential that lob threshold=1050000 be specified as this parameter disables the use of Large Objects (LOB) locators within the WebSphere environment. Instead, LOBs are processed directly rather than through a locator.

Note

You should replace the RALYAS4C with the TCP/IP host information of the AS/400 holding your remote repository.

Additionally, we need to add the AS/400 Java toolbox JAR files into the WebSphere Administrative Server classpath. To do this we add a : at the end of the existing classpath entry and add a new line containing the fully qualified path to the AS/400 Toolbox JAR file shown in Figure 287 on page 331.

```
admin.classpath=/QIBM/ProdData/WebASAdv/lib/wsa400.jar:
admin.classpath=+/QIBM/UserData/WebASAdv/new/properties:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ibmwebas.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/servlet.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/webtlsrn.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ns.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/ejs.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/ujc.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/repository.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/admin.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/tasks.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/x509v1.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/databeans.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ejscp.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/lotusxsl.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/xml4j.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/dertrjrt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/sslight.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/iioprt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/iioptools.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/deployTool.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/vaprt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/console.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/swingall.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ibmjndi.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/jsp10.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/properties:
admin.classpath=+/QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar
```

Figure 287. Changing the Administrative Server classpath for the Java Toolbox JDBC driver

11.10 Creating a cluster

A cluster is created when two or more nodes use the same Administrative Repository.

All other nodes in the cluster have to use the remote repository. This is done by following the steps in 11.14.0.2, “Remote Administrative Repository” on page 205 for each node.

For each of the nodes, the Administrative Repository must use the same database schema. This is kept in the `admin.dbSchema` property of the <was_instance_root>/properties/admin.properties file. The value for this property must be the same for all nodes. The default schema is `ejsadmin`.

The repository can either reside:

1. Locally on one of the nodes, and remotely for additional nodes
2. Remotely for all nodes

11.10.1 Local repository for first node

WebSphere defaults to using a local Administrative Repository. If you wish to use this for one of the nodes, starting the Administrative Server in the normal manner will create the repository.

11.10.2 Remote repository for first node

See 11.9, “Using a separate AS/400 for the admin instance repository” on page 325.

11.10.3 Remote repository additional nodes

Excluding the setup of the remote AS/400, this is essentially the same process as defined in 11.9, “Using a separate AS/400 for the admin instance repository” on page 325.

Please review this section in conjunction with your company’s security policies.

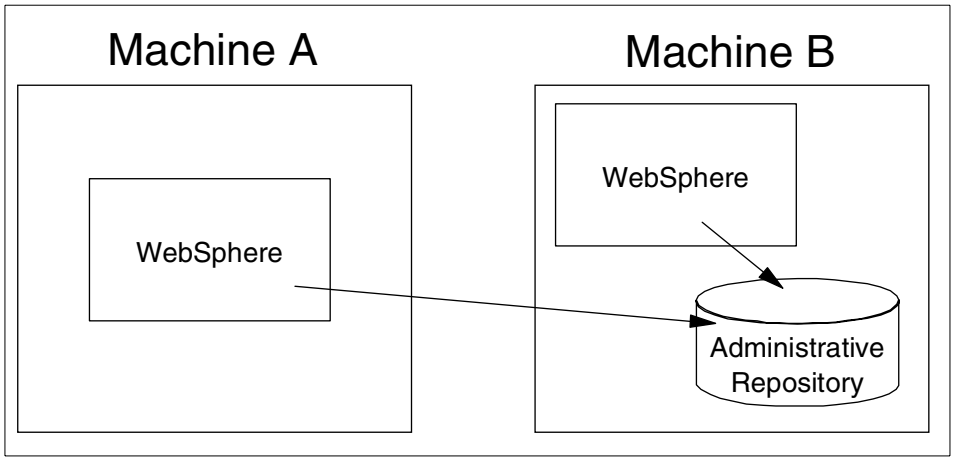

Figure 288. Connecting an additional Administrative Server to an existing Administrative Repository

11.10.4 Configuration changes for the remote AS/400

You must specify a password for the QEJB profile and change the password expiry interval to *NOMAX. Additionally it is recommended that:

- The user profile should be set up so that it cannot sign-on.
- Optionally, the user profile should be disabled from using any Operations Navigator functionality. To do this you must set up Customized Access for

each Operations Navigator Application and Client Applications function available. For further information please refer to the following redbooks:

- *Managing AS/400 V4R4 with Operations Navigator,* SG24-5646
- *Client Access for Windows: Implementing V4R4M0,* SG24-5194

```
Command Entry                          BDFTMSS7
                                                        Request level:   4
Previous commands and messages:

  (No previous commands or messages)




                                                                      Bottom
Type command, press Enter.
===> CHGUSRPRF USRPRF(QEJB)  INLPGM(*NONE) INLMNU(*SIGNOFF)
       LMTCPB(*YES) DSPSGNINF(*YES) PWDEXPITV(*NOMAX) LMTDEVSSN(*YES)


F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
F11=Display full      F12=Cancel    F13=Information Assistant   F24=More keys
```

Figure 289. Changing the QEJB profile to stop 5250 signon

11.10.5 Configuration changes on additional Admin Server

This process is identical to 11.9.2, “Configuration changes for WebSphere Administrative Server” on page 328. Please refer to that section.

11.11 Configuring administrative instance to be an Admin agent

When you configure a WebSphere Administrative Server to run as an Administrative agent, the administrative requests are routed to your full Administrative Server instance. Unlike connecting another Full Administrative Server only the instance properties need to be changed, so no user profile changes are required.

11.11.1 Modifying the instance properties files

Before we can change the property files we need two pieces of information about the full Administrative Server instance that we will connect to. These are:

Table 20. Parameters for a full Administrative Server

Information	Description
Host Name	The host name of the AS/400 system on which the full-service Administrative Server is running
administrative.bootstrapPort	The admin.bootstrapPort value that is specified in the admin.properties file for the full-service Administrative Server

The host name must be defined in either the AS/400 host table or one of the DNS servers used by the AS/400 to resolve host names.

Using this information we must edit the admin.properties file in the <was_insatnce_root>/properties directory. Additionally we must comment out the following lines in the properties file by insert a ; at the start of the line:

admin.dbDriver
admin.dbSchema
admin.dbUrl
admin.dbUser
admin.dbPassword
admin.nameServiceJar
admin.jarFile (all occurrences)

For example, in 11.5, “Multi-instance support for WebSphere V3 Advanced” on page 316, we create a new WebSphere Administrative Server instance. If we were to connect an Administrative agent to this full Administrative Server our required parameters would be:

Table 21. Parameters for an Administrative agent

Information	Value
Host Name	RALYAS4B
administrative.bootstrapPort	7799

We would make the following changes to the admin.properties file:

```
#   Administrative Server properties:
#
#      admin.dbSchema - name of Administrative repository (AS/400 collection)
#      admin.dbUrl - URL location of the administrative database (local default)
#      admin.dbDriver - qualified class name of the database driver
#      admin.dbUser - optional user id to access administrative databases
#      admin.dbPassword - optional password to access administrative databases
#      admin.bootstrapPort - TCP/IP port number of the admin server
#      admin.lsdPort - TCP/IP port number of location server daemon
#      admin.traceString - trace options (see documentation for options)
#      admin.traceOutput - trace output
#      admin.jarFile - admin server jar files
#      admin.nameServiceJar - name server jar file
#      admin.initializer - initializer classes (multiple initializers allowed)
#      admin.classpath - Websphere classpath (=+ to concatenate multiple lines)
#      admin.instance.root - server root and admin/app server default current
#                            working directory
#
##############################################################################

;admin.dbSchema=ejsadmin
;admin.dbUrl=jdbc:db2:*local
;admin.dbDriver=com.ibm.db2.jdbc.app.DB2Driver
;admin.dbUser=
;admin.dbPassword=
admin.bootstrapPort=7799
admin.lsdPort=9000
admin.bootstrapHost=RALYAS4B
;admin.nodeName=
;admin.nameServiceJar=/QIBM/ProdData/WebASAdv/lib/server/ns.jar
;admin.jarFile=/QIBM/ProdData/WebASAdv/lib/server/repository.jar
;admin.jarFile=/QIBM/ProdData/WebASAdv/lib/server/tasks.jar
admin.traceString=com.ibm.*=all=disabled
admin.traceOutput=logs/tracefile
admin.logFile=tranlog/tranlog1,tranlog/tranlog2
admin.initializer=com.ibm.ejs.security.Initializer
admin.initializer=com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer
admin.initializer=com.ibm.servlet.config.AS400SetupInitializer
```

Note

These files have an ASCII code page and must have an ASCII code for the WebSphere Admistrative Server instance to work.

11.11.2 Administrative agent template

We have provided a template in order to help you gather the information required when configuring a new WebSphere Administrative Server instance as an Administrative agent. The Appendix F, “Multi-instance template for

AS/400 Admin-agent mode" on page 529 also provides a before and after view of the property files using our example configuration.

11.12 Persistent session information

If you will be utilizing persistent HTTP sessions in your environment and you plan to use horizontal scaling techniques for WebSphere workload management you should create a persistent DataSource/JDBC driver using the Java Toolbox JDBC driver. As only a single DataSource and JDBC driver are specified for all of the clones it must work across all nodes in your environment.

If you are only using vertical scaling techniques then you should use the AS/400 Native JDBC driver.

11.13 Configuring an HTTP server instance

Unlike Windows and the UNIX versions of WebSphere Administrative Server V3, the configuring of an HTTP server instance to connect to a WebSphere Administrative Server instance envolves some manual steps. This is due to the multi-instance nature of both WebSphere V3 and the HTTP server on the AS/400.

Similar restrictions to Windows and UNIX apply when configuring a HTTP server instance on the AS/400, in that a HTTP server instance can only be connected to a single WebSphere Administrative Server instance. However, multiple HTTP server instances can be used to route/forward servlet requests to a WebSphere Administrative Server instance on a single AS/400 as shown in Figure 290 on page 337. Additionally, a HTTP server instance can only be configured to connect either to a WebSphere V2 or V3 instance.

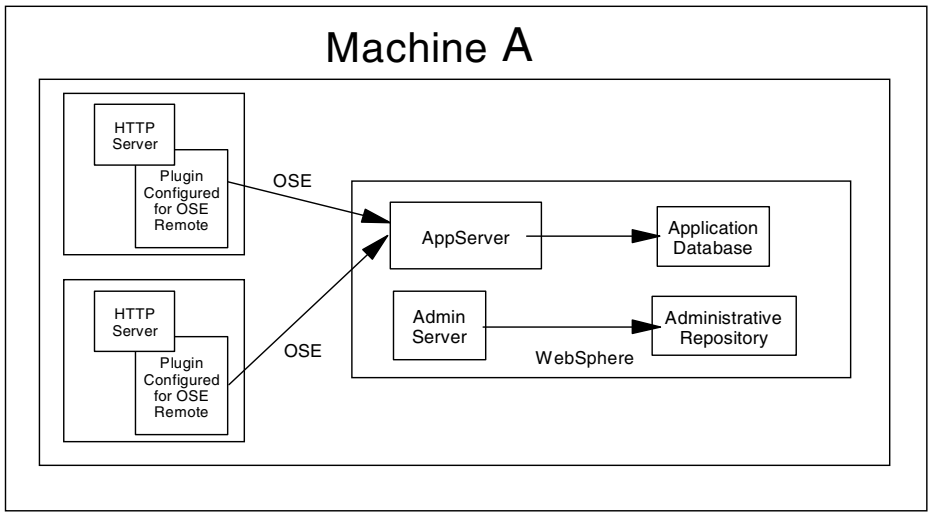

Figure 290. Multiple HTTP instances connected to an Administrative Server on a single AS/400

Typically the HTTP servers would be run on separate machines.

We assume that you are already familiar with creating a new HTTP server instance and working with existing HTTP server instances. If you are unfamiliarly with this process then please refer to the following:

HTTP Server for AS/400 Webmaster's Guide V4R4, GC41-5434

Remember that you must use unique TCP/IP port numbers for your HTTP server if the default HTTP and HTTPS, ports 80 and 443 respectively are being utilized by another HTTP server instance. If you cannot use the default ports for HTTP and HTTPS then you must append the port numbers to the host alias information inside your WebSphere Administrate Server instance.

11.13.1 Automated configuration of a HTTP server instance

WebSphere Application Server Advanced Edition V3 Group PTF 1, includes two PTF's that enhance the HTTP Adminstrative Server support for configuring an HTTP server instance for WebSphere V3 support.

Table 22. OS/400 versions and Group PTFs

OS/400 Version	Group PTF for 5769-WA2	Group PTF for 5769-WA3
V4R4	SF99028	SF99029

OS/400 Version	Group PTF for 5769-WA2	Group PTF for 5769-WA3
V4R5	SF99135	SF99136

These PTF's allow configuration of a new HTTP server instance or reconfiguration of an existing HTTP Server instance from WebSphere V2 to V3 in a simple and error free manner.

One restriction in using this method is that the <was_instance_root> is expected to be in the '/QIBM/UserData/WebAsAdv' IFS directory.

11.13.1.1 HTTP Administrative Server browser interface

Using then the HTTP Administration Server browser interface, work with your HTTP server instance configuration file, in our example JJD. Click on the **Java servlets** and you will be presented with the Configuration and Administration Java servlets HTML page. Enable support for WebSphere V3 by clicking in the radio box next to the text "WebSphere V3" which should automatically tick the check box for Servlet and JSP support. From the WebSphere domain drop down list, select the WebSphere Application Server instance you require. Then click on the **Apply** button, as shown in Figure 291 on page 339.

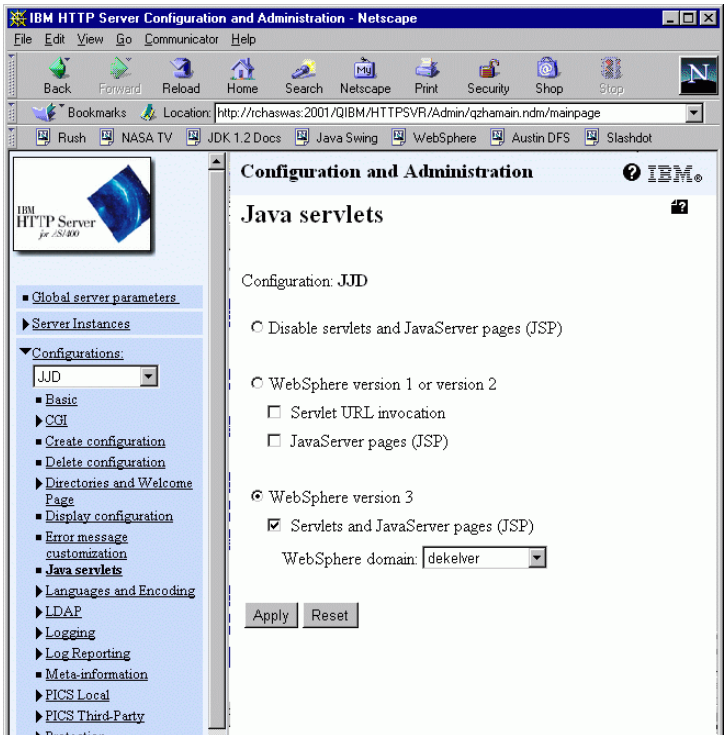

Figure 291. Automated HTTP Server instance configuration

This automatically removes any WebSphere V2 configuration information that may have existed in the HTTP server instance configuration.

Note

The information displayed in the WebSphere drop down list is the WebSphere mulit-instance directories that exist in the /QIBM/UserData/WebAsAdv IFS directory.

11.13.2 Manually configuring a HTTP server instance

Two options exist for manual configuration of a HTTP server instance. Either you utilse the OS/400 commands or the HTTP Adminstartive server browser interface to perform the configuration. Both processes are described bellow.

11.13.2.1 WRKHTTPSVR command

Using the Work with HTTP Configuration command you must add the following information to the HTTP server configuration.

HTTP methods

These are the CGI scripts and servlet methods that will be allowed by the HTTP server. We recommend that you enable the following:

Table 23. HTTP methods

HTTP Allowed Methods
Enable Get
Enable Post
Enable Head

Note: Head is required for an *IBM Network Dispatcher* environment

Request routing entries

The request routing entries allow the HTTP server to route, or forward, requests to the WebSphere Application Server instance. They must be added in the order they are shown, NameTrans and Service followed by the Pass statements. In a production environment it is unlikely that you would include the Pass statements.

Table 24. Request routing

Request Routing
NameTrans /* /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:nametrans_exit
Service IBMWebSphere /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:service_exit
Pass /theme/* /QIBM/ProdData/WebASAdv/theme/*
Pass /WebSphereSamples/* /QIBM/ProdData/WebASAdv/WebSphereSamples/*

Typically, you would place these entries before any other Request Routing entries, such as Pass, Map and Fail. This ensures that any HTTP request for WebSphere resources is managed and mapped correctly for WebSphere. If these statements are placed after other Request Routing statements, the HTTP request may be inadvertently modified and cause resources, such as servlets, not to be found.

Server API application processing

These statements define the WebSphere HTTP plug-in. Replace the <was_instance_root> with the instance root for your WebSphere Application

Server instance; refer to 11.5, “Multi-instance support for WebSphere V3 Advanced” on page 316 for further information.

Table 25. Server API

Server API
ServerInit /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:init_exit <was_intance_root>/properties/bootstrap.properties (A space should separate these on a single configuration line)
Authorization * /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:authorization_exit
ServerTerm /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:term_exit

Although these statements can be placed anywhere within your HTTP configuration file, we recommend that you place them at the end of the file.

11.13.2.2 HTTP Administrative Server browser interface

Using the HTTP Administration Server browser interface, work with your HTTP server instance configuration file, in our example W3_TRADE. Click **Request processing** and then **Methods** in the left-hand navigation frame. You will see an HTML page similar to Figure 292 on page 342. You must enable the Get, Post and Head methods by clicking the relevant check box. A tick will appear in the check box next to the option when enabled. Click the **Apply** button to change the HTTP configuration.

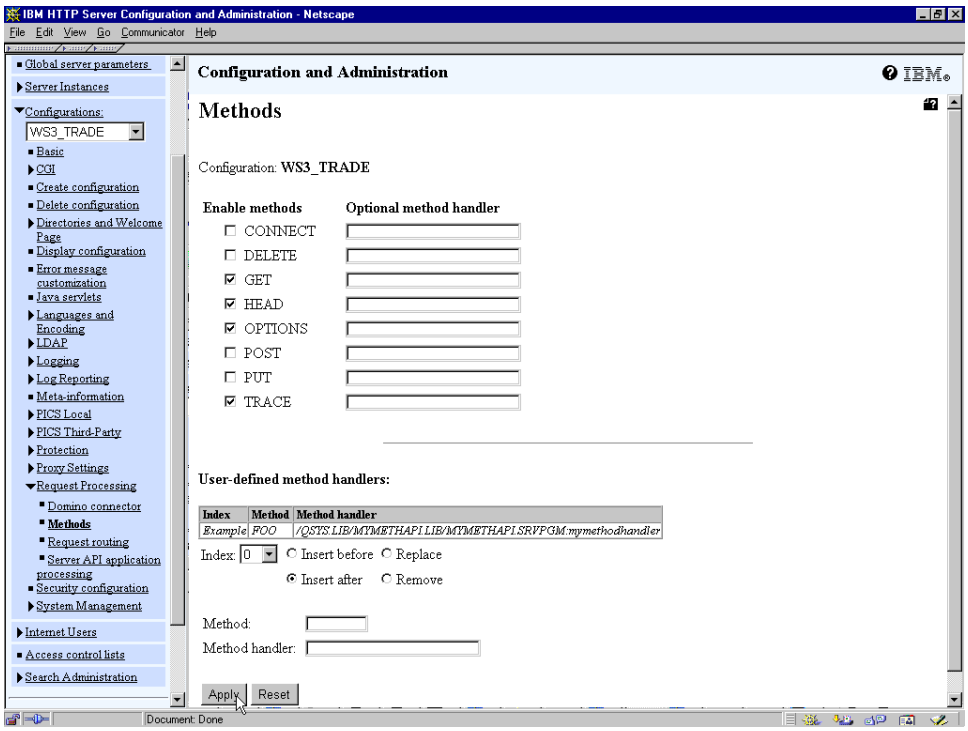

Figure 292. Enabling the required HTTP server methods

Now click **Request Routing** from the left-hand navigation frame. You will see the current routing configuration information in the right-hand browser frame. Typically, you would place the WebSphere V3 Requested Routing entries before any other Request Routing entries, such as Pass, Map and Fail you may have. This ensures that any HTTP request for WebSphere resources are managed and mapped correctly for WebSphere. If these statements are placed after other Request Routing statements, the HTTP request may be inadvertently modified and cause resources, such as servlets, not to be found. You can choose the placement of an entry by selecting an **Index** number and selecting **Insert before** or **Insert After**. The following entries should be added by filling in the fields and clicking the **Apply** button. They must be added in the order specified and the first routing entry should be

index 1. If you already have entries, simply select 1 for the **Index** number and **Insert before** for the first routing entry.

Table 26. WebSphere Routing entry, index1

Action:	NameTrans
URL template:	/*
Replacement file path:	/QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:nametrans_exit

Table 27. WebSphere Routing entry, index2

Action:	Service
URL template:	IBMWebSphere
Replacement file path:	/QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:service_exit

Table 28. WebSphere Routing entry, index 3

Action:	Pass
URL template:	/theme/*
Replacement file path:	/QIBM/ProdData/WebAsAdv/theme/*

Table 29. WebSphere Routing entry, index4

Action:	Pass
URL template:	/WebSphereSamples/*
Replacement file path:	/QIBM/ProdData/WebAsAdv/WebSphereSamples/*

In a production environment it is unlikely that you would include the Pass statements. Figure 293 on page 344 shows the process for adding the routing entries.

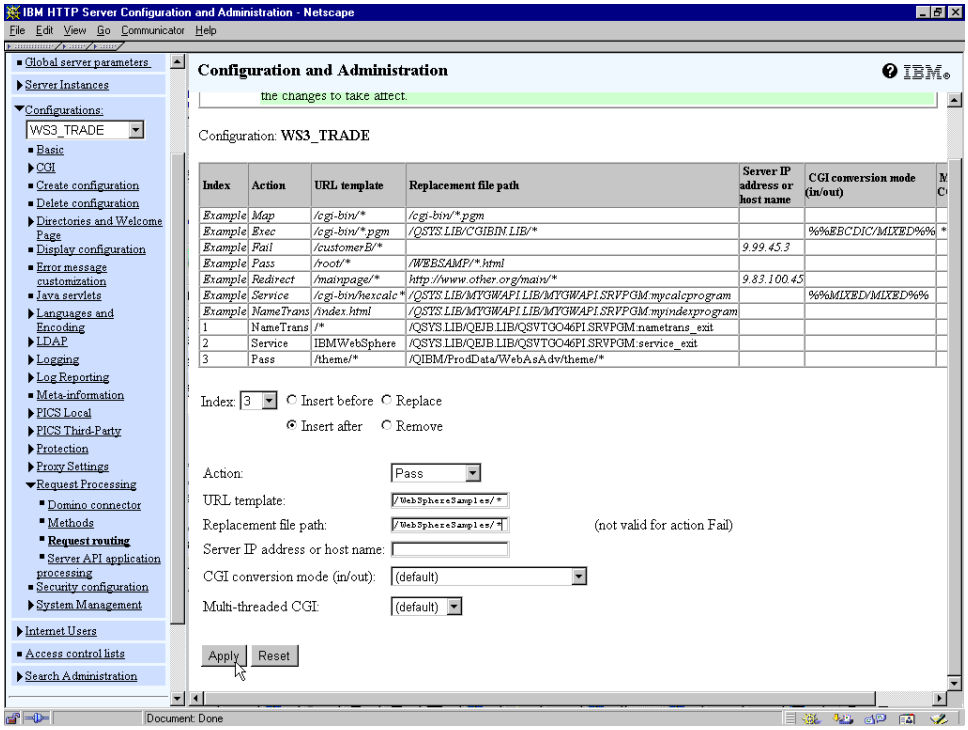

Figure 293. Enabling the required HTTP request routing

Click **Server API application processing** from the left-hand navigation frame. You will see the current configuration information in the right-hand browser frame. Add the WebSphere initialization entries by filling in the fields as follows and then clicking the **Apply** button. Replace the <was_instance_root> with the instance root for your WebSphere Application

Server instance, refer to 11.5, “Multi-instance support for WebSphere V3 Advanced” on page 316 for further information.

Table 30. ServerInit

ServerInit	
Url template	
Application path and file name	ServerInit /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:init_exit <was_intance_root>/properties/bootstrap.properties (A space should separate these on a single configuration line)

Table 31. Authorization

Authorization	
Url template	WebSphere
Application path and file name	/QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:authorization_exit

Table 32. ServerTerm

ServerTerm	
Url template	
Application path and file name	/QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:term_exit

Figure 294 on page 346 shows the process for adding the Server API application processing entries.

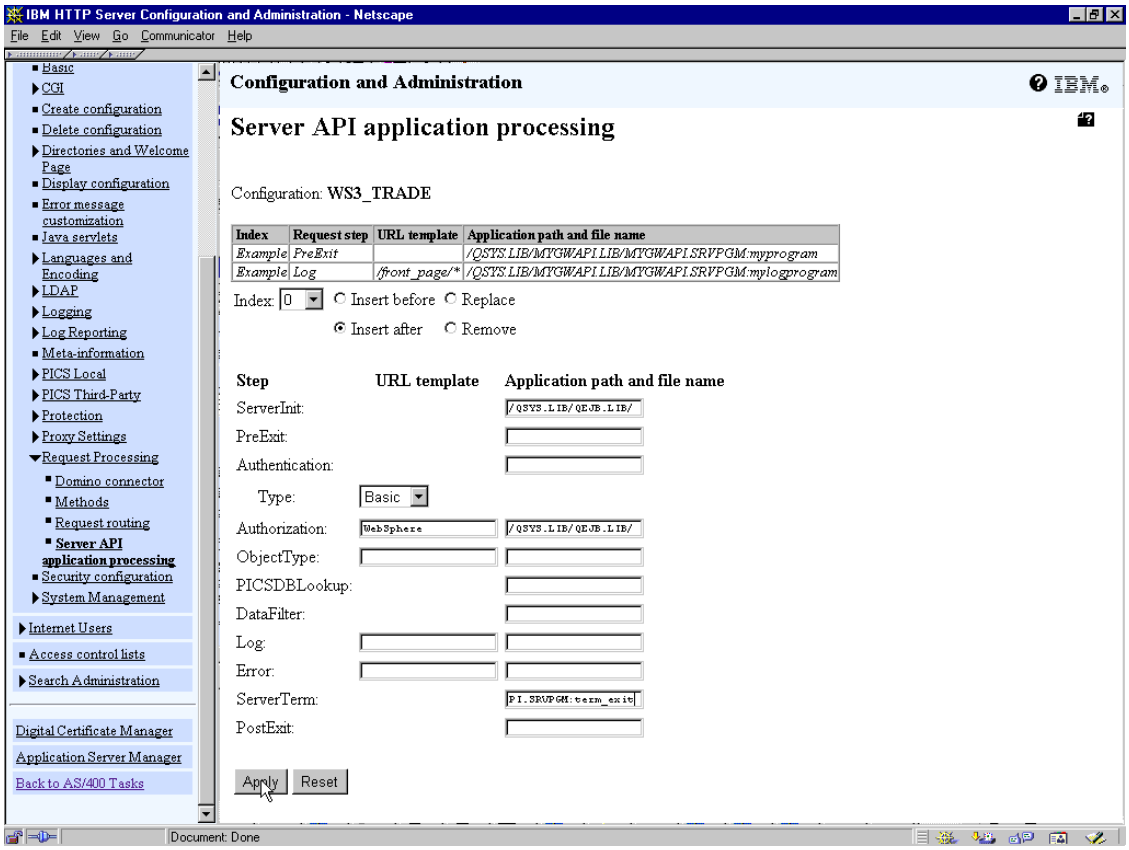

Figure 294. Enabling the required HTTP server API application processing

11.13.3 Manually reconfiguring HTTP server instance from V2 to V3

If you have an HTTP server instance that is configured to utilize a WebSphere V2 instance then you must remove the V2 configuration before configuring for V3.

You should also review your Request Routing Statements to ensure they do not reflect your V2 configuration. If so, you should update these for your V3 environment.

11.13.3.1 Using the WRKHTTPSVR command

Four statements in the HTTP server configuration file need to be removed. These are:

Table 33. HTTP configuration statements

HTTP Configuration statements
`Service /*.jsp /QSYS.LIB/QAPPSVR.LIB/QZHJSVLT.SRVPGM:AdapterService`
`Service /servlet/* /QSYS.LIB/QAPPSVR.LIB/QZHJSVLT.SRVPGM:AdapterService`
`ServerInit /QSYS.LIB/QAPPSVR.LIB/QZHJSVLT.SRVPGM:AdapterInit /QIBM/UserData/IBMWebAs/HMVWEB1/properties/bootstrap.properties`
`ServerTerm /QSYS.LIB/QAPPSVR.LIB/QZHJSVLT.SRVPGM:AdapterExit`

Note

If your are utilizing multiple WebSphere Version 2 instances, then the ServerInit statement may appear slightly different.

Now add the WebSphere V3 HTTP server configuration statements as described in 11.13.2, “Manually configuring a HTTP server instance” on page 339.

11.13.3.2 HTTP Administrative Server browser interface

Using the HTTP Administration Server browser interface, work with your HTTP server instance configuration file, in our example V2TEST. Click **Java servlets** and you will be presented with the Configuration and Administration Java servlets HTML page. Deselect support for both URL invocation and JavaServer Pages by clicking in the check boxes to remove the tick that may be there. Then click **Apply** button to remove the WebSphere V2 configuration, as shown in Figure 295 on page 348.

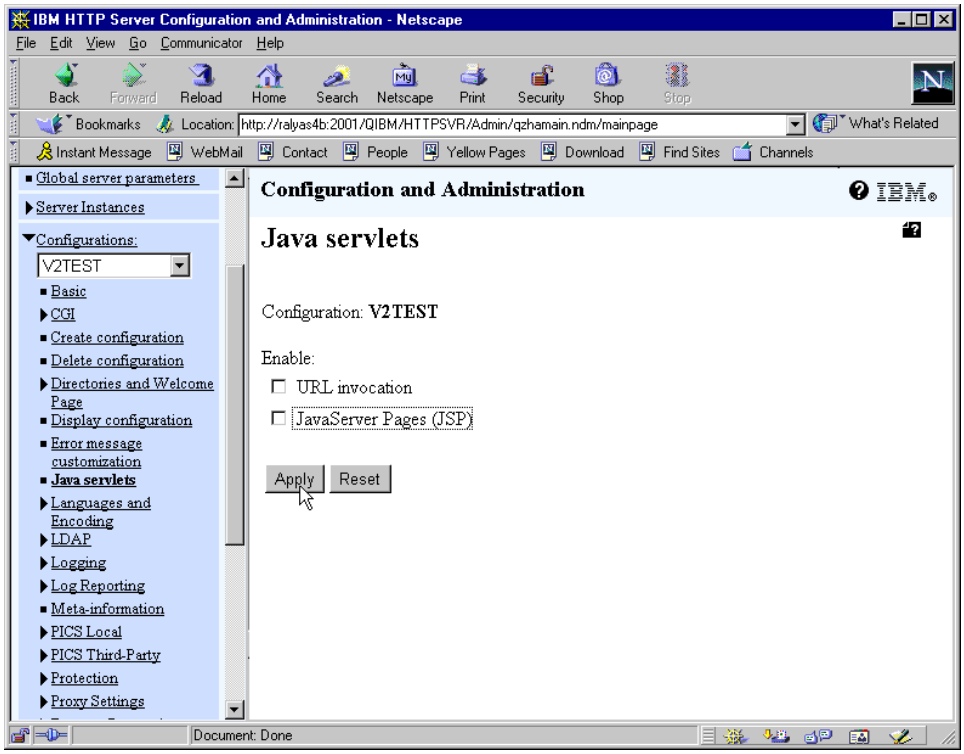

Figure 295. Removing WebSphere V2 from an HTTP server instance configuration

Now add the WebSphere V3 HTTP server configuration statements as described in 11.13.2, “Manually configuring a HTTP server instance” on page 339.

11.14 Using AS/400 HTTP server in Network Dispatcher environment

This section describes the AS/400 specific configurations required to use an AS/400 in a Network Dispatcher load balanced environment. Figure 296 on page 349 shows a simple high availability scenario that we will use to discuss the AS/400 setup.

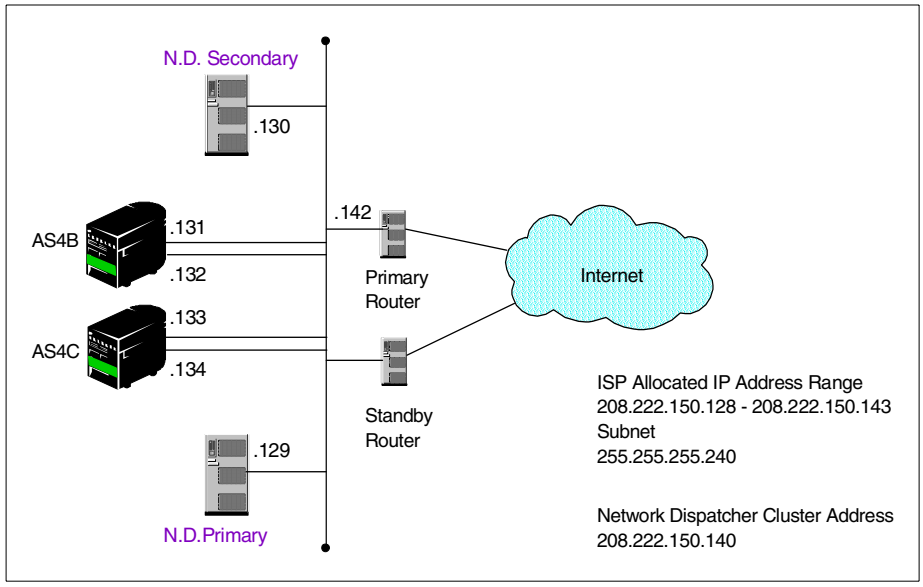

Figure 296. Simple AS/400 HTTP high availability environment with Network Dispatcher

Note

Figure 296 shows a simple high availability scenario and does not take into account all security aspects or design choices available.

The existing WebSphere Performance Pack redbook, *IBM WebSphere Performance Pack: Load Balancing with IBM secureWay Network Dispatcher,* SG24-5858 provides extensive information for configuring non-AS/400 environments for use with the Network Dispatcher. This redbook talks about defining an alias to the cluster address on the loopback interface when configuring the Web server machines. On the AS/400 you should equate this with creating an OS/400 TCP/IP virtual interface or virtual IP address for the cluster address. A virtual IP address can be thought of as a “System” IP address, which can only be contacted via an indirect route through a real TCP/IP interface. As virtual IP addresses do not respond to ARP requests, the same virtual IP address can be defined on multiple machines without causing any ARP conflicts and allow several AS/400 systems to appear as a single system to the outside world. When used in a Web environment, you would usually define a host entry for the virtual IP address. You would then

configure your HTTP server to specifically bind to this host or virtual IP address.

11.14.1 Configure AS/400 for Network Dispatcher Web environment

11.14.1.1 Creating the virtual IP address

You can create a virtual IP address from either a 5250 session or client access provided the user profile you are using has *IOSYSCFG special authority. We have chosen to create our address via a 5250 session as shown in Figure 297.

```
Add TCP/IP Interface (ADDTCPIFC)

Type choices, press Enter.

Internet address . . . . . > 208.222.150.140
Line description . . . . . *VIRTUALIP    Name, *LOOPBACK...
Subnet mask  . . . . . 255.255.255.255
Associated local interface . . . *NONE
Type of service  . . . . . *NORMAL       *MINDELAY, *MAXTHRPUT...
Maximum transmission unit  . . . 16388          576-16388, *LIND
Autostart  . . . . . *YES          *YES, *NO
PVC logical channel identifier                  001-FFF
            + for more values
X.25 idle circuit timeout  . . . 60             1-600
X.25 maximum virtual circuits  . 64             0-64
X.25 DDN interface . . . . . *NO            *YES, *NO
TRLAN bit sequencing . . . . . *MSB          *MSB, *LSB



                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 297. Create a virtual IP address

You must specify an MTU size when creating a virtual IP address as this is a system IP address and not associated with a physical interface. The real MTU size is determined by the route and physical interface taken out of the system. Further information is available in the following redbook: *V4 TCP/IP for AS/400: More Cool Things Than Ever,* SG24-5190

11.14.2 Outbound IP load balancing

Depending on the number of physical interfaces connected to the Internet LAN segmen,t your TCP/IP configuration may be causing a bottlneck for your outbound TCP/IP traffic.

Typically, for an AS/400 with a single physical interface you would define only a single default route without specifying a preferred binding interface as shown in Figure 298.

```
Work with TCP/IP Routes
                                                        System:   AS4B
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display

     Route            Subnet           Next             Preferred
Opt  Destination      Mask             Hop              Interface

     *DFTROUTE        *NONE            208.222.150.142 *NONE




                                                                  Bottom
F3=Exit      F5=Refresh   F6=Print list   F11=Display type of service
F12=Cancel   F17=Top      F18=Bottom
```

Figure 298. Single default route with no preferred default route

The OS/400 TCP/IP stack would direct all outbound traffic via the single physical interface. When you have multiple physical interfaces on a single AS/400 as shown in Figure 296 on page 349, and only a single default route with no preferred binding interface defined, the TCP/IP stack would direct all outbound TCP/IP traffic over the first available interface only. In a high volume environment your outbound TCP/IP traffic could become bottlenecked by over-utilization of this physical interface.

To ensure that you can utilize all your physical interfaces for outbound TCP/IP traffic, you must specify multiple default routes, one for each physical interface, specifying a preferred binding interface of the actual IP address of the physical interface and a duplicate route priority value of 6. An example of the AS/400 `ADD TCP/IP Route (ADDTCPRTE)` command is shown in Figure 299 on page 352.

```
Command Entry                          AS4B
                                                        Request level:   4
Previous commands and messages:

  (No previous commands or messages)




                                                                       Bottom
Type command, press Enter.
===> ADDTCPRTE RTEDEST(*DFTROUTE) SUBNETMASK(*NONE) NEXTHOP('208.222.150.142')
       BINDIFC('208.222.150.131') DUPRTEPTY(6)


F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
F11=Display full      F12=Cancel    F13=Information Assistant   F24=More keys
```

Figure 299. Adding a default route with a preferred binding interface

By adding multiple default routes this enables the OS/400 TCP/IP stack to send the outbound TCP/IP traffic back through any of the physical interfaces defined for the default route in a round robin fashion, allowing us to increase the outbound TCP/IP traffic throughput dramatically. As the virtual IP address will be used as the source IP address in the outbound TCP/IP packets, the TCP/IP client will be unaware that the outbound physical interface may not have been the inbound physical interface. Figure 300 shows the Work with TCP/IP routes (WRKTCPRTE) display after adding the multiple default routes as shown in Figure 300.

```
Work with TCP/IP Routes
                                                        System:   AS4B
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display

     Route            Subnet           Next             Preferred
Opt  Destination      Mask             Hop              Interface

     *DFTROUTE        *NONE            208.222.150.142  208.222.150.131
     *DFTROUTE        *NONE            208.222.150.142  208.222.150.132




                                                                       Bottom
F3=Exit      F5=Refresh   F6=Print list   F11=Display type of service
F12=Cancel   F17=Top      F18=Bottom
```

Figure 300. Multiple default routes with preferred binding interface

Note

The Duplicate Route Priority value does not have to be 6; it can be any value higher than the default of 5.

If you do not specify the same value for each of your default routes, then the OS/400 TCP/IP stack will only utilize the default route(s) with the highest duplicate route priority.

11.15 AS/400 WebSphere in heterogeneous WebSphere environment

Unlike other operating system environments discussed within this book, the AS/400 WebSphere Application Server plug-ins are currently designed to operate with a pure EBCDIC environment. This curtails some of the following scenarios to an AS/400 homogeneous environment in that you can only utilize an AS/400 HTTP server as the Web Server when utilizing OSE remote and the thin Servlet Redirector.

Heterogeneous scenarios where your servlets run on non-AS/400 platforms and your EJB's run on AS/400s, or vice-versa, work without any issues.

Within subsequent releases of the WebSphere Application Server, the heterogeneous limitations should be removed.

11.16 Current AS/400 JDBC driver limitations

11.16.1 Two phase commit capabilities

Due to limitations in the AS/400 host servers, two phase commit is not available when using the Java Toolbox JDBC driver. If you require two phase commit capabilities, then you must utilize the Native AS/400 JDBC driver.

11.16.2 Accessing remote databases with native JDBC driver

Currently, DRDA is not fully implemented over TCP/IP in the OS/400 operating system. When you require remote AS/400 database access, especially when you require two phase commit or utilize large objects, such as BLOB or CLOBS, you must configure your system with the Native JDBC driver to use DRDA over SNA.

Note

For the latest information on how to set up the AS/400 Toolbox driver and the AS/400 Native Driver for remote Administrative Repository, please refer to the latest AS/400 WebSphere documentation and addendum. These documents can be found at:

www.as400.ibm.com/products/websphere/docs/doc.html#AE302

www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html

www.as400.ibm.com/products/websphere/docs/relnotes302_addendum.html

Chapter 12. Application server clones

Cloning provides a method of vertically scaling WebSphere Application Servers. The servlets, EJBs, and Web resources are shared by the clones, but each clone uses its own Java Virtual Machine (JVM) to run the application code. This provides identical, yet independent processes for the application to run in. For more information see 2.1, “Cloning” on page 15.

This chapter provides instructions for the administrative tasks required to configure a single AS/400 to be the HTTP server, application server, and database server. The application server will have clones based on a single model.

We will discuss the following steps for setting up this configuration:

1. Product installation
2. Create the model
3. Configure the model for WLM
4. Create the clones
5. Configure the HTTP server
6. Start the servers
7. Test the servers
8. Related topologies

12.1 Overview of the configuration

A single AS/400 (RALYAS4B) will run the HTTP server, application server clones, and database.

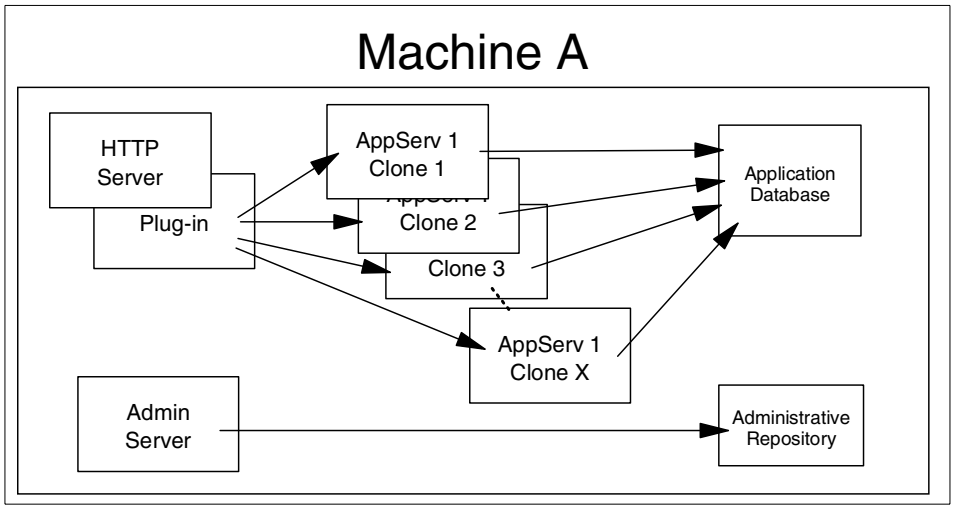

Figure 301. Clones of an application server

12.2 Installation summary

Table 34 summarizes the software products and options required to support this topology.

Table 34. Product install

Machine A	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1

Please refer to the following for further information:

- *AS/400 Software Installation,* SC41-5120
- `www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html`
- *Building AS/400 Appls for WebSphere Advanced 3.0,* SG24-5691 (available as a “redpiece” at http://www.redbooks.**ibm.com**/)

12.3 Start the Administrative Console

Before we configure the clones, we need to start the Administrative tools.

1. Start the Administrative Server on the AS/400.

```
Command Entry                          AS4B
                                                        Request level:   4
Previous commands and messages:

  (No previous commands or messages)




                                                                      Bottom
Type command, press Enter.

 ===> SBMJOB     CMD(CALL PGM(QEJB/QEJBMNTR) PARM('-P' '/QIBM/USERDATA/WEBASADV/DEFAULT/PROPERTIES
/ADMIN.PROPERTIES')) JOB(QEJBMTR) JOBD(QEJB/QEJBJOBD)
JOBQ(QEJB/QEJBJOBQ) USER(QEJB)

 F3=Exit   F4=Prompt   F9=Retrieve   F10=Include detailed messages
 F11=Display full      F12=Cancel    F13=Information Assistant   F24=More keys
```

Ensure that the Administrative Server is completely started before continuing.

2. Start the Administrative Console for the AS/400 on a workstation.

```
C:\>adminclinet RALYAS4B 900
Remote AdminServer RALYAS4B will be accessed on port 900.
```

When the Administrative Console is ready, we can start creating the model and clones.

12.4 Create the model

A model is a template for creating additional, nearly identical copies of a WebSphere item, such as an application server or servlet engine. The copies are called clones.

In the Administrative Console, select the **Topology** tab. Expand the node (RALYAS4B). Locate the application server in the list. The default server that is created is called `Default Server`.

To create the model, right click on the application server to be cloned, and select **Create ->Model** as shown in Figure 302 on page 358.

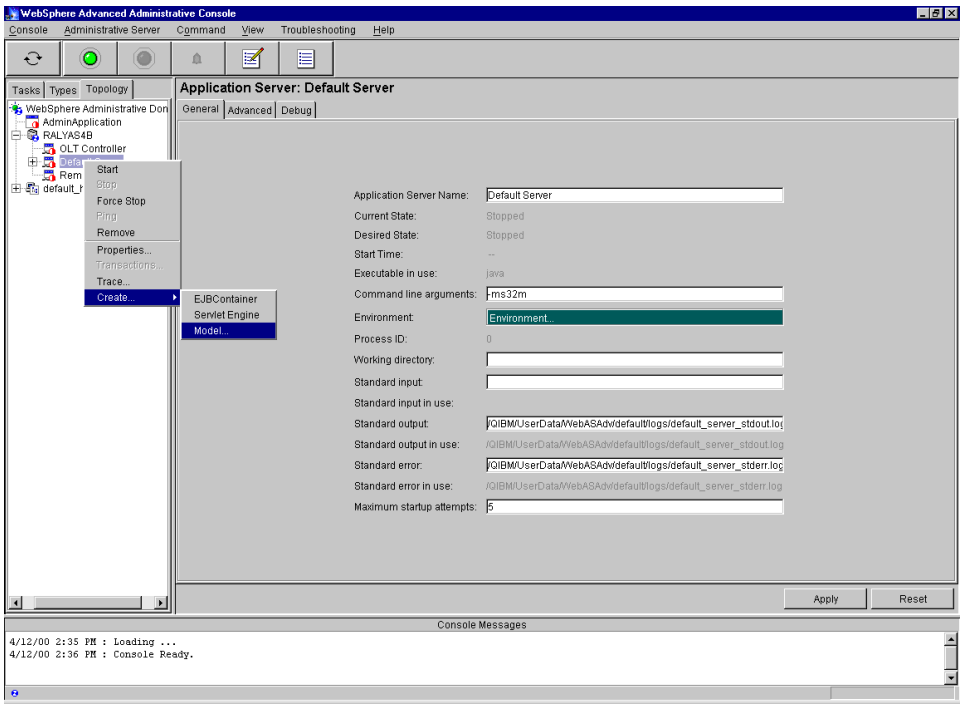

Figure 302. Create model

The Clone Properties window will appear. Enter a model name and check the box to `Recursively Model all Instances under the Server` as shown in Figure 303 on page 359.

Click the **Create** button.

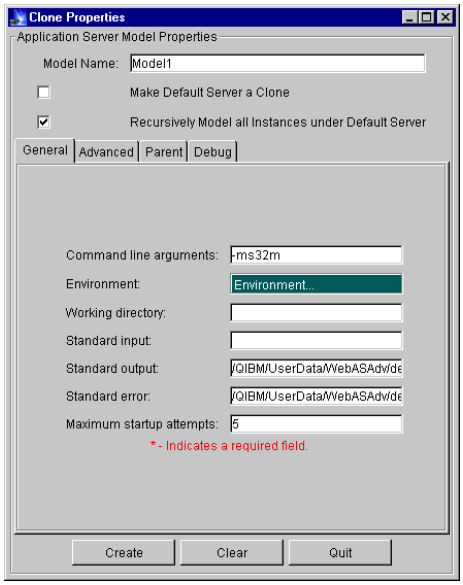

Figure 303. Clone properties

Note

Typically, you would not want to convert your initial Web application into a clone; however if you plan to enhance this topology to utilize the Servlet Redirector then you should tick the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model's clones to fail unless this server is made into a clone.

When the model has been created you should see a message as shown in Figure 304.

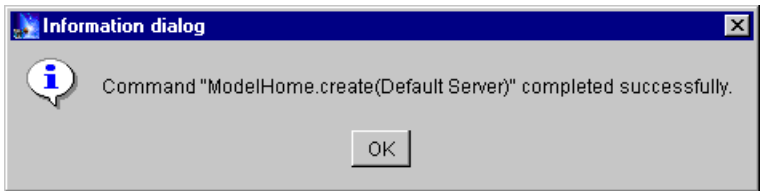

Figure 304. Model created message

12.5 Configure the model for WLM

Select the model in the **Topology** tab. Choose the **Advanced** tab. The default for workload management selection policy is `Round Robin Prefer Local`. Since all the clones will reside on the local node, you will not change this value. For more information see 2.3, “WLM” on page 23.

When `Round Robin Prefer Local` is selected, all requests that can be served locally will be served locally. If you were to clone two nodes, you would need to select `Round Robin` in order for all the clones to be used evenly.

Select the policy the model is to use as shown in Figure 305.

Click the **Apply** button if you updated.

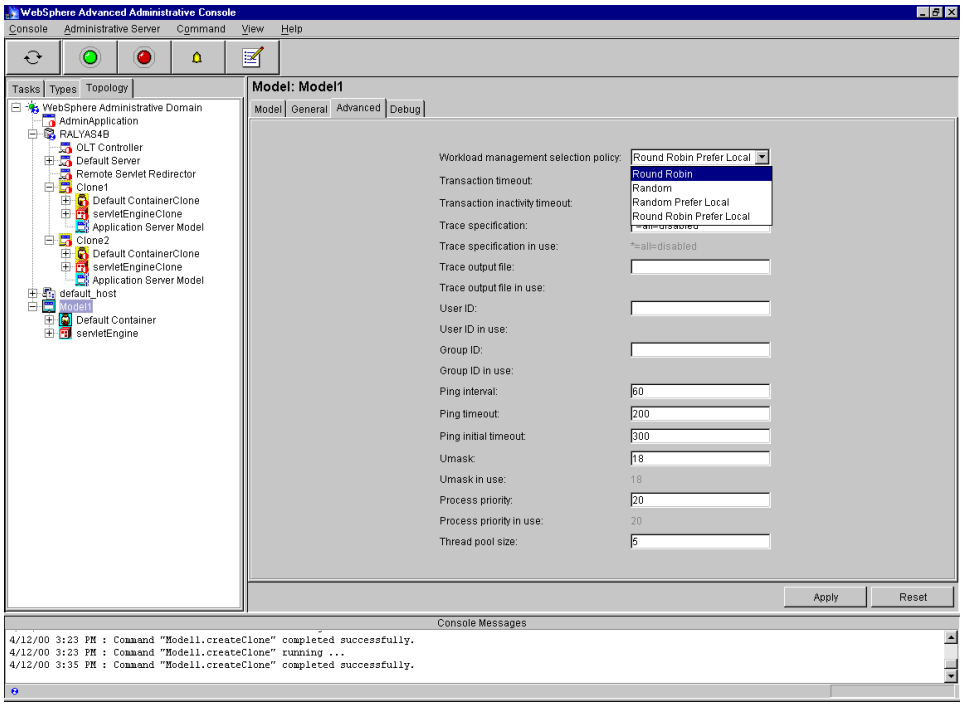

Figure 305. Model WLM

12.6 Create the clone

Using the model you created, you can now create a clone. To do this, right click on the model and select **Create->Clone** as shown in Figure 306.

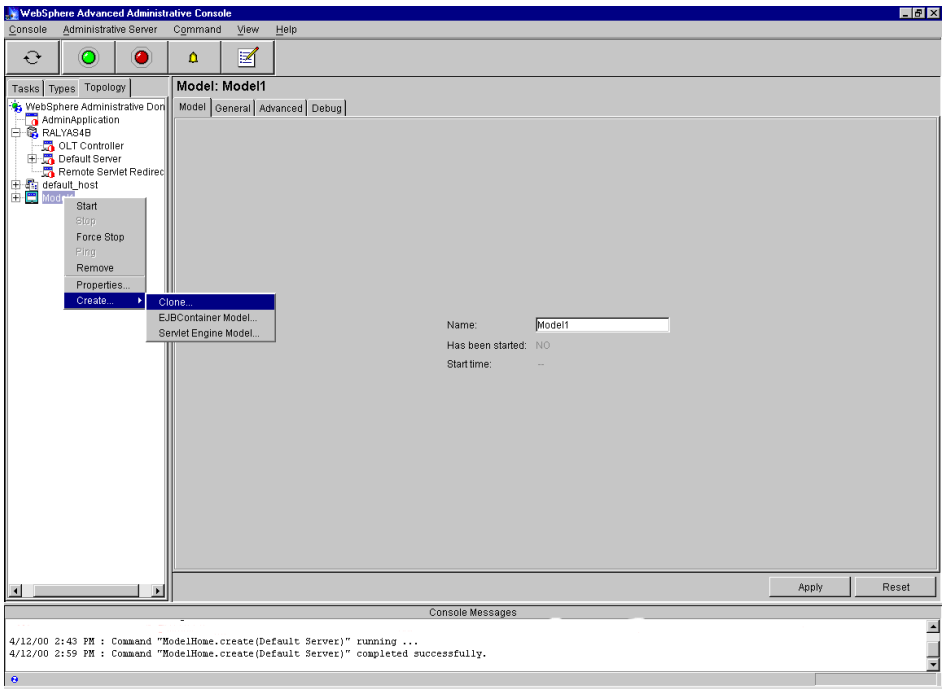

Figure 306. Create clone

The Clone Parent window will appear. Enter a clone name and select the node for the clone as shown in Figure 307 on page 361.

Click the **Create** button.

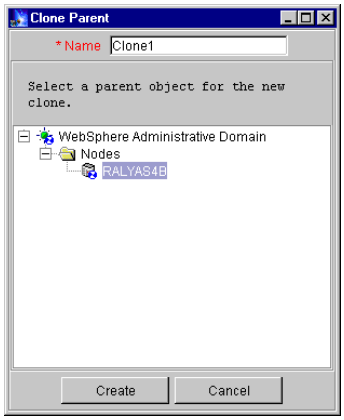

Figure 307. Clone Parent

Note

If you selected the option to make the server a clone when you generated the model, you will already have one clone associated with the model (the server you cloned).

When the clone has been created, you will see a message as shown in Figure 308.

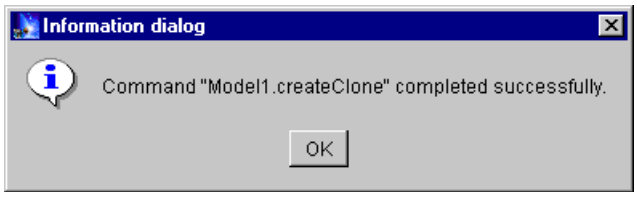

Figure 308. Clone created message

Additional Clones

Repeat this process for each clone you wish to create.

The clones will appear in the **Topology** tab as shown in Figure 309 on page 363.

In our case, the clones are called `Clone1` and `Clone2`. The model is called `Model1`. The model was created from the `Default Server`. The `Default Server` is still in the **Topology** tab, but since we did not select the option to make it a clone, it is not associated with the model.

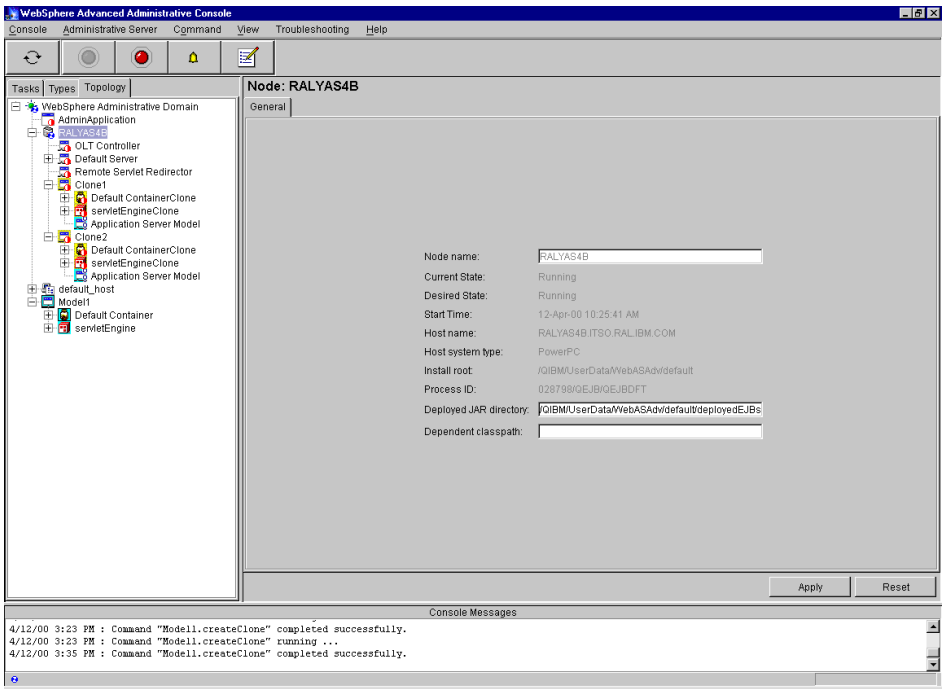

Figure 309. Clones in the topology tab

12.7 Configure the HTTP server

The following entries need to be added to the HTTP server configuration file.

These entries can be added from and AS/400 sign-on using the `WRKHTTPCFG` command or using the HTTP server Administrative Console.

1. `Enable POST`
2. `Enable GET`
3. `Enable HEAD`
4. `NameTrans /* /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:nametrans_exit`
5. `Authorization * /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:authorization_exit`
6. `Service IBMWebSphere /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:service_exit`
7. `ServerInit /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:init_exit /QIBM/UserData/WebASAdv/default/properties/bootstrap.properties`
8. `ServerTerm /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:term_exit`
9. `Pass /theme/*  /QIBM/ProdData/WebASAdv/theme/*`

```
10.Pass /html/*  /QIBM/UserData/WebASAdv/default/html/*
11.Pass /WebSphereSamples/*  /QIBM/ProdData/WebASAdv/WebSphereSamples/*
```

Note

Each entry listed above should be on a separate line. Any single entry should be on one line with no word wrap. The lines are shown here with word wrap due to space considerations.

12.8 Start the servers

We need to start the WebSphere clones and HTTP server.

12.8.1 Start the clones

The clones can be started two ways:

1. Together: using the model
2. Individually: using the clone

12.8.1.1 Starting the clones using the model

Starting the model will start all the clones associated with the model.

Start the model by right clicking on the model name in the **Topology** tab and select **Start** as shown in Figure 310 on page 365.

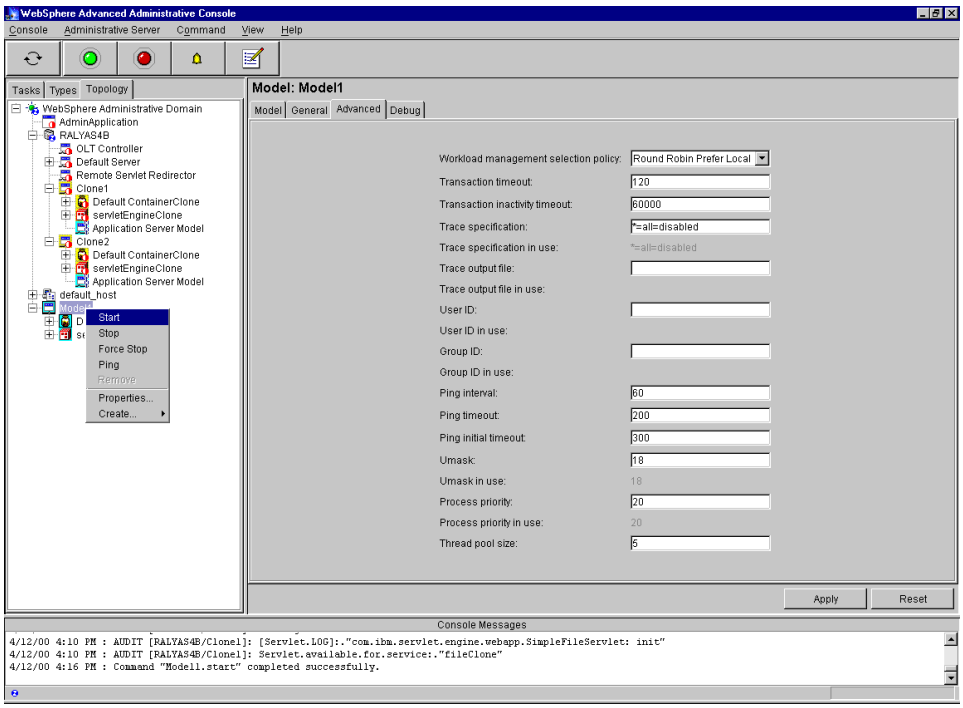

Figure 310. Start the model

When the model has started, you will see a message as shown in Figure 311.

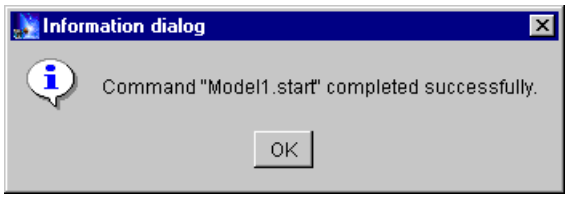

Figure 311. Model started message

Note

At the time of writing, an issue with the Administrative Console causes the clone icons not to always display as started. We have found that refreshing the node subtree will only occasionally correct the icons. Stop and restart the Administrative Console to display the correct information.

12.8.1.2 Starting the clones individually

Any clone can be started individually. This is done by selecting the clone in the **Topology** tab, right clicking on it and selecting **Start**.

12.8.2 Start the HTTP server

The HTTP server can be started using the STRTCPSVR command or by using the HTTP server Administrative Console.

```
MAIN                              AS/400 Main Menu
                                                              System:   AS4B
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. Console Access/400 tasks

    90. Sign off

Selection or command
===> STRTCPSVR SERVER(*HTTP) HTTPSVR(WASCFG)

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
```

12.9 Test the servers

Now that the HTTP server and the WebSphere clones are running, we need to test the servers to make sure that everything is configured properly. Using a Web browser, access the following URI (ralyas4b is the HTTP server and /webapp/examples/showCfg is the servlet):

```
http://ralyas4b/webapp/examples/showCfg
```

Figure 312 on page 367 shows the results of the request.

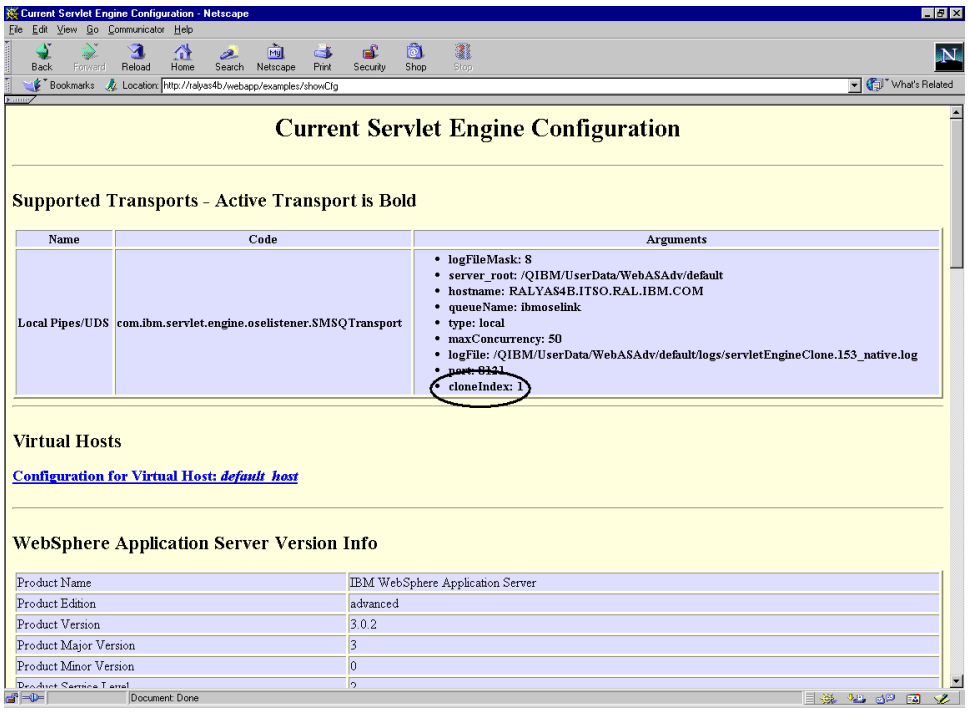

Figure 312. Results in browser 1

Notice that the request was served by Clone1 (cloneIndex=1).

A second call to the same URI will verify that we are using `Round Robin Prefer Local` for WLM:

Figure 313 on page 368 shows the results of the request.

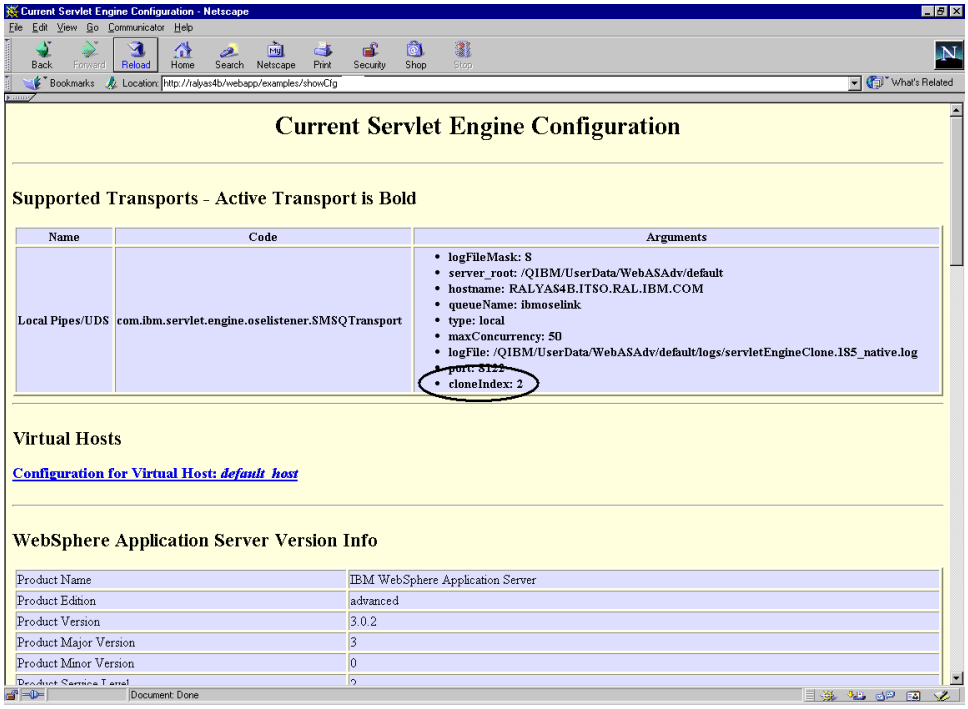

Figure 313. Results in browser 2

Notice that the request was served by Clone2 (cloneIndex=2).

12.10 Related topology

Cloning models of application servers provides a way for vertical scaling of the WebSphere environment.

12.10.1 Clones of distinct application servers

A single AS/400 will run the HTTP server, application server clones, and database. The system will contain two distinct application servers where each has a distinct Web application which has been cloned.

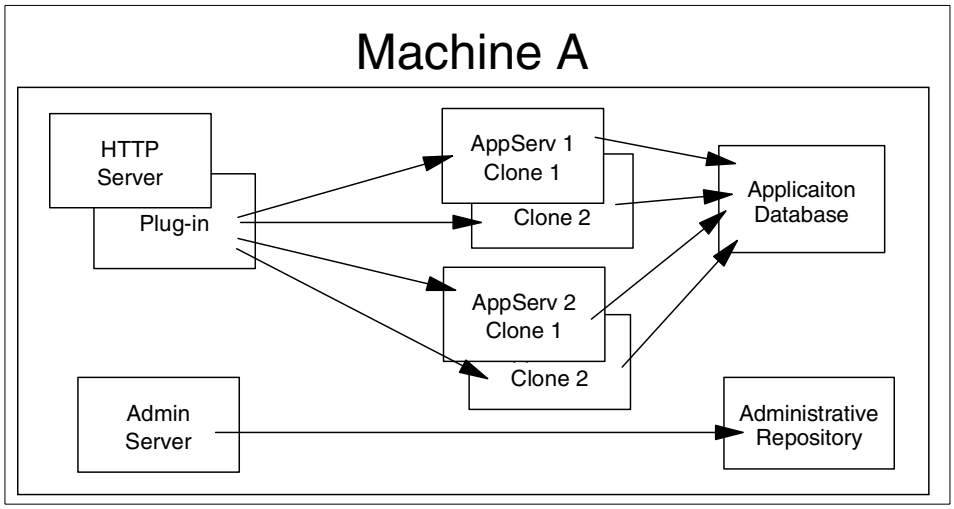

Figure 314. Cloning of multiple Web applications with a single node

This configuration can be achieved by configuring two distinct application servers as described in D.1, “Definition and configuration of an application server” on page 499 and following the cloning process discussed in this chapter for each application server.

Chapter 13. OSE Remote

The WebSphere Application Server uses OSE transport to communicate between the plug-in and the servlet engine. When running both the HTTP server and WebSphere on the same machine, OSE is run over local pipes. If the HTTP server and WebSphere are running on different machines, we can use OSE Remote.

OSE Remote allows the HTTP server to run on a distinct machine and send requests to application server(s) running on remote machines. OSE supports clustering and workload management of Application Servers. This means that the HTTP server can send requests that require intensive processing, to different jobs or machines, freeing up the HTTP server machine to process more requests. This provides both vertical and horizontal scaling of the WebSphere environment.

For highly-available topologies, a front-end load balancing system such as IBM WebSphere Edge Server (Network Dispatcher) can be used to serve requests to multiple HTTP servers. Each HTTP server can be configured to access the application servers using OSE Remote. See 16.13.2, “Variation 2: OSE Remote with high availability” on page 450.

If you want to place a firewall between the HTTP server and the application server nodes, OSE Remote uses one port for each Application Server which makes the configuration of the firewall simpler than using Servlet Redirector. In addition, OSE Remote will work with Network Address Transaction (NAT) while Servlet Redirector will not. However, OSE Remote does not support encryption of the data stream between the HTTP server and WebSphere. Servlet Redirector supports SSL encryption of the data stream.

This chapter provides instructions for the administrative tasks required to configure the plug-in for OSE Remote and remote application server machines.

We will discuss the following steps for setting up this configuration:

1. Configuration overview
2. Installation summary
3. Configure the application server
   a. Configure the host aliases
   b. Configure the transport type
4. Configure the HTTP plug-in for OSE Remote

5. Configure the HTTP server
6. Start the servers
7. Test the servers
8. Related topologies

13.1 Overview of the configuration

An HTTP server and application server are installed on two physically separate machines Machine A (RALYAS4C) and Machine B (RALYAS4B) respectively. Machine B will also be the database server. All requests received by the HTTP server on Machine A are redirected by the HTTP server plug-in, which is a component of WebSphere, to the application server in Machine B for processing. Machine B requires an Administrative Server to manage its resources.

.

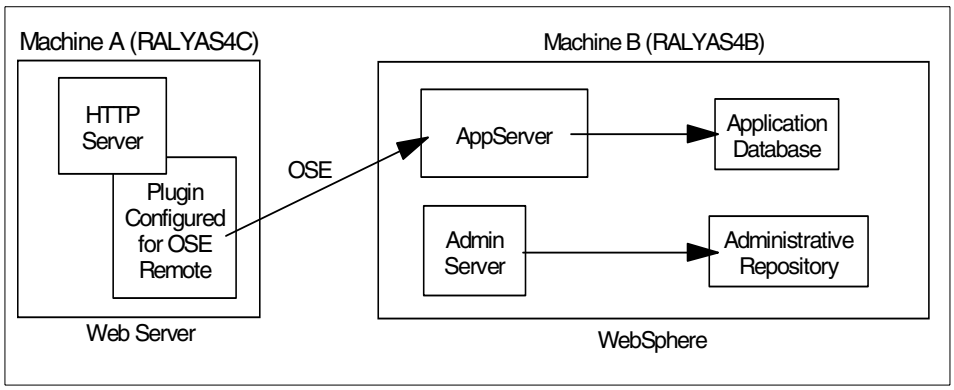

Figure 315. OSE Remote

13.2 Installation summary

Table 35 summarizes the software products and options required to support this topology.

Table 35. Product install

Machine A	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1
Machine B	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1

Please refer to the following for further information:

- *AS/400 Software Installation,* SC41-5120
- `www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html`
- *Building AS/400 Appls for WebSphere Advanced 3.0,* SG24-5691 (available as a “redpiece” at http://www.redbooks.**ibm.com**/)

13.3 Configure the application server on Machine B

Before we start configuring the plug-in for OSE Remote, we need to start the administrative tools:

1. Start the Administrative Server on Machine B.
2. Start the Administrative Console for Machine B on a workstation.

13.3.1 Adding host aliases for Machine A

The virtual host uses these aliases to accept requests redirected from the plug-in on Machine A with Machine A’s URI. You may add Machine A’s hostname, IP address, and fully qualified name.

To add the host aliases, click the **Topology** tab of the Administrative Console. Select the **default_host** item and click the **Advanced** tab.

Add aliases in the `Host Aliases` section as shown in Figure 316.

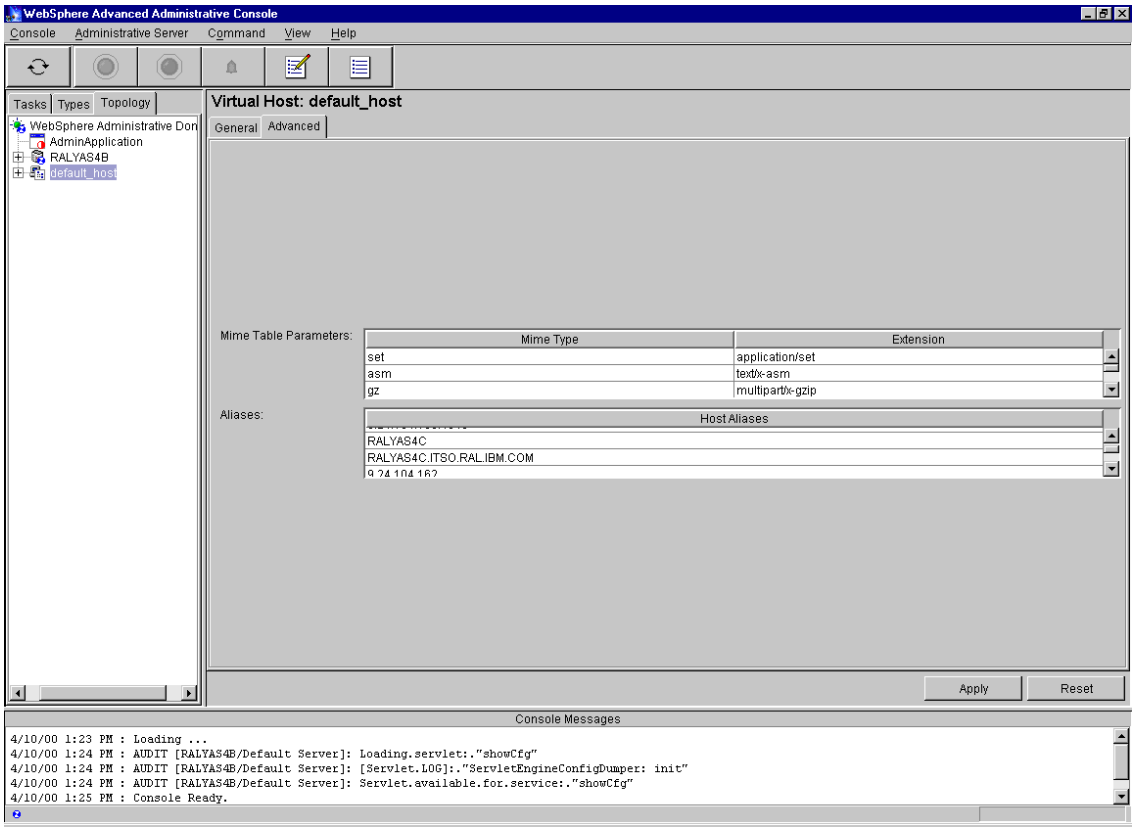

Figure 316. Add aliases

Click the **Apply** button.

When the apply completes successfully, the host aliases will have been updated.

13.3.2 Configure the transport type

The servlet engine needs to be configured to use sockets instead of local pipes.

From the **Topology** tab, select the servlet engine (`servletEngine`) of the application server (`Default Server`) under Machine B. Select the **Advanced** tab inside the Servlet Engine panel, and make sure that `OSE` is selected for the Queue Type as shown in Figure 317 on page 375.

Click the **Settings** button.

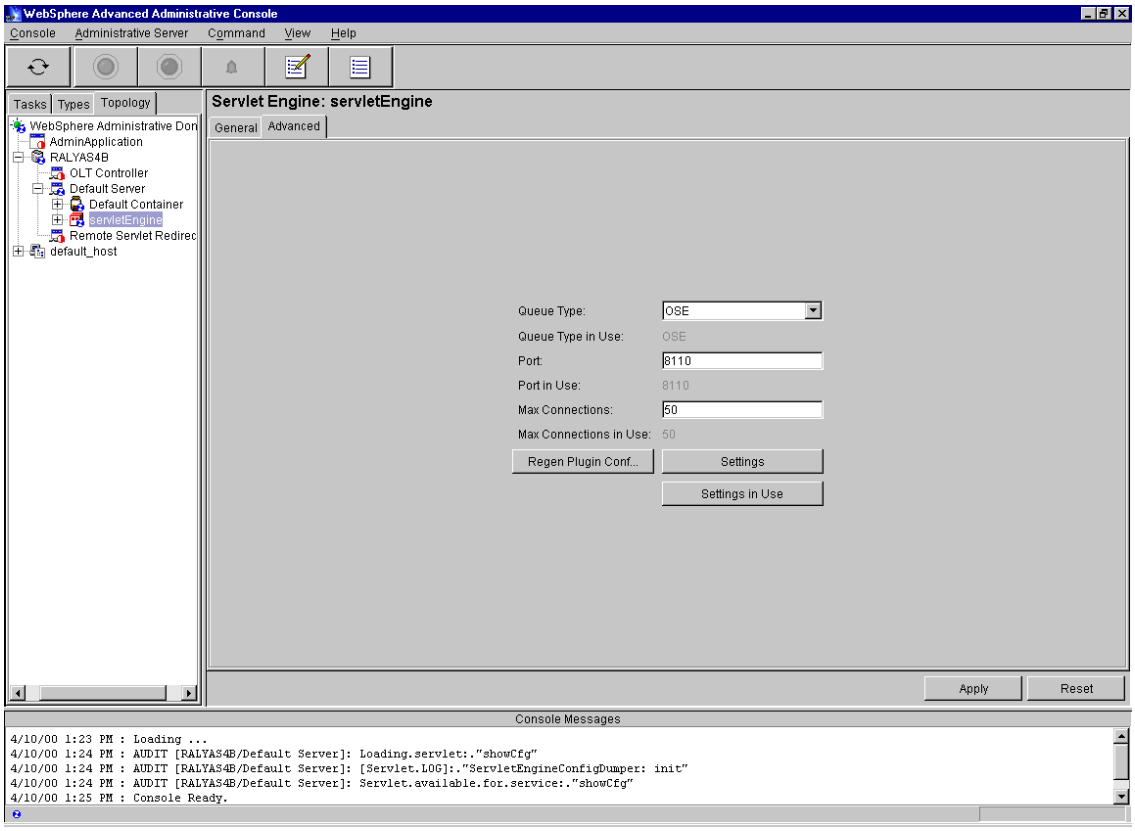

Figure 317. Servlet engine properties

The Edit Servlet Engine Transport window will appear. OSE Remote requires that we use INET Sockets for the Transport Type for the OSE queue.

Select INET Sockets for the **Transport Type** as shown in Figure 318 on page 376.

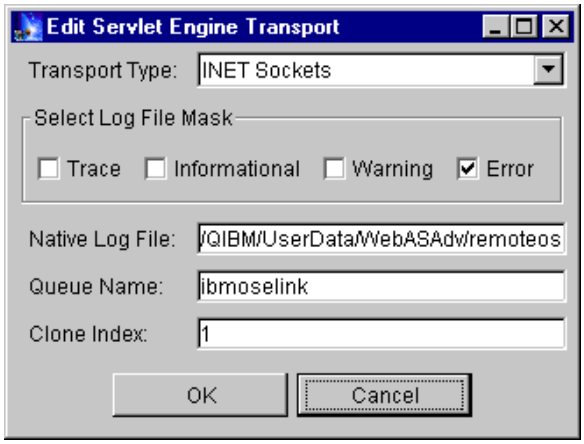

Figure 318. Transport type

Click the **OK** button.

You will be returned to the Administrative Console (Figure 317 on page 375).

Click the **Apply** button.

The servlet engine is now configured.

Note

Any changes to the application server such as adding a Web application, servlet, or EJB, should be made now. Any changes you make after you generate the plug-in files require you to re-create these files by repeating the steps in 13.4, “Configure the plug-in for OSE Remote on Machine A” on page 376.

After the plug-in files have been regenerated, the HTTP server must be restarted.

13.4 Configure the plug-in for OSE Remote on Machine A

The WebSphere plug-in for the HTTP server uses three properties files generated by WebSphere to define its configuration:

1. queues.properties
2. rules.properties

3. vhosts.properties

These files need to be generated on the AS/400 that will be used as the HTTP server.

The plug-in can be configured two ways:

1. Configure the plug-in manually

   Manual configuration requires that the files be moved (copied, FTPed and so on) from the WebSphere machine to the HTTP server machine.

   Some changes to the copied files are required.

2. Configure the plug-in using a script

   The script uses IIOP to connect to the administrative server to get the configuration information. This requires that IIOP traffic be allowed through any firewall that is between the HTTP server and WebSphere machines.

   Some changes to the generated files are required when using the script.

13.4.1 Configure the plug-in manually

1. Copy the following files from <instance_root>/temp on Machine B (WebSphere) to the same directory on Machine A (HTTP server). (See, “Note about temp directory” on page 379.)

   a. rules.properties

```
#IBM WebSphere Plugin URL Mapping Rules
#Mon Apr 10 17:09:34 GMT+00:00 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 319. Sample rules.properties

b. vhosts.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Mon Apr 10 17:09:34 GMT+00:00 2000
localhost=default_host
9.24.104.162=default_host
9.24.104.163=default_host
RALYAS4B=default_host
127.0.0.1=default_host
RALYAS4C=default_host
RALYAS4C.ITSO.RAL.IBM.COM=default_host
RALYAS4B.ITSO.RAL.IBM.COM=default_host
```

Figure 320. Sample vhosts.properties

c. queues.properties

```
#IBM WebSphere Plugin Communication Queues
#Mon Apr 10 17:09:34 GMT+00:00 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
```

Figure 321. Sample queues.properties

Note about temp directory

The <was_instance_root> does not need to be the same on both AS/400s. If this is the case, the temp files should be located in the directory specified in the `ose.tmp.dir` property in the bootstrap.properties file.

2. Edit the queues.properties file on Machine A and add an entry for the host the clone resides on. This will be the name of the AS/400 that is running the application server.

   The form of the entry is:

   ```
   ose.srvgrp.<queue>.<clone>.host=system
   ```

   In our example, the `queue` is `ibmoselink`, the `clone` is `clone1` and the `system` is `RALYAS4B` (the IP address of RALYAS4B could have been used instead). The resulting line is shown in bold in Figure 322.

```
#IBM WebSphere Plugin Communication Queues
#Mon Apr 10 15:21:04 GMT+00:00 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone1.host=RALYAS4B
```

Figure 322. Modified queues.properties

The files are now ready for use. Go to 13.5, “Configure the HTTP server” on page 382.

13.4.2 Configure the plug-in using a script

A sample shell script shown in Figure 323 on page 381 is run on Machine A. It uses the Administrative Server on Machine B to build the temporary files needed by the plug-in. These files will reside in the subdirectory /QIBM/UserData/WebASAdv/default/temp on the HTTP server (Machine A). The files are queues.properties, rules.properties, and vhosts.properties.

The script requires the `com.ibm.servlet.engine.oselistener.systemsmgmt.OSERemotePluginCfg` class. This is available in WebSphere Version 3.021.

Create the startOSEConfig script file shown in Figure 323 on page 381 in /QIBM/ProdData/WebASAdv/bin.

```
WAS_ROOT=/QIBM/ProdData/WebASAdv
INSTANCE_ROOT=/QIBM/UserData/WebASAdv/default

CP=$WAS_ROOT/lib/wsa400.jar
CP=$CP:$WAS_ROOT/lib/server/ibmwebas.jar
CP=$CP:$INSTANCE_ROOT/properties
CP=$CP:$WAS_ROOT/lib/servlet.jar
CP=$CP:$WAS_ROOT/lib/webtlsrn.jar
CP=$CP:$WAS_ROOT/lib/lotusxsl.jar
CP=$CP:$WAS_ROOT/lib/server/ns.jar
CP=$CP:$WAS_ROOT/lib/ejs.jar
CP=$CP:$WAS_ROOT/lib/ujc.jar
CP=$CP:$WAS_ROOT/lib/server/repository.jar
CP=$CP:$WAS_ROOT/lib/server/admin.jar
CP=$CP:$WAS_ROOT/lib/server/swingall.jar
CP=$CP:$WAS_ROOT/lib/console.jar
CP=$CP:$WAS_ROOT/lib/server/tasks.jar
CP=$CP:$WAS_ROOT/lib/xml4j.jar
CP=$CP:$WAS_ROOT/lib/x509v1.jar
CP=$CP:$WAS_ROOT/lib/vaprt.jar
CP=$CP:$WAS_ROOT/lib/iioprt.jar
CP=$CP:$WAS_ROOT/lib/iioptools.jar
CP=$CP:$WAS_ROOT/lib/dertrjrt.jar
CP=$CP:$WAS_ROOT/lib/sslight.jar
CP=$CP:$WAS_ROOT/lib/server/ibmjndi.jar
CP=$CP:$WAS_ROOT/lib/deployTool.jar
CP=$CP:$WAS_ROOT/lib/databeans.jar
CP=$CP:$WAS_ROOT/classes

export -s CLASSPATH=$CP

system -v "SBMJOB CMD(JAVA
CLASS(com.ibm.servlet.engine.oselistener.systemsmgmt.OSERemotePluginCfg
) PARM('-tracestring' 'com.ibm.servlet.engine.*=all=enabled'
'-tracefile' '$INSTANCE_ROOT/logs/configRegen.log' '-serverRoot'
'$INSTANCE_ROOT' '-adminNodeName' 'RALYAS4B' '-nameServicePort' '900')
CLASSPATH('$CP')) JOB($JOB_NAME) JOBD(QGPL/QDFTJOBD)
JOBQ(QEJB/QEJBJOBQ) USER(QEJBSVR)"
```

Figure 323. startOSEConfig script

Edit startOSEConfig and change the `-adminNodeName` parameter from `RALYAS4B` to the name of your WebSphere node.

Run the following command from QShell (STRQSH)

```
/QIBM/ProdData/WebASAdv/bin/startOSEConfig
```

When the script has finished running, the temporary files will have been created for the plug-in.

13.5 Configure the HTTP server

The following entries need to be added to the HTTP server config.

These entries can be added from an AS/400 sign-on using the WRKHTTPCFG command or using the HTTP server Administrative Console.

```
1. Enable POST
2. Enable GET
3. Enable HEAD
4. NameTrans /* /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:nametrans_exit
5. Authorization * /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:authorization_exit
6. Service IBMWebSphere /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:service_exit
7. ServerInit /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:init_exit
   /QIBM/UserData/WebASAdv/default/properties/bootstrap.properties
8. ServerTerm /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:term_exit
9. Pass /theme/*  /QIBM/ProdData/WebASAdv/theme/*
10.Pass /html/*  /QIBM/UserData/WebASAdv/default/html/*
11.Pass /WebSphereSamples/*  /QIBM/ProdData/WebASAdv/WebSphereSamples/*
```

Note

Each entry listed above should be on a separate line. Any single entry should be on one line with no word wrap. The lines are shown here with word wrap due to space considerations.

13.6 Start the servers

1. The application server can be started by clicking **Default Server** in the Topology tab and selecting **Start**.
2. The HTTP server can be started using the STRTCPSVR command or by using the HTTP server Administrative Console.

13.7 Test the servers

Now that the HTTP server and WebSphere are running, we need to test the servers to make sure that everything is configured properly. Using a Web browser, access the following URI (ralyas4c is the HTTP server and /webapp/examples/showCfg is the servlet):

```
http://ralyas4c/webapp/examples/showCfg
```

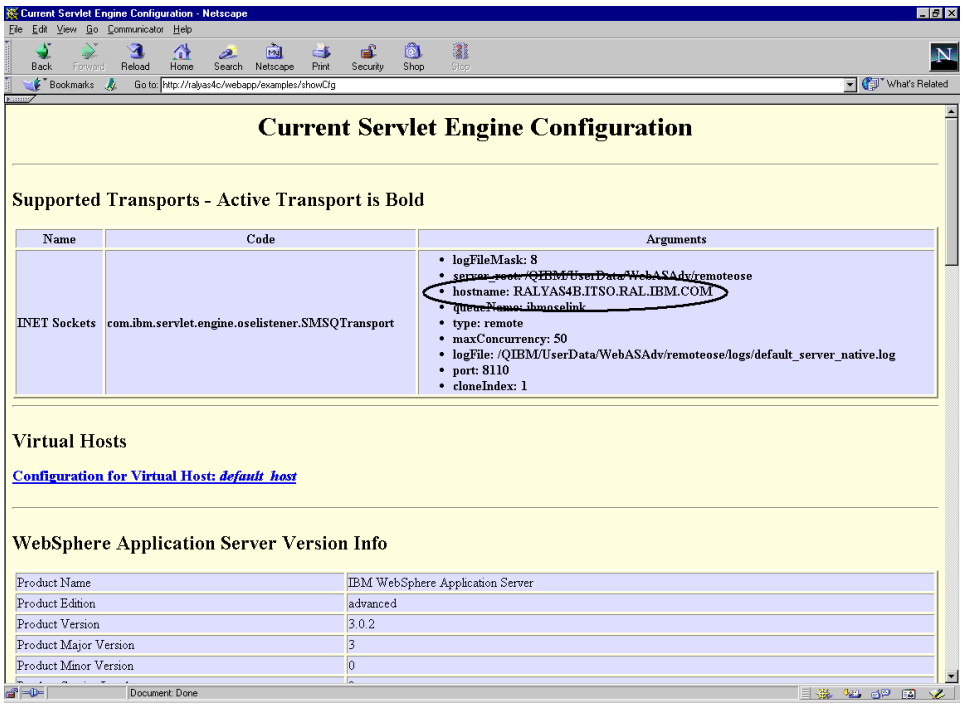

Figure 324. Results in browser

Notice that the hostname (RALYAS4B.ITSO.RAL.IBM.COM) is the name of the AS/400 running WebSphere and not the one that is the HTTP server (RALYAS4C).

13.8 Related topologies

OSE Remote can be used for both vertical and horizontal scaling of the WebSphere environment.

13.8.1 Variation 1: horizontal scaling

An AS/400 acting as the HTTP server (Machine A) will use OSE Remote to route requests between two nodes (Machine B and Machine C). Each node will be running a clone of the application server. The nodes will share a repository stored on Machine A.

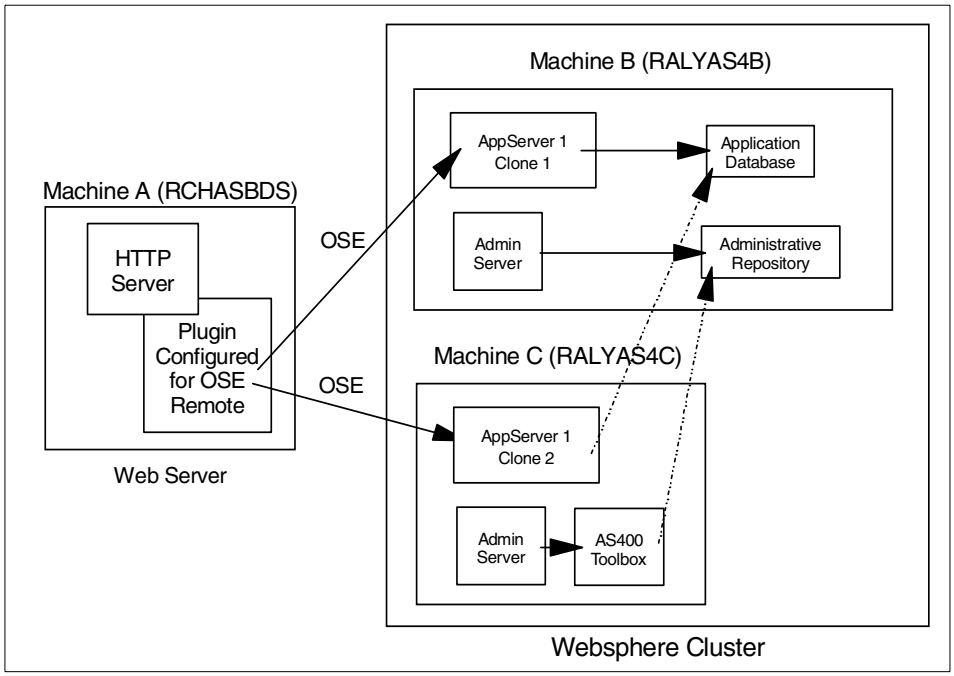

Figure 325. Remote OSE with cloned servers

This configuration can be achieved by creating the WebSphere cluster of Machine A and Machine B as described in 11.10, “Creating a cluster” on page 331 and then cloning the server as described in Chapter 12, “Application server clones” on page 355.

After configuring the cluster, generate the plug-in files on Machine A using the method described in 13.4, “Configure the plug-in for OSE Remote on Machine A” on page 376.

If you use the procedure described in 13.4.2, “Configure the plug-in using a script” on page 380, not all nodes will be listed in the queues.properties files after the files have been generated. These will need to added manually. For each node that is not listed, add the following properties:

```
1. ose.srvgrp.<queue>.<clone>.host
```

2. ose.srvgrp.<queue>.<clone>.port

   The port the clone uses can be found by expanding the clone in the **Topology** tab, selecting the servlet engine, and then the **Advanced** tab.

3. ose.srvgrp.<queue>.<clone>.type

The files you create by using script or by hand should look similar to the ones shown below.

```
#IBM WebSphere Plugin Communication Queues
#Thu Apr 27 12:23:41 CDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.host=RALYAS4C.ITSO.RAL.IBM.COM
ose.srvgrp.ibmoselink.clone1.port=8140
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone2.host=RALYAS4B.ITSO.RAL.IBM.COM
ose.srvgrp.ibmoselink.clone2.port=8141
ose.srvgrp.ibmoselink.clone2.type=remote
```

Figure 326. queues.properties on the HTTP server

Note

You need to ensure that the queues.properties file on the HTTP server points to all the available servlet engines:

- Make sure all the clones are represented.
- Make sure that the type of the clones is set to remote.
- Make sure that the appropriate port number is set for each clone.
- Make sure that the host entry is set for each clone.
- Make sure that the clones count is equal to the total number of clones.

```
#IBM WebSphere Plugin URL Mapping Rules
#Thu Apr 27 12:23:41 CDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 327. rules.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu Apr 27 12:23:41 CDT 2000
9.24.104.163=default_host
9.24.104.162=default_host
RALYAS4C=default_host
RALYAS4B=default_host
RALYAS4C.ITSO.RAL.IBM.COM=default_host
RALYAS4B.ITSO.RAL.IBM.COM=default_host
localhost=default_host
127.0.0.1=default_host
RCHASBDS=default_host
RCHASBDS.RCHLAND.IBM.COM=default_host
9.5.186.35=default_host
```

Figure 328. vhosts.properties

13.8.2 Variation 2: distinct Web applications

An AS/400 acting as the HTTP server (Machine A) will use OSE Remote to route requests between two nodes (Machine B and Machine C). Each node will be running a distinct application server. The nodes will share a repository stored on Machine B.

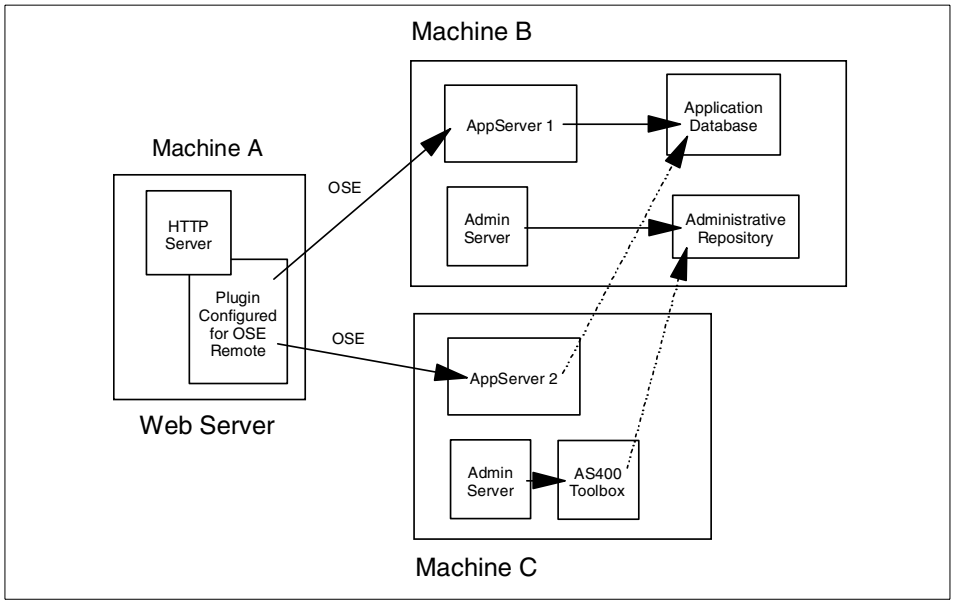

Figure 329. OSE Remote with distinct application servers

This configuration can be achieved by creating the WebSphere cluster of Machine A and Machine B as described in 11.10, “Creating a cluster” on page 331 and configuring an application server on each node.

After configuring the cluster and application servers, generate the plug-in files on Machine A using the method described in 13.4.2, “Configure the plug-in using a script” on page 380.

The files you create should look similar to the ones shown below.

```
#IBM WebSphere Plugin Communication Queues
#Fri Apr 28 12:01:06 GMT+00:00 2000
ose.srvgrp.ibmoselink.clonescount=1
ose.srvgrp.queue1.clone1.host=RALYAS4C.ITSO.RAL.IBM.COM
ose.srvgrp.queue1.clone1.port=8110
ose.srvgrp.queue1.clone1.type=local
ose.srvgrp.queue1.type=FASTLINK
ose.srvgrp=ibmoselink,queue1
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.host=RALYAS4B.ITSO.RAL.IBM.COM
ose.srvgrp.ibmoselink.clone1.port=8110
ose.srvgrp.ibmoselink.clone1.type=local
ose.srvgrp.queue1.clonescount=1
```

Figure 330. queues.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu Apr 27 12:23:41 CDT 2000
9.24.104.163=default_host
9.24.104.162=default_host
RALYAS4C=default_host
RALYAS4B=default_host
RALYAS4C.ITSO.RAL.IBM.COM=default_host
RALYAS4B.ITSO.RAL.IBM.COM=default_host
localhost=default_host
127.0.0.1=default_host
RCHASBDS=default_host
RCHASBDS.RCHLAND.IBM.COM=default_host
9.5.186.35=default_host
```

Figure 331. vhosts.properties

```
#IBM WebSphere Plugin URL Mapping Rules
#Fri Apr 28 12:01:06 GMT+00:00 2000
default_host/webapp/examples/HitCount=queue1
default_host/webapp/AS4BServlet=ibmoselink
default_host/webapp/examples/ErrorServlet=queue1
default_host/webapp/AS4CApp/AS4CServlet=queue1
default_host/webapp/examples/simpleJSP.servlet=queue1
default_host/servlet=queue1
default_host/ErrorReporter=queue1
default_host/admin/install=queue1
default_host/webapp/examples/showCfg=queue1
default_host/webapp/examples/*.jsp=queue1
default_host/*.jsp=queue1
default_host/webapp/examples/verify=queue1
default_host/webapp/examples/ping=queue1
default_host/servlet/snoop2=queue1
default_host/servlet/snoop=queue1
default_host/admin/*.jsp=queue1
default_host/webapp/examples/SourceCodeViewer=queue1
default_host/webapp/examples/=queue1
default_host/webapp/examples=queue1
default_host/admin=queue1
default_host/admin/servlet=queue1
default_host/servlet/hello=queue1
default_host/admin/=queue1
default_host/admin/ErrorReporter=queue1
default_host/webapp/examples/simpleJSP=queue1
```

Figure 332. rules.properties

Note

When using OSE Remote to distribute requests between two distinct application servers, each URI will be served by only one queue. All requests to a URI will go to the queue specified in the <instance_root>/temp/rules.properties file.

For example, both of our application servers have a URI for /webapp/examples/showCfg. Notice that in the rules.properties file there is only one listing for this URI and it is associated with the queue `queue1`. All requests for this URI will be routed to `queue1`.

13.8.3 Variation 3: horizontal scaling

An AS/400 acting as the HTTP server (Machine A) will use OSE Remote to route requests between two nodes (Machine A and Machine B). Each node will be running a clone of the application server. The nodes will share a repository stored on Machine A.

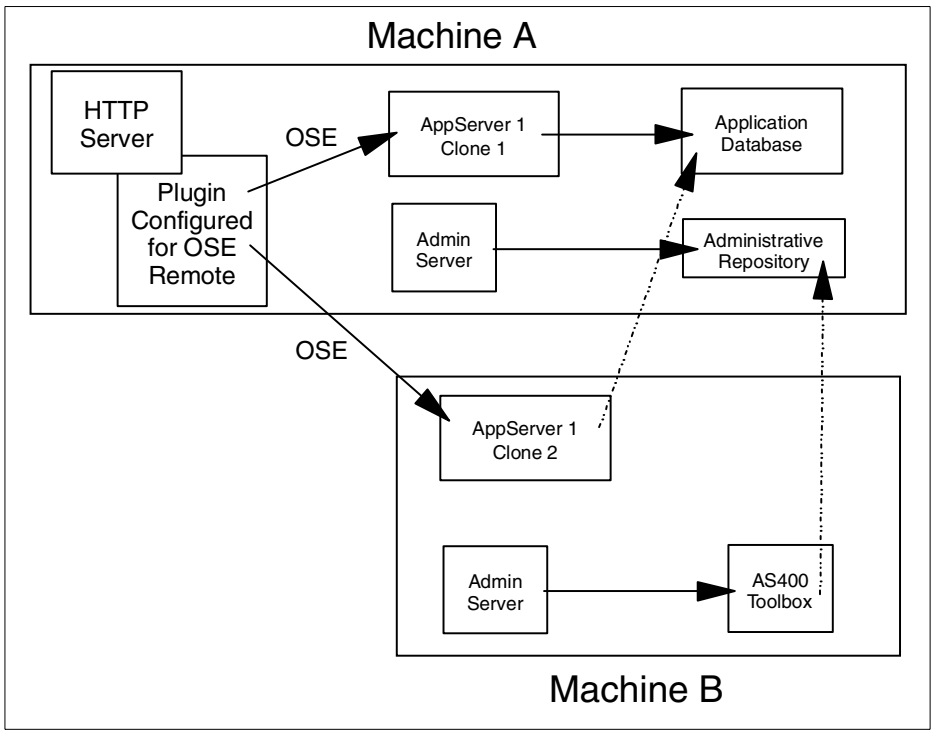

Figure 333. OSE Remote with local and remote application servers

This configuration can be achieved by creating the WebSphere cluster of Machine A and Machine B as described in 11.10, “Creating a cluster” on page 331 and then cloning the server as described in Chapter 12, “Application server clones” on page 355.

After configuring the cluster, generate the plug-in files on Machine A using the method described in 13.4.2, “Configure the plug-in using a script” on page 380.

The files you create should look similar to the ones shown below.

```
#IBM WebSphere Plugin Communication Queues
#Thu Apr 27 12:23:41 CDT 2000
ose.srvgrp.ibmoselink.clonescount=2
ose.srvgrp=ibmoselink
ose.srvgrp.ibmoselink.type=FASTLINK
ose.srvgrp.ibmoselink.clone1.host=RALYAS4C.ITSO.RAL.IBM.COM
ose.srvgrp.ibmoselink.clone1.port=8140
ose.srvgrp.ibmoselink.clone1.type=remote
ose.srvgrp.ibmoselink.clone2.host=RALYAS4B.ITSO.RAL.IBM.COM
ose.srvgrp.ibmoselink.clone2.port=8141
ose.srvgrp.ibmoselink.clone2.type=remote
```

Figure 334. queues.properties

```
#IBM WebSphere Plugin URL Mapping Rules
#Thu Apr 27 12:23:41 CDT 2000
default_host/webapp/examples/HitCount=ibmoselink
default_host/webapp/examples/ErrorServlet=ibmoselink
default_host/webapp/examples/simpleJSP.servlet=ibmoselink
default_host/servlet=ibmoselink
default_host/ErrorReporter=ibmoselink
default_host/admin/install=ibmoselink
default_host/webapp/examples/showCfg=ibmoselink
default_host/webapp/examples/*.jsp=ibmoselink
default_host/*.jsp=ibmoselink
default_host/webapp/examples/verify=ibmoselink
default_host/webapp/examples/ping=ibmoselink
default_host/servlet/snoop2=ibmoselink
default_host/servlet/snoop=ibmoselink
default_host/admin/*.jsp=ibmoselink
default_host/webapp/examples/SourceCodeViewer=ibmoselink
default_host/webapp/examples/=ibmoselink
default_host/webapp/examples=ibmoselink
default_host/admin=ibmoselink
default_host/admin/servlet=ibmoselink
default_host/servlet/hello=ibmoselink
default_host/admin/=ibmoselink
default_host/admin/ErrorReporter=ibmoselink
default_host/webapp/examples/simpleJSP=ibmoselink
```

Figure 335. rules.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu Apr 27 12:23:41 CDT 2000
9.24.104.163=default_host
9.24.104.162=default_host
RALYAS4C=default_host
RALYAS4B=default_host
RALYAS4C.ITSO.RAL.IBM.COM=default_host
RALYAS4B.ITSO.RAL.IBM.COM=default_host
localhost=default_host
127.0.0.1=default_host
```

Figure 336. vhosts.properties

Chapter 14. Thick Servlet Redirector

With the Servlet Redirector, each WebSphere Administrative Server connects directly to the WebSphere Administrative Repository. This requires that either the database be installed locally or that the appropriate database drivers be installed and configured on the machine. More importantly from a security standpoint a database user ID and password must be stored on the machine for use by the database processes. And also the Administrative Server on the HTTP server machine requires a TCP connection to the remote database. Therefore you must open a port in the firewall to pass DB traffic. In topologies where security is a concern, such as a server running outside a firewall, this may not be acceptable.

This chapter provides instructions for the administrative tasks required to configure the Servlet Redirector and remote application server machines.

We will discuss the following steps for setting up this configuration.

1. Configuration overview
2. Installation summary
3. Configuring the WebSphere Application Server environments
   a. The full Administrative Server for servlets and EJBs
   b. The full Administrative Server for the thick Redirector
4. Changing the application server transport type
5. Configuring the Servlet Redirector
6. Updating the host server information
7. Configure the HTTP server
8. Testing the servers
9. Related topologies

14.1 Overview of the configuration

A Web server and a full WebSphere Administrative Server are installed on Machine A (RALYAS4C) to provide the thick Servlet Redirector support. All requests are redirected or forwarded to the WebSphere Administrative Server running the application server on Machine B (RALYAS4B). Machine B has only the WebSphere default application, which is typically installed when you first start your WebSphere Administrative Server.

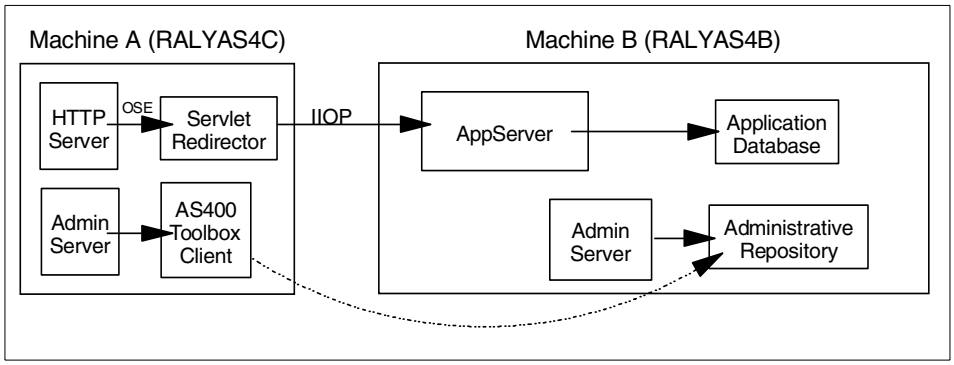

Figure 337. Thick Servlet Redirector

Within our scenario we have decided that all servlets and EJBs will be run only on Machine B. This is a typical scenario where Machine B would be behind a firewall in a back office environment.

14.2 Installation summary

Table 36 summarizes the software products and options required to support this topology.

Table 36. Product Install

Machine A	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and option 2
	5769JC1 AS/400 Toolbox for Java 5769JC1
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and option 1
Machine B	OS/400 V4R4 or above, options 12 and 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and option 2
	5733WA2 / 5733WA3 WebSphere *BASE and option 1

Please refer to the following for further information:

- *AS/400 Software Installation,* SC41-5120

- http://www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html
- *Building AS/400 Appls for WebSphere Advanced 3.0,* SG24-5691 (available as a “redpiece” at http://www.redbooks.**ibm.com**/)

14.3 Configuring the WebSphere Administrative Servers

We used two additional instances for our WebSphere Administrative Server for this scenario. If you run as an additional instance then an additional configuration step is required in 14.6, “Configuring the Servlet Redirector for machine A” on page 398. Our chosen configuration and instance options are as follows.

14.3.1 Machine B: WebSphere Admistrative Server

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 37.

Table 37. Properties for RALYAS4B WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebASAdv/ThickSR
mntr.admin.name	TSRADMIN
install.initial.config	true
admin.dbSchema	ejsscn14
admin.bootstrapPort	7714
admin.lsdPort	9914
ose.srvgrp.ibmappserve.clone1.port	8814

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports.

Additionally as we intend to connect to the administrative respository collection from another full Administrative Server we needed to undertake the additional setup described in 11.10.4, “Configuration changes for the remote AS/400” on page 332.

14.3.2 Machine A: thick Servlet Redirector

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 38.

Table 38. Properties for RALYAS4C WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebASAdv/RemoteThickSR
mntr.admin.name	TSRADMIN
install.initial.config	false
admin.dbSchema	ejsscn14
admin.bootstrapPort	7714
admin.lsdPort	9914
ose.srvgrp.ibmappserve.clone1.port	8814

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports.

As we have multiple instances of the HTTP server and we are not binding these to a specific IP address, we need to choose a distinct TCP/IP port for our HTTP server instance.

As we intend to connect to a remote administrative respository collection, we needed to undertake the additional setup describe in 11.10.5, “Configuration changes on additional Admin Server” on page 333. We chose to synchronize the passwords for both QEJB profiles.

Additionally, we will not be running any servlets or EJB’s on this machine, so we changed our WebSphere Administrative Server instance to disable the creation of the default application server, as described in 11.6, “Configuring Admin Server instance to not install the default application server” on page 324.

14.4 Starting your WebSphere Application Servers

14.4.1 Machine B: WebSphere Administrative Server

Start the WebSphere Administrative Server on Machine B. If this is the default WebSphere Administrative Server, start the QEJBSBS subsystem in the QEJB library. If this an additional WebSphere Administrative Server instance, refer to 11.5.3, “Starting the new instance and connecting an Admin Console”

on page 322 for further information. When the server has started, we start an Administrative Console on a workstation.

14.4.2 Machine A: thick Servlet Redirector

Start the WebSphere Administrative Server on Machine A. If this is the default WebSphere Administrative Server, start the QEJBSBS subsystem in the QEJB library. If this an additional WebSphere Administrative Server instance, refer to 11.5.3, “Starting the new instance and connecting an Admin Console” on page 322 for further information.

14.5 Changing the transport type on Machine B’s servlet engine

If you will only access the servlets on Machine B via a Servlet Redirector then you can optionally set the queue type to NONE for your servlet engine on the **Advanced** tab of the servlet engine properties of the Administrative Console as shown in Figure 338.

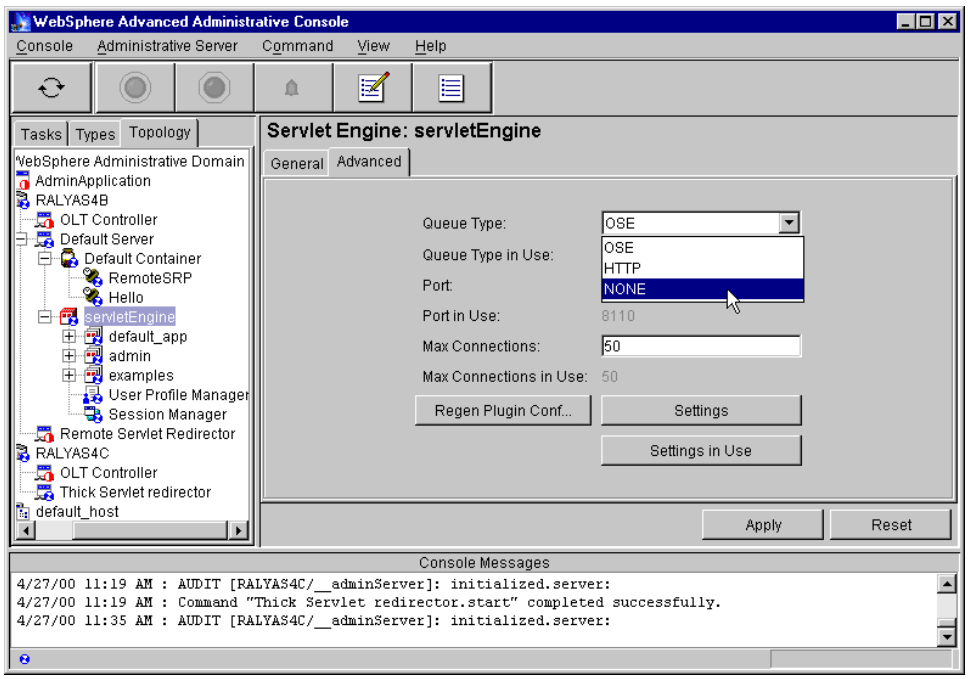

Figure 338. Changing the servlet engine transport type

After selecting NONE, click the **Apply** button. The OSE transport mechanism has been changed. When you start or next start your WebSphere

Administrative Server, the servlet engine will not start an OSE listener process, which can improve your servlet engine performance.

Attention

Typically, in an Internet environment, where Machine A is outside your firewall, you may have an HTTP server running on Machine B connecting to the WebSphere instance to allow you to ensure that your environment is working as expected. If you set the queue type to NONE for your servlet engine you will not be able to do this as the default transport mechanism for an HTTP server instance and WebSphere Application Server instance is OSE.

14.6 Configuring the Servlet Redirector for machine A

14.6.1 Creating the Servlet Redirector

As we did not install the default application server on Machine A (RALYAS4C) we need to create a Servlet Redirector.

Note

If you decide to install the initial configuration, then you will see a Remote Servlet Redirector for Machine A in your Topology tab. This is created automatically when you start your Administrative Server. In this case, skip down several paragraphs to information on Edit Servlet Redirector Transport.

There are two methods for bringing up the Create Servlet Redirector dialog box. The first method is from the Type pane as shown in Figure 339 on page 399.

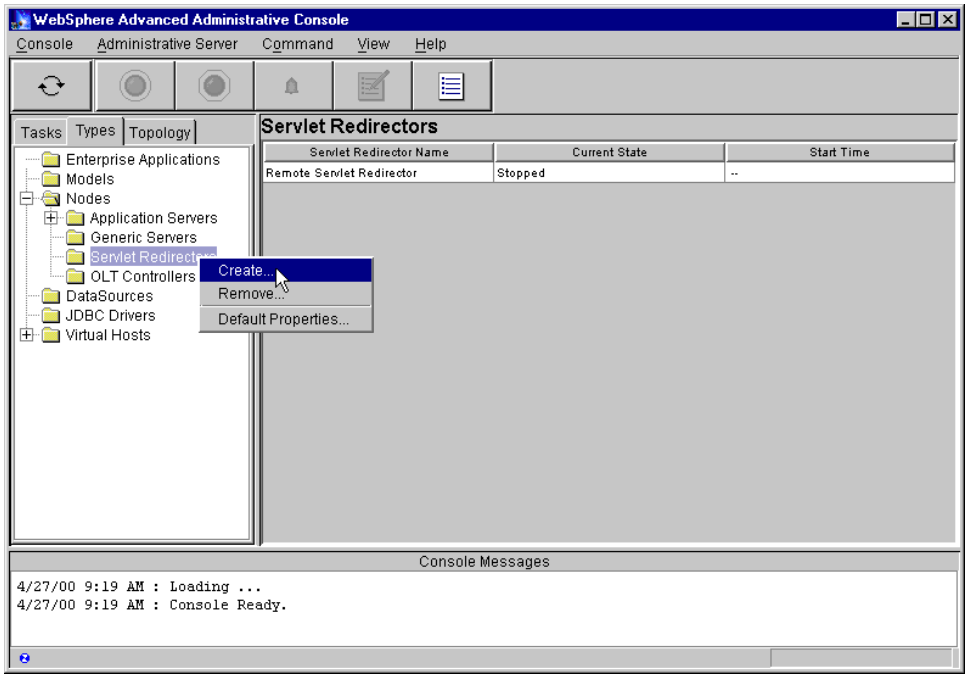

Figure 339. Creating a Servlet Redirector from the Types pane

Expand the Nodes subtree and right clicking **Servlet Redirectors**. In the resulting context menu click **Create**.

Alternatively, from the Topology pane right click the desired node, in our case RALYAS4C. In the resulting context menu, select **Create** and then **Servlet Redirector** as shown in Figure 340 on page 400.

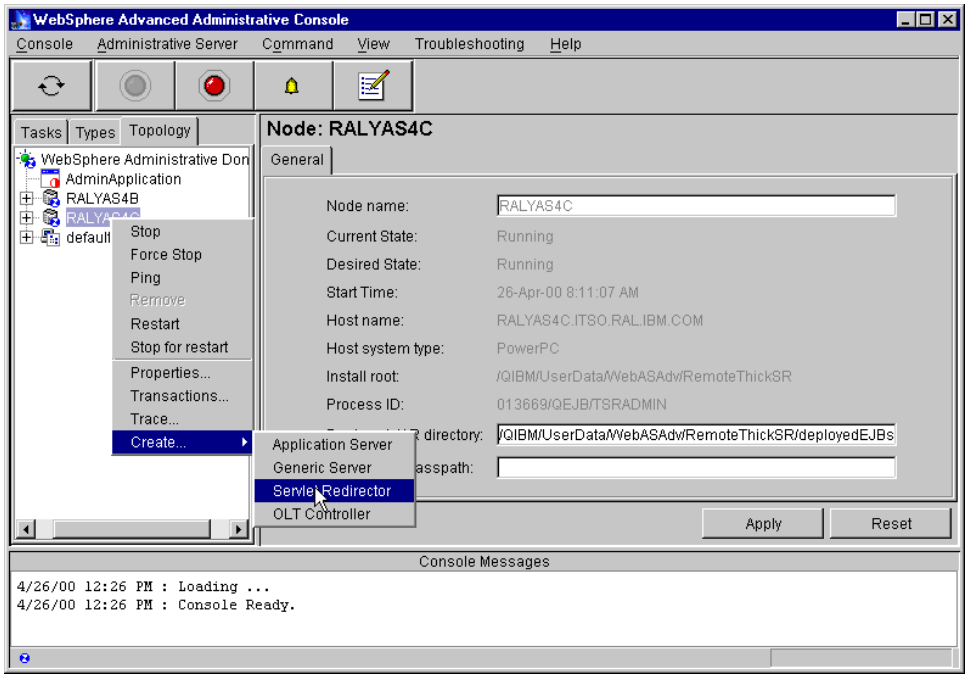

Figure 340. Creating a Servlet Redirector from the Topology pane

This causes the Create Servlet Redirector dialog to be displayed. In the Create Servlet Redirector dialog, choose the node name for Machine A (RALYAS4C) from the drop down list. In our example, we use “Thick Servlet redirector” as the name of our Servlet Redirector as shown in Figure 341 on page 401.

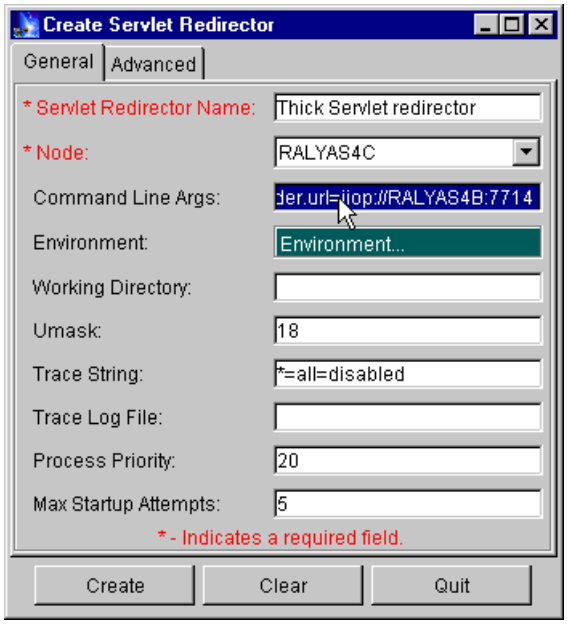

Figure 341. Creating a Servlet Redirector - part 1

As our example uses an additional WebSphere Application Server instance we must specify an additional command line argument to define where the Servlet Redirector should connect to the name service daemon.

Command Line argument for Servlet Redirector
`-Djava.naming.provider.url=iiop://RALYAS4C:7714`

This argument is not required if you are using the default WebSphere Administrative Server instance.

Click the **Advanced** tab to display the additional configuration options. When you create a Servlet Redirector the default standard output and error files are created in the administrative instance root directory with the names stdout.txt and stderr.txt respectively. We recommend that you change these to first have the log files generated in the <was_instance_root>/logs directory and utilize names that indicate that these are your Servlet Redirector files.

option	New value
Stdout	<was_insatnce_root>/logs/redirector_stdout.log
Stderr	<was_instance_root>/logs/redirector_stderr.log

These name changes are as shown in Figure 342.

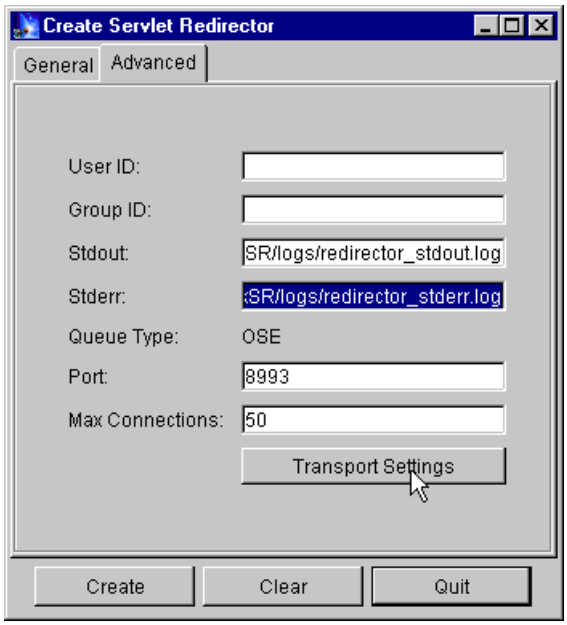

Figure 342. Creating a Servlet Redirector - part 2

We recommend that you also change the values for the Servlet Redirector transport queue name and native log file. Click **Transport Settings** to display the Edit Servlet Redirector Transport dialog. By default, “ibmoselink” is specified in the Queue Name field. You should modify this to a unique name to identify the Servlet Redirector. The HTTP plug-in uses this queue name when it forwards requests to the Servlet Redirector. We also changed and qualified the Native Log File value to ensure it uniqueness. Our values were:

option	New value
Native Log File	<was_instance_root>/logs/redirectorA_native.log

option	New value
Queue Name	redirectorA

These changes are as shown in Figure 343.

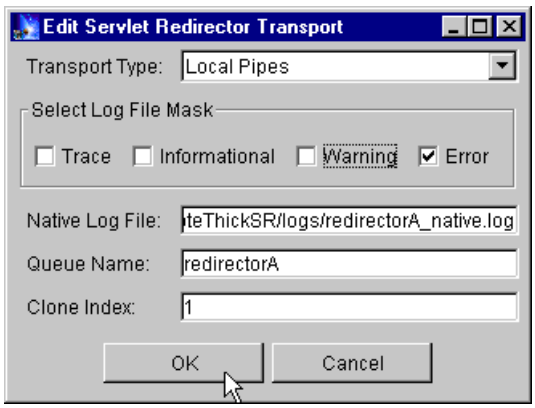

Figure 343. Editing the Servlet Redirector Transport details

Click the **OK** button on the Edit Servlet Redirector Transport dialog and then click the **Create** button on the Create Servlet Redirector dialog to generate your Servlet Redirector.

Note

If you are intending to support servlets/EJBs in the WebSphere Administrative Server with the thick Servlet Redirector, you must change the Servlet Redirector transport queue name from the default for the Servlet Redirector.

14.6.2 Enabling the Servlet Redirector

Finally, you must enable your Servlet Redirector, by right clicking the Servlet Redirector and selecting **Enabled** from the pop-up context menu as shown in Figure 344 on page 404.

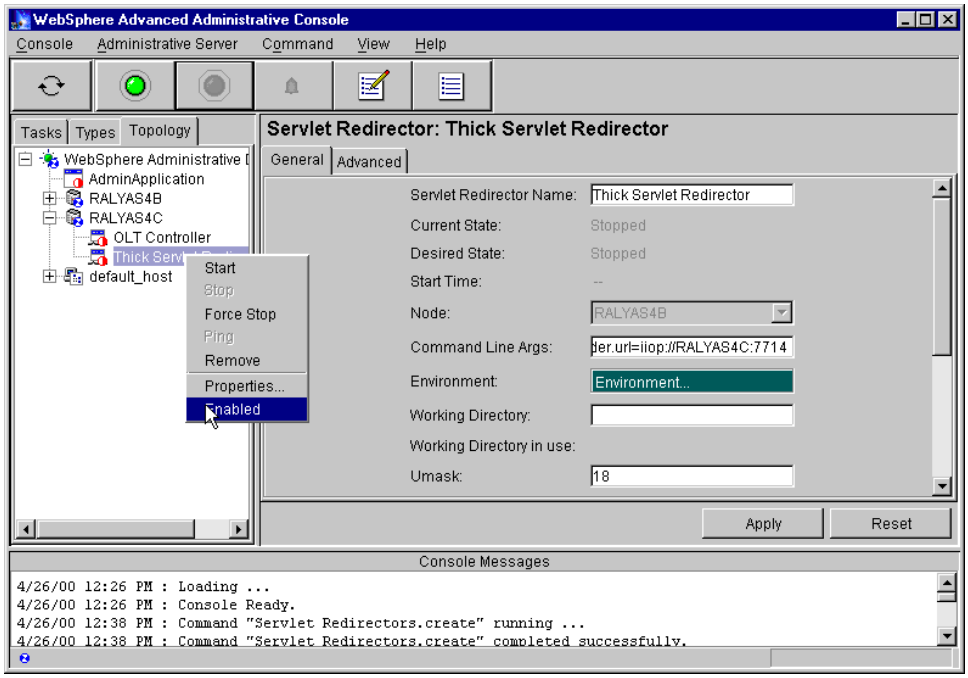

Figure 344. Enabling the Servlet Redirector

Redisplaying the Servlet Redirector pop-up context menu, you should now see a tick next to the Enabled option to indicate that the Servlet Redirector has been enabled.

14.7 Updating the host alias information

Using the Administrative Console, configure the host aliases for the Servlet Redirector machine. The fully qualified host name information should be added to the host aliases for the virtual hosts.

Note

Within the AS/400 version of WebSphere Advanced V3, you must specify a fully qualified host name within the host aliases. This is different for the Windows and AIX environments where you need only specify the host name.

Select **default_host** in the Topology tab and click the **Advanced** tab. Enter the host name, fully qualified host name and IP address for Machine A as shown in Figure 345.

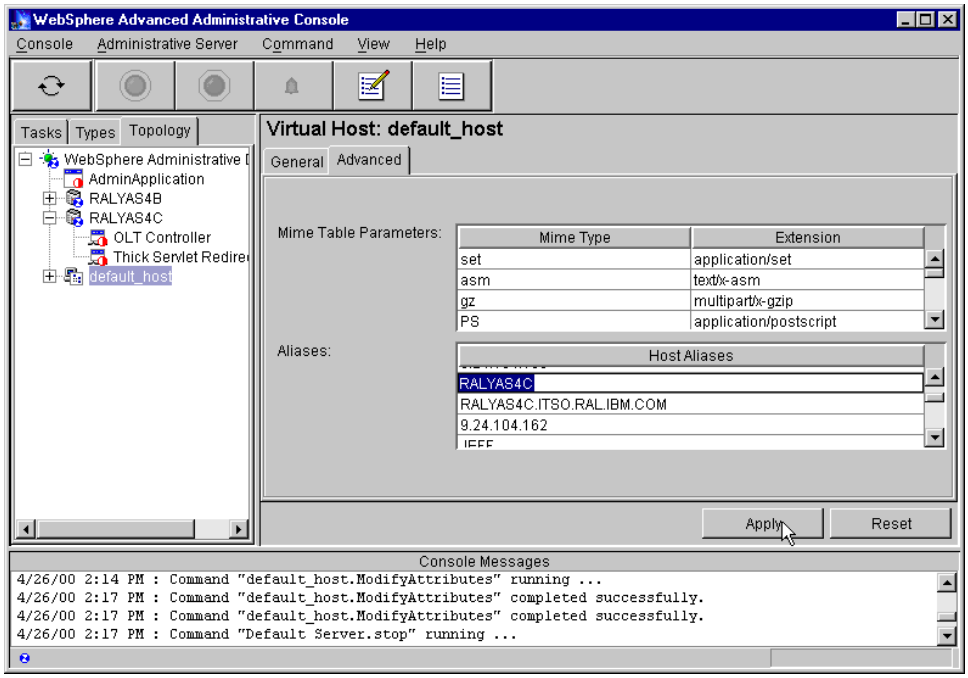

Figure 345. Adding the additional host alias information

Then click the **Apply** button.

The host aliases have been updated.

14.8 Start the servers

Now, we are ready to start the servers.

14.8.1 Start the application server

Start the application server, in our example the Default Server on node RALYAS4B. If the application server is already started, stop and restart it to ensure that the host alias changes are activated.

14.8.2 Start the Servlet Redirector

Start the Servlet Redirector on node RALYAS4C. On the AS/400 the Servlet Redirector will appear as a separate job in the subsystem, with a job name of the first 10 characters of the name you specified when the Servlet Redirector was created. In our example the job name would be 'THICK_SERV' as shown in Figure 346.

```
Work with Active Jobs                       AS4C
                                                          04/27/00  15:46:08
CPU %:    1.0     Elapsed time:   01:21:52     Active jobs:   155
Opt  Subsystem/Job  User        Type  CPU %  Function        Status
     QEJBSBS        QSYS        SBS      .0                  DEQW
       THICK_SERV   QEJB        BCI      .0                  JVAW
       TSRADMIN     QEJB        BCI      .3                  JVAW
       TSRMNTR      QEJB        BCH      .0  PGM-QEJBMNTR    EVTW




                                                                        Bottom
===>
F21=Display instructions/keys
```

Figure 346. WRKATJOB showing the thick Servlet Redirector job

Unfortunately, there is no message appended to the job log of the Servlet Redirector job when it is fully ready. Wait until the CPU utilization percentage drops down to zero and the job status is JVAW.

14.8.3 Verify the plug-in properties files

Verify that the plug-in properties files: rules.properties, queue.properties, and vhost.properties, in the WebSphere instance /temp directory on Machine A (RALYAS4C) were generated correctly.

> **Note**
>
> We found that regeneration time of the plug-in property files varied greatly between a few seconds and several hours depending on the individual topology.

These files define which requests are being provided by the Servlet Redirector to Machine B.

14.8.3.1 Aliases in the vhost properties

This file contains the TCP/IP host names and IP address mapping onto the virtual hosts within the WebSphere domain. WebSphere will substitute the virtual host information for the Host name or IP address of the HTTP request as specified in this file. This is then validated against the rules.properties file.

The vhost.properties file should contain the host alias information for Machine B (RALYAS4A) and the information we entered for Machine A (RALYAS4C). Figure 347 shows our vhost.properties file.

```
#IBM WebSphere Plugin Virtual Host Mappings
#Thu Apr 27 11:25:11 GMT+00:00 2000
9.24.104.163=default_host
9.24.104.162=default_host
RALYAS4C=default_host
RALYAS4B=default_host
JEFF=default_host
RALYAS4C.ITSO.RAL.IBM.COM=default_host
RALYAS4B.ITSO.RAL.IBM.COM=default_host
localhost=default_host
JEFF.OTHERDOMAIN.COM=default_host
127.0.0.1=default_host
```

Figure 347. <was_instance_root>/temp/vhosts.properties file on Machine A (RALYAS4C)

If the host alias information for Machine A (RALYAS4C) cannot be found within the vhosts.properties file, then your requests will not be processed.

14.8.3.2 Queue name in the rules.properties file

The rules.properties file contains information on which servlet request is processed via which WebSphere transport mechanism. The modified URL is parsed against this file looking for a full or partial match. If one is found then the HTTP server plug-in checks the queues.properties file.

In our scenario we only forward requests via the Servlet Redirector to Machine B, so all requests are handled via the redirectorA queue.

```
#IBM WebSphere Plugin URL Mapping Rules
#Wed Apr 26 19:10:13 GMT+00:00 2000
default_host/webapp/examples/HitCount=redirectorA
default_host/webapp/examples/ErrorServlet=redirectorA
default_host/webapp/examples/simpleJSP.servlet=redirectorA
default_host/servlet=redirectorA
default_host/ErrorReporter=redirectorA
default_host/admin/install=redirectorA
default_host/webapp/examples/showCfg=redirectorA
default_host/webapp/examples/*.jsp=redirectorA
default_host/*.jsp=redirectorA
default_host/webapp/examples/verify=redirectorA
default_host/webapp/examples/ping=redirectorA
default_host/servlet/snoop2=redirectorA
default_host/servlet/snoop=redirectorA
default_host/admin/*.jsp=redirectorA
default_host/webapp/examples/SourceCodeViewer=redirectorA
default_host/webapp/examples/=redirectorA
default_host/webapp/examples=redirectorA
default_host/admin=redirectorA
default_host/admin/servlet=redirectorA
default_host/servlet/hello=redirectorA
default_host/admin/=redirectorA
default_host/admin/ErrorReporter=redirectorA
default_host/webapp/examples/simpleJSP=redirectorA
```

Figure 348. <was_instance_root>/temp/rules.properties file on Machine A

If the queue name specified in the servlet redirect transport dialog is not found in this file, then it is likely that the plug-in refresh has not occurred.

14.8.3.3 Queue properties in the queues.properties file

The queue.properties file contains information on the queue transport definitions. In our scenario this defines the transport mechanism between the HTTP plug-in and the thick Servlet Redirector.

```
#IBM WebSphere Plugin Communication Queues
#Wed Apr 26 19:10:13 GMT+00:00 2000
ose.srvgrp.redirectorA.clonescount=1
ose.srvgrp=redirectorA
ose.srvgrp.redirectorA.type=FASTLINK
ose.srvgrp.redirectorA.clone1.port=8993
ose.srvgrp.redirectorA.clone1.type=local
```

Figure 349. <was_instance_root>/temp/queues.properties file on Machine A

14.8.4 Start the Web server on Machine A

The HTTP server on Machine A (RALYAS4C) can be started using the STRTCPSVR command or by using the HTTP server Administrative Console.

14.8.5 Confirm the scenario is working

Now that the HTTP server, thick Servlet Redirector, and WebSphere are running, we need to test the servers to make sure that everything is configured properly.

Start up an instance of your preferred browser, in our examples we use Netscape, and enter the following URI (RALYAS4C is the HTTP server) and press Enter:

```
http://RALYAS4C/webapp/AS1default/showCfg
```

You should see an HTML page similar to Figure 350 on page 410. Notice the host and queue names. This indicates that the servlet request was redirected to the WebSphere Application Server on Machine B (RALYAS4B).

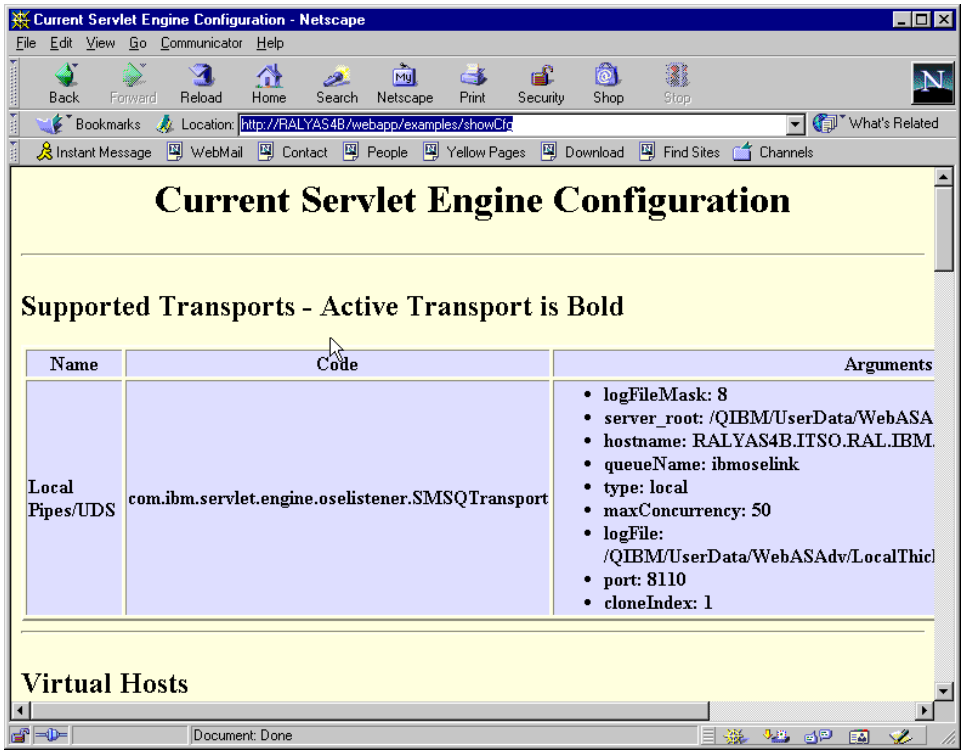

Figure 350. HTML page returned by HTTP request

Chapter 15. Standalone (thin) Servlet Redirector

Normally an Administrative Server will reside on the same machine as the HTTP server. This requires that the user ID and password for the database repository also be stored on that machine. In addition, the HTTP plug-in normally uses the OSE transport to direct requests to an application server. OSE does not support encryption. In topologies where security is a concern, using OSE Remote may not be acceptable.

The standalone Servlet Redirector allows us to send the requests over IIOP (which supports encryption) without having direct access to the Administrative Server. The Redirector uses pre-generated files to route the requests to the application server. Hence, if the application server is changed in any way, such as adding a Web application, servlet, or EJB, the pre-generated files must be regenerated. The HTTP server and Redirector will deal with the changes automatically without restarting.

This chapter describes how to install and configure a standalone Servlet Redirector based on eXtensible Markup Language (XML).

We will discuss the following steps for setting up this configuration:

1. Installation summary
2. Configure the application server
3. Configure the Servlet Redirector
4. Generate the HTTP server plug-in configuration files
5. Configure the HTTP server
6. Start the servers
7. Test the servers
8. Related topologies

15.1 Overview of the configuration

An HTTP server and a standalone Servlet Redirector will be installed on Machine A (RALYAS4C). An application server will run on Machine B (RALYAS4B). Requests will be redirected from Machine A to the application server running on Machine B.

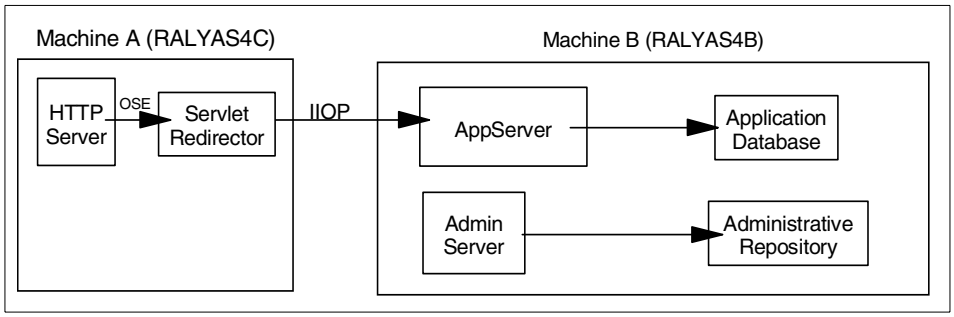

Figure 351. Standalone Servlet Redirector

15.2 Installation summary

Table 39 summarizes the software products and options required to support this topology.

Table 39. Product install

Machine A	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1
Machine B	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1

Please refer to the following for further information:

- *AS/400 Software Installation,* SC41-5120
- `www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html`
- *Building AS/400 Appls for WebSphere Advanced 3.0,* SG24-5691 (available as a “redpiece” at http://www.redbooks.**ibm.com**/)

15.3 Configure the application server

Before you start configuring the thin Servlet Redirector, you need to start the administrative tools and add host aliases for the HTTP server:

1. Start the Administrative Server on Machine B.
2. Start the Administrative Console for Machine B on a workstation.
3. Add host aliases for Machine A (HTTP server).

Using the Administrative Console, configure the host aliases for the WebSphere. The host information such as host name, IP address and fully qualified name for the HTTP server needs to be added to the table.

Select **default_host** in the Topology tab and click the **Advanced** tab. Enter the host name and IP address for the HTTP server machine as shown in Figure 352.

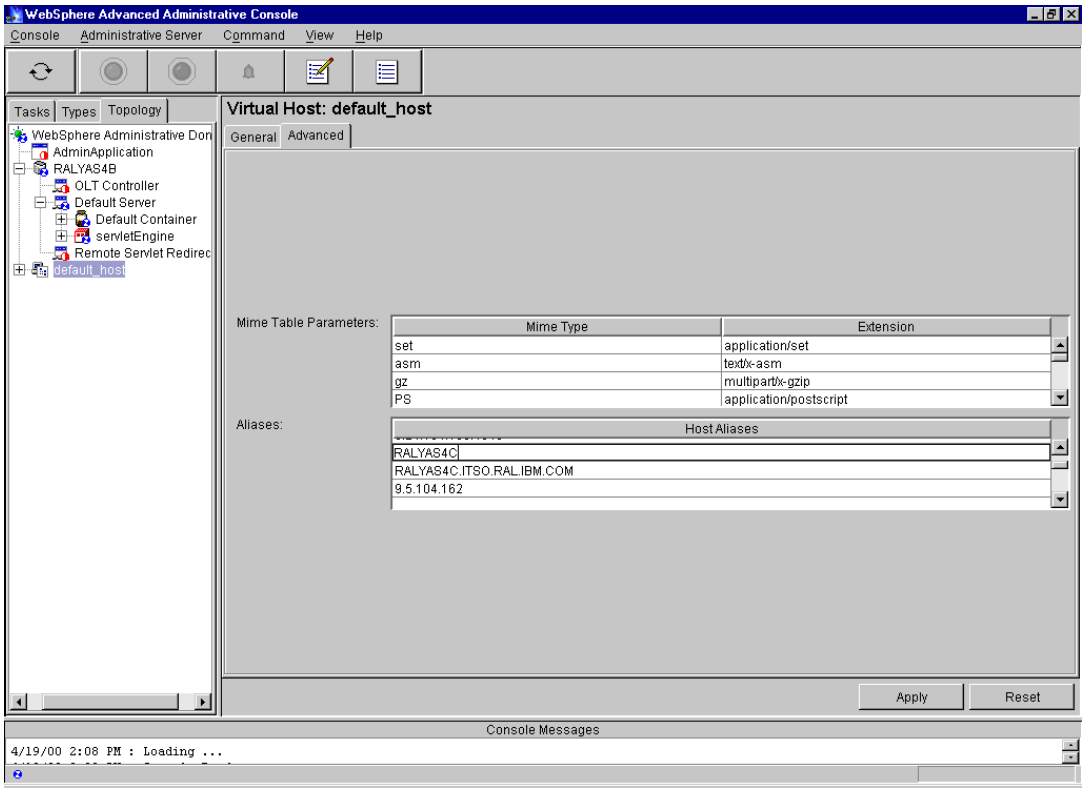

Figure 352. Host aliases

Then click the **Apply** button.

The host aliases have been updated.

> **Note**
>
> Any changes to the application server such as adding a Web application, servlet, or EJB, should be made now. Any changes we make after we generate the plug-in files require us to re-create these files by repeating the steps in 15.5, “Generate the HTTP server plug-in configuration” on page 416.

15.4 Configure the Servlet Redirector

The <instance_root>/properties/iiopredirector.xml file (see, “Note about <instance_root>” on page 416) is used to provide a configuration for starting the Servlet Redirector and for generating HTTP server plug-in configuration files. It contains the Servlet Redirector transport information as well as Administrative Server information.

For a description of the entries in the file, refer to H.1, “Standalone Servlet Redirector iiopredirector.xml file parameters” on page 549.

We do not need to modify any of the transport entries.

We need to modify the Administrative Server information in the iiopredirector.xml file on the HTTP server (RALYAS4C) to point to the WebSphere Application Server (RALYAS4B). We do this by changing the following entries in the XML file:

1. admin-node-name

   The original line is:

   ```
   <admin-node-name>localhost</admin-node-name>
   ```

   In our case we need to change it to:

   ```
   <admin-node-name>RALYAS4B</admin-node-name>
   ```

2. name-service-node-name

   This is generally the same as admin-node-name.

   The original line is:

   ```
   <name-service-node-name>localhost</name-service-node-name>
   ```

   In our case we need to change it to:

```
<name-service-node-name>RALYAS4B</name-service-node-name>
```

3. name-service-port

   If you are using a port other than 900 (the default) for the application server you will need to change this value.

The resulting file will look similar to Figure 353.

```
<?xml version="1.0"?>
<iiop-redirector>
   <active-transport>ose</active-transport>
   <transport>
      <name>ose</name>
      <code>com.ibm.servlet.engine.oselistener.SMSQTransport</code>
      <arg name="port" value="8110"></arg>
      <arg name="queueName" value="queue1"></arg>
      <arg name="maxConcurrency" value="50"></arg>
      <arg name="type" value="local"></arg>
      <arg name="server_root" value="$server_root$"></arg>
      <arg name="cloneIndex" value="1"></arg>
   </transport>
   <admin-node-name>RALYAS4B</admin-node-name>
   <qualify-home-names>true</qualify-home-names>
   <name-service-node-name>RALYAS4B</name-service-node-name>
   <name-service-port>900</name-service-port>
</iiop-redirector>
```

Figure 353. iiopredirector.xml

Note about <instance_root>

<instance_root> specifies the fully qualified path to the WebSphere instance root.

The default instance is located at /QIBM/UserData/WebASAdv/default. If you created a new instance of WebSphere using the `crtnewinst` script, the <instance_root> is the directory you passed in as the parameter to the script.

For example, if you used the command:

```
crtnewinst /QIBM/UserData/WebASAdv/myinstance
```

to create the instance the <instance_root> would be /QIBM/UserData/WebASAdv/myinstance.

15.5 Generate the HTTP server plug-in configuration

On the machine containing the HTTP server and standalone Redirector, run the Standalone Plug-in Configuration program to generate a properties file defining the configuration of the HTTP server plug-in.

15.5.1 Generate the plug-in files

WebSphere includes a script to generate the HTTP server plug-in files. The script /QIBM/ProdData/WebASAdv/bin/redirectorconfig (shown in Figure 354 on page 417) defines the necessary environment variables and queries the Administrative Server for the plug-in information.

```
WAS_ROOT=/QIBM/ProdData/WebASAdv
JOB_NAME=$1
INSTANCE_ROOT=$2
ADMIN_NODE=$3
NS_NODE=$4
NS_PORT=$5
TRACE_STRING=$6
CP=$WAS_ROOT/lib/wsa400.jar
CP=$CP:$WAS_ROOT/lib/server/ibmwebas.jar
CP=$CP:$INSTANCE_ROOT/properties
CP=$CP:$WAS_ROOT/lib/servlet.jar
CP=$CP:$WAS_ROOT/lib/webtlsrn.jar
CP=$CP:$WAS_ROOT/lib/lotusxsl.jar
CP=$CP:$WAS_ROOT/lib/server/ns.jar
CP=$CP:$WAS_ROOT/lib/ejs.jar
CP=$CP:$WAS_ROOT/lib/ujc.jar
CP=$CP:$WAS_ROOT/lib/server/repository.jar
CP=$CP:$WAS_ROOT/lib/server/admin.jar
CP=$CP:$WAS_ROOT/lib/server/swingall.jar
CP=$CP:$WAS_ROOT/lib/console.jar
CP=$CP:$WAS_ROOT/lib/server/tasks.jar
CP=$CP:$WAS_ROOT/lib/xml4j.jar
CP=$CP:$WAS_ROOT/lib/x509v1.jar
CP=$CP:$WAS_ROOT/lib/vaprt.jar
CP=$CP:$WAS_ROOT/lib/iioprt.jar
CP=$CP:$WAS_ROOT/lib/iioptools.jar
CP=$CP:$WAS_ROOT/lib/dertrjrt.jar
CP=$CP:$WAS_ROOT/lib/sslight.jar
CP=$CP:$WAS_ROOT/lib/server/ibmjndi.jar
CP=$CP:$WAS_ROOT/lib/deployTool.jar
CP=$CP:$WAS_ROOT/lib/databeans.jar
CP=$CP:$WAS_ROOT/classes
if test -n "$TRACE_STRING"
then
   TRACE_STRING_PARM="'-traceString' '$TRACE_STRING'"
else
   TRACE_STRING_PARM=
fi
system -v "SBMJOB CMD(JAVA
CLASS(com.ibm.servlet.engine.oselistener.systemsmgmt.StandalonePluginCf
g) PARM('-serverRoot' '$INSTANCE_ROOT' '-adminNodeName' $ADMIN_NODE
'-nameServiceNodeName' $NS_NODE '-nameServicePort' $NS_PORT
'-queueProps' '$INSTANCE_ROOT/properties/iiopredirector.xml'
$TRACE_STRING_PARM) CLASSPATH('$CP')) JOB($JOB_NAME)
JOBD(QGPL/QDFTJOBD) JOBQ(QEJB/QEJBJOBQ) USER(QEJBSVR)"
```

Figure 354. redirectorconfig script

The script should be called in QShell (STRQSH).

The command syntax is:

```
/qibm/proddata/webasadv/bin/redirectorconfig <job name> <instance root>
<admin node> <name service node> <name service port> [trace string]
```

The command we use to start the tool is:

```
/qibm/proddata/webasadv/bin/redirectorconfig qcfgjob
/qibm/userdata/webasadv/default RALYAS4B RALYAS4B 900
```

The job will run under the QEJBSBS subsystem. When the job has ended, the files should be in the <instance_root>/temp directory on the HTTP server.

15.5.2 Verify the plug-in properties files

Verify that the plug-in properties files: rules.properties, queue.properties, and vhosts.properties, in the <instance_root>/temp directory on the HTTP server machine (Machine A) were generated correctly. The files should look similar to the ones shown below.

The values in the queues.properties file should be the same as the values specified in the iiopredirector.xml file.

```
#IBM WebSphere Plugin Communication Queues
#Wed Apr 26 10:32:58 GMT+00:00 2000
ose.srvgrp.queue1.clone1.port=8110
ose.srvgrp.queue1.clone1.type=local
ose.srvgrp.queue1.type=FASTLINK
ose.srvgrp=queue1
ose.srvgrp.queue1.clonescount=1
```

Figure 355. queues.properties

```
#IBM WebSphere Plugin URL Mapping Rules
#Wed Apr 26 10:32:58 GMT+00:00 2000
default_host/webapp/examples/HitCount=queue1
default_host/webapp/examples/ErrorServlet=queue1
default_host/webapp/examples/simpleJSP.servlet=queue1
default_host/servlet=queue1
default_host/ErrorReporter=queue1
default_host/admin/install=queue1
default_host/webapp/examples/showCfg=queue1
default_host/webapp/examples/*.jsp=queue1
default_host/*.jsp=queue1
default_host/webapp/examples/verify=queue1
default_host/webapp/examples/ping=queue1
default_host/servlet/snoop2=queue1
default_host/servlet/snoop=queue1
default_host/admin/*.jsp=queue1
default_host/webapp/examples/SourceCodeViewer=queue1
default_host/webapp/examples/=queue1
default_host/webapp/examples=queue1
default_host/admin=queue1
default_host/admin/servlet=queue1
default_host/servlet/hello=queue1
default_host/admin/=queue1
default_host/admin/ErrorReporter=queue1
default_host/webapp/examples/simpleJSP=queue1
```

Figure 356. rules.properties

```
#IBM WebSphere Plugin Virtual Host Mappings
#Wed Apr 26 10:32:58 GMT+00:00 2000
9.24.104.163=default_host
9.24.104.162=default_host
RALYAS4C=default_host
RALYAS4B=default_host
RALYAS4B.ITSO.RAL.IBM.COM=default_host
localhost=default_host
127.0.0.1=default_host
```

Figure 357. vhosts.properties

Note

The files in the <instance_root>/temp directory on the HTTP server machine (Machine A) will not be the same as the files on the application server machine (Machine B).

15.6 Configure the HTTP server

The following entries need to be added to the HTTP server configuration.

These entries can be added from an AS/400 sign-on using the `WRKHTTPCFG` command or using the HTTP server Administrative Console.

```
1. Enable POST
2. Enable GET
3. Enable HEAD
4. NameTrans /* /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:nametrans_exit
5. Authorization * /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:authorization_exit
6. Service IBMWebSphere /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:service_exit
7. ServerInit /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:init_exit
   /QIBM/UserData/WebASAdv/default/properties/bootstrap.properties
8. ServerTerm /QSYS.LIB/QEJB.LIB/QSVTGO46PI.SRVPGM:term_exit
9. Pass /theme/*  /QIBM/ProdData/WebASAdv/theme/*
10.Pass /html/*  /QIBM/UserData/WebASAdv/default/html/*
11.Pass /WebSphereSamples/*  /QIBM/ProdData/WebASAdv/WebSphereSamples/*
```

Note

Each entry listed above should be on a separate line. Any single entry should be on one line with no word wrap. The lines are shown here with word wrap due to space considerations.

15.7 Start the servers

The application server must be running before we start the Redirector. The Redirector obtains the port number to send the requests to from the application server when the Redirector is started. If the application server is not running when the Redirector is started, the Redirector cannot obtain the

port number. Also, if the application server is restarted for any reason, the Redirector must be restarted as well.

15.7.1 Start the application server on Machine B

Start the application server by clicking **Default Server** and selecting **Start** as shown in Figure 358.

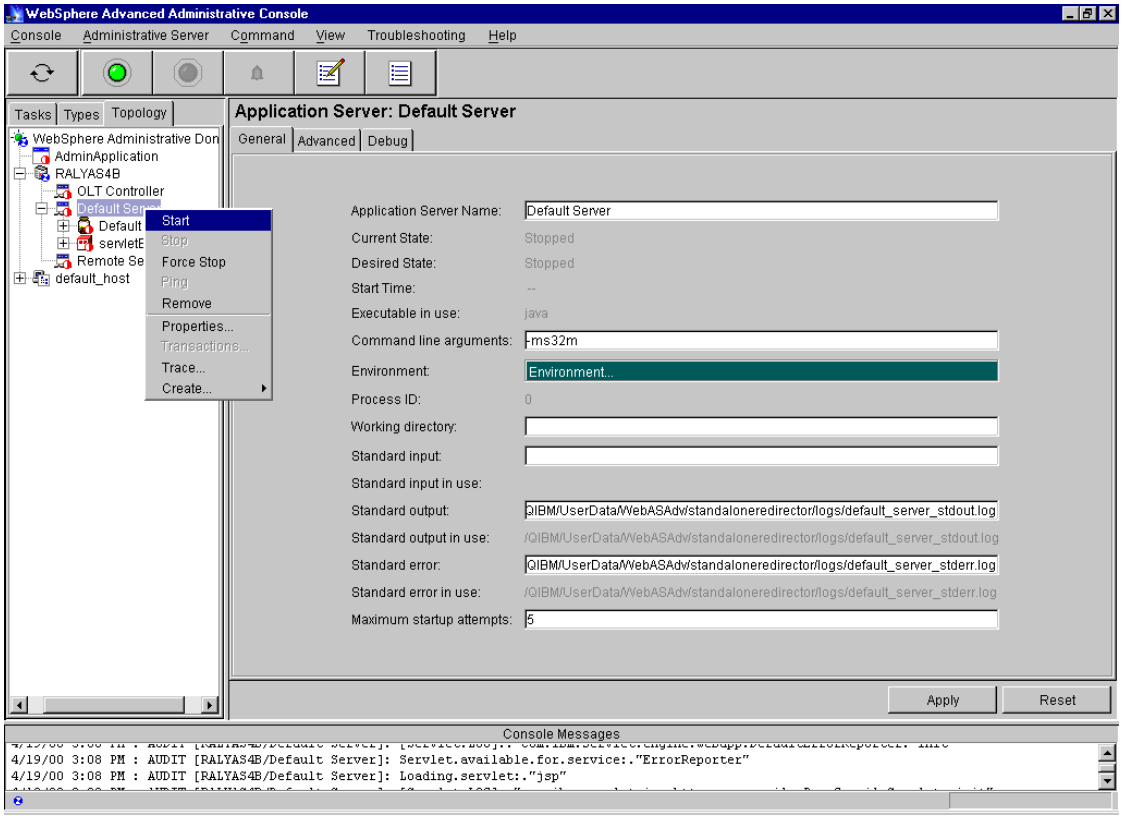

Figure 358. Start application server

15.7.2 Start the HTTP server on Machine A

The HTTP server can be started using the `STRTCPSVR` command or by using the HTTP server Administrative Console.

15.7.3 Start the standalone Redirector on Machine A

WebSphere includes a script to start the standalone Redirector. The script /QIBM/ProdData/WebASAdv/bin/redirector (shown in Figure 359 on page

423) defines the necessary environment variables and starts the redirector job.

```
WAS_ROOT=/QIBM/ProdData/WebASAdv
JOB_NAME=$1
INSTANCE_ROOT=$2
TRACE_STRING=$3
CP=$WAS_ROOT/lib/wsa400.jar
CP=$CP:$WAS_ROOT/lib/server/ibmwebas.jar
CP=$CP:$INSTANCE_ROOT/properties
CP=$CP:$WAS_ROOT/lib/servlet.jar
CP=$CP:$WAS_ROOT/lib/webtlsrn.jar
CP=$CP:$WAS_ROOT/lib/lotusxsl.jar
CP=$CP:$WAS_ROOT/lib/server/ns.jar
CP=$CP:$WAS_ROOT/lib/ejs.jar
CP=$CP:$WAS_ROOT/lib/ujc.jar
CP=$CP:$WAS_ROOT/lib/server/repository.jar
CP=$CP:$WAS_ROOT/lib/server/admin.jar
CP=$CP:$WAS_ROOT/lib/server/swingall.jar
CP=$CP:$WAS_ROOT/lib/console.jar
CP=$CP:$WAS_ROOT/lib/server/tasks.jar
CP=$CP:$WAS_ROOT/lib/xml4j.jar
CP=$CP:$WAS_ROOT/lib/x509v1.jar
CP=$CP:$WAS_ROOT/lib/vaprt.jar
CP=$CP:$WAS_ROOT/lib/iioprt.jar
CP=$CP:$WAS_ROOT/lib/iioptools.jar
CP=$CP:$WAS_ROOT/lib/dertrjrt.jar
CP=$CP:$WAS_ROOT/lib/sslight.jar
CP=$CP:$WAS_ROOT/lib/server/ibmjndi.jar
CP=$CP:$WAS_ROOT/lib/deployTool.jar
CP=$CP:$WAS_ROOT/lib/databeans.jar
CP=$CP:$WAS_ROOT/classes
if test -n "$TRACE_STRING"
then
   TRACE_STRING_PARM="'-traceString' '$TRACE_STRING'"
else
   TRACE_STRING_PARM=
fi

system -v "SBMJOB CMD(JAVA
CLASS(com.ibm.servlet.engine.ejs.IIOPRedirector)
PARM($TRACE_STRING_PARM) PROP((server.root '$INSTANCE_ROOT'))
CLASSPATH('$CP')) JOB($JOB_NAME) JOBD(QGPL/QDFTJOBD)
JOBQ(QEJB/QEJBJOBQ) USER(QEJBSVR)"
```

Figure 359. Redirector script

The command syntax is:

```
/qibm/proddata/webasadv/bin/redirector <job name> <instance root> [trace
string]
```

The command we use to start the tool is:

```
/QIBM/ProdData/WebASAdv/bin/redirector QREDIRECT
/qibm/userdata/webasadv/default
```

15.8 Test the servers

Now that the HTTP server, standalone Servlet Redirector, and WebSphere are running, we need to test the servers to make sure that everything is configured properly.

Using a Web browser, access the following URI (ralyas4c is the HTTP server and /webapp/examples/showCfg is the servlet):

```
http://ralyas4c/webapp/examples/showCfg
```

Figure 360 shows the results of the request.

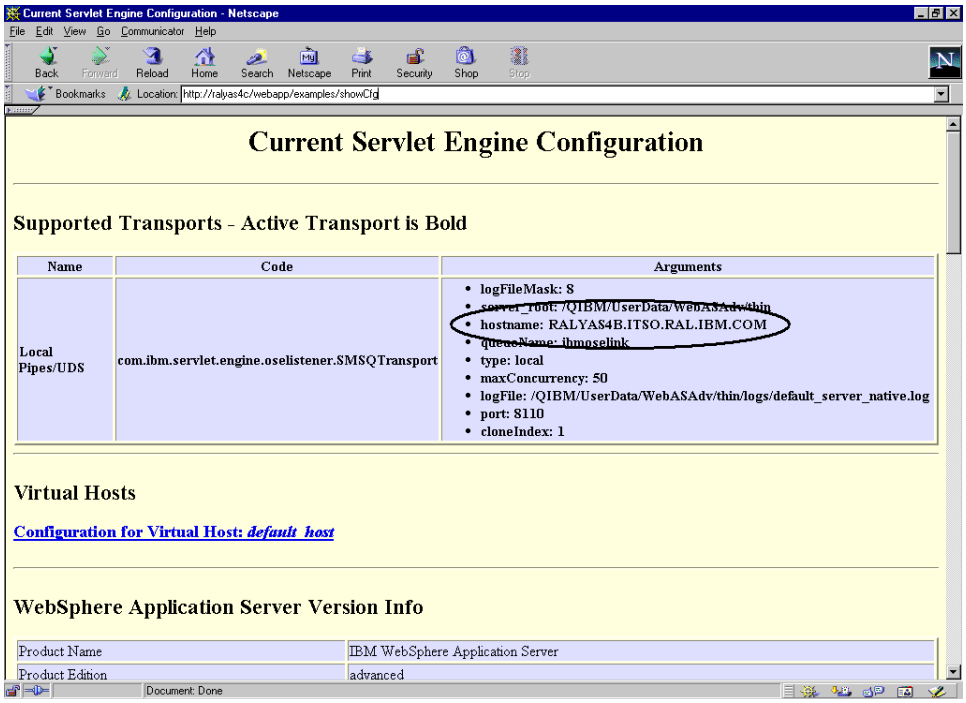

Figure 360. Output in browser

Notice that the hostname is the name of the node running the application server (RALYAS4B), and not the HTTP server (RALYAS4C).

15.9 Related topology

WebSphere Version 3.021 supports the use of clones and workload management with Servlet Redirector.

An HTTP server and a standalone Servlet Redirector will be installed on Machine A (RALYAS4C). Two application server clones will run on Machine B (RALYAS4B). Requests will be redirected from Machine A to the application server clones running on Machine B.

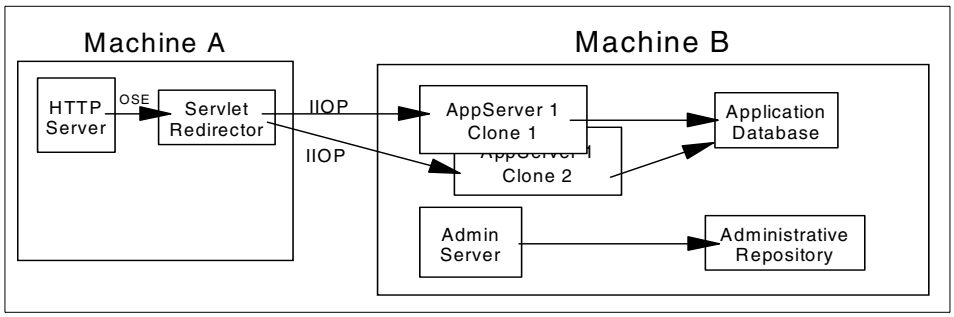

Figure 361. Standalone Servlet Redirector with clones

The clones are configured using the process described in 12.4, “Create the model” on page 357.

Note

Typically, you would not want to covert your initial Web application into a clone; however since we are using Servlet Redirector we need to select the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model’s clones to fail unless this server is made into a clone.

The configuration of the HTTP server and Servlet Redirector are the same as described previously in this chapter.

Chapter 16. Horizontal scaling with IBM Network Dispatcher

Unlike the previous chapters this environment utilizes an external load balancing mechanism, the *IBM Network Dispatcher*. This technology improves the performance of servers by distributing the incoming TCP/IP requests, in our case HTTP requests, amongst a group of servers typically known as a cluster. The *IBM Network Dispatcher* product utilizes intelligent load balancing, taking into account the availability, capability, workload and user definable criteria when determining which server the TCP/IP request is sent to.

This chapter provides instructions for the administrative tasks required to configure a WebSphere Administrative Server to support multiple cloned application servers, accessing a remote AS/400 for the repository and back office database.

We will discuss the following steps for setting up this configuration:

1. Configuration overview
2. Installation summary
3. Creating the ND environment
4. Configuring the WebSphere Application Server environments
5. Starting the Administrative Servers and console
6. Updating host aliases
7. Creation of the models
8. Creation of the clones
9. Changing the WLM
10. Configuring the HTTP server
11. Starting the servers
12. Testing the configuration
13. Related topologies

16.1 Overview of the configuration

A Web server and a full WebSphere Administrative Server are installed on Machine A (RALYAS4C) and Machine B (RALYAS4B). The WebSphere Administrative Server on Machine B can be connected either as Administrative agent as shown in Figure 362 on page 428 or as a full

Administrative Server as shown in Figure 363 on page 429. All HTTP requests arrive at the Network Dispatcher machine and are dispatched to one of the AS/400s for processing.

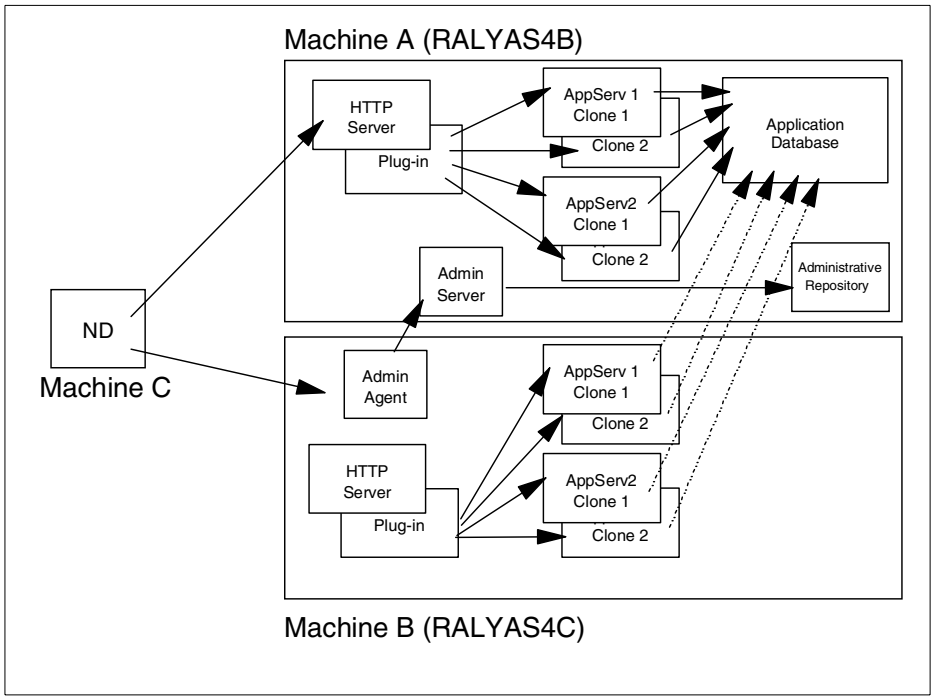

Figure 362. Network Dispatcher with two node WebSphere Domain connected using an Admin agent

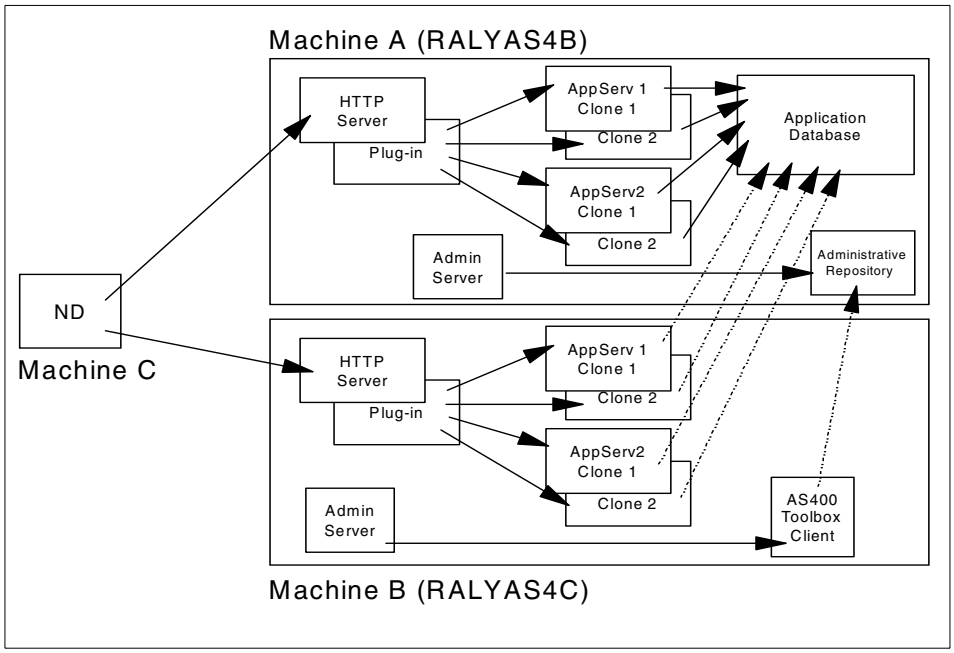

Figure 363. Network Dispatcher with two full Administrative Server nodes in a WebSphere domain

In our scenario both WebSphere Administrative Servers are configured identically.

16.2 Installation summary

Table 40 summarizes the software products and options required to support this topology.

Table 40. Product install

Machine A	OS/400 V4R4 or above, options 12 and 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1

Machine B	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and Option 2
	5769JC1 AS/400 Toolbox for Java 5769JC1
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and Option 1

Please refer to the following for further information:

- *AS/400 Software Installation,* SC41-5120
- `www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html`
- *Building AS/400 Appls for WebSphere Advanced 3.0,* SG24-5691 (available as a “redpiece” at http://www.redbooks.**ibm.com**/)

16.3 Creating the ND environment

This technology improves the performance of servers by distributing the incoming TCP/IP requests, in our case HTTP requests, amongst a group of servers typically known as a cluster. Network Dispatcher utilizes intelligent load balancing, taking into account the availability, capability, workload and user definable criteria when determining which server the TCP/IP request is sent to.

16.3.1 Configuring Network Dispatcher

The configuration of *IBM Network Dispatcher* is a complex process beyond the scope of this redbook. For information on using the *IBM Network Dispatcher* to load balance in your environment, refer to the following redbook:

IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher, SG24-5858

Within our scenarios, the *IBM Network Dispatcher* runs on an RS/6000 AIX system. We have included our sample scripts in Appendix G, “Sample Network Dispatcher configuration script” on page 545.

16.3.2 Configuring the AS/400 load balancing environment

Load Balancing with the *IBM Network Dispatcher* requires additional TCP/IP setup on the AS/400. We provide this information in 11.14, “Using AS/400

HTTP server in Network Dispatcher environment” on page 348. Configuring the environment.

16.4 Configuring WebSphere Administrative Servers and environment

The WebSphere Application Server instance utilized for this scenario is an additional instance. Unlike previous chapters, the HTTP server instances listen on the default TCP/IP ports. This is because the *IBM Network Dispatcher* provides advisors for monitoring HTTP and HTTPS requests that work on TCP/IP ports 80 and 443 respectively. If you require different TCP/IP ports for the HTTP server you must modify the *IBM Network Dispatcher* advisors. Additionally, the base topology of the node is imported from an XML template and is not the default topology.

Our chosen configuration and instance options follow.

16.4.1 Machine A: WebSphere Admistrative Server

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 41.

Table 41. Properties for RALYAS4B WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/scenario25
mntr.admin.name	SCN25ADMIN
install.initial.config	false
admin.dbSchema	EJSSCN25
ose.srvgrp.ibmappserve.clone1.port	8825
admin.bootstrapPort	7725
admin.lsdPort	9925

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports. As we are importing our configuration via XML, we changed our WebSphere Administrative Server instance to disable the creation of the default application server, as described in 11.6, “Configuring Admin Server instance to not install the default application server” on page 324.

Additionally, if Machine B is to be run as a full Administrative Server instance, we needed to undertake the additional setup described in 11.10.4, “Configuration changes for the remote AS/400” on page 332.

16.4.2 Machine B: WebSphere Administrative Server

16.4.2.1 Configuring as an Administrative agent

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 42.

Table 42. Properties for RALYAS4C WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/scenario25
mntr.admin.name	SCN25ADMIN
install.initial.config	false
admin.bootstrapHost	RALYAS4B
admin.bootstrapPort	7725
admin.lsdPort	9925
ose.srvgrp.ibmappserve.clone1.port	8825

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports. As we are importing our configuration via XML, we changed our WebSphere Administrative Server instance to disable the creation of the default application server, as described in 11.6, “Configuring Admin Server instance to not install the default application server” on page 324.

16.4.2.2 Configuring as a full Administrative Server

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 43.

Table 43. Properties for RALYAS4C WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/scenario25
mntr.admin.name	SCN25ADMIN
admin.dbSchema	EJSSCN25
install.initial.config	false

Configuration options	Value
admin.bootstrapPort	7725
admin.lsdPort	9925
ose.srvgrp.ibmappserve.clone1.port	8825

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports. As we have multiple instances of the HTTP server and we are not binding these to a specific IP address, we need to choose a distinct TCP/IP port for our HTTP server instance.

As we intend to connect to a remote Administrative Repository collection, we needed to undertake the additional setup describe in 11.10.5, “Configuration changes on additional Admin Server” on page 333. We chose to synchronize the password for both QEJB profiles.

16.5 Starting your WebSphere Application Servers

16.5.1 Machine A: WebSphere Administrative Server

Start the WebSphere Administrative Server on Machine A. If this is the default WebSphere Administrative Server, start the QEJBSBS subsystem in the QEJB library. If this is an additional WebSphere Administrative Server instance, refer to 11.5.3, “Starting the new instance and connecting an Admin Console” on page 322 for further information. When the server has started, we start an Administrative Console on a workstation.

16.5.2 Machine B: WebSphere Administrative Server

Start the WebSphere Administrative Server on Machine B. If this is the default WebSphere Administrative server, start the QEJBSBS subsystem in the QEJB library. If this is an additional WebSphere Administrative Server instance, refer to 11.5.3, “Starting the new instance and connecting an Admin Console” on page 322 for further information.

If you are running this machine as an Administrative agent, then you must start the full Administrative Server instance on Machine A first.

16.6 Update the host alias information of your virtual hosts

When utilizing *IBM Network Dispatcher* a common base URI is used to access the resources of any node in the repository. In this scenario we used www.itso.ibm.com. We updated the host alias information to reflect this as shown in Figure 364.

Note

Unless a requirement exists to utilize distinct HTTP server instances outside of the *IBM Network Dispatcher* environment, for example testing scenarios, it is recommended you remove the host alias information for the individual nodes.

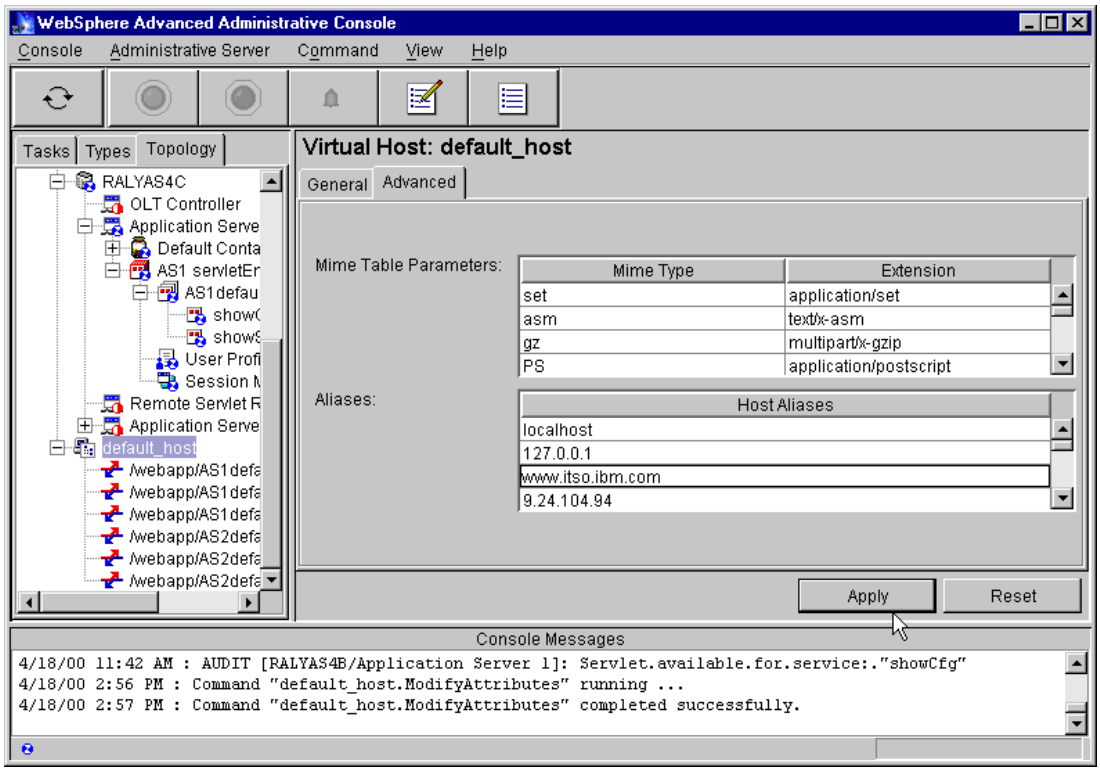

Figure 364. Updating the host alias information

16.7 Creation of the models

> **Note**
>
> Creating a model of a Web application server is identical across all platforms.

Within this scenario, we are creating models of two Web applications on node A. For brevity, we have shown this process for the first model only.

16.7.1 Creating the models

From the Topology pane of the WebSphere Administrative Console, select the first Web application you wish to clone, right click and select **Create->Model** as shown in Figure 365.

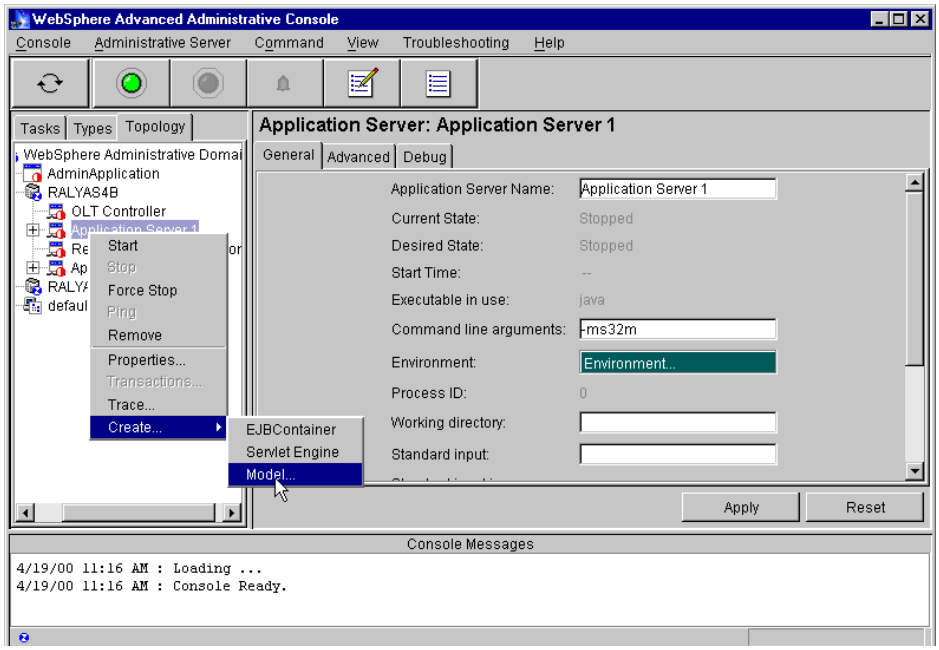

Figure 365. Creating a model of a Web application

In the Clone properties dialog box that is displayed, enter your chosen model name and check the box to recursively model all instance under the server as shown in Figure 366.

Note

Typically, you would not want to covert your initial Web application into a clone; however if you plan to enhance this topology to utilize the Servlet Redirector then you should tick the option to make this server a clone. This is due to an issue within the Servlet Redirector that causes requests to this model's clones to fail unless this server is made into a clone.

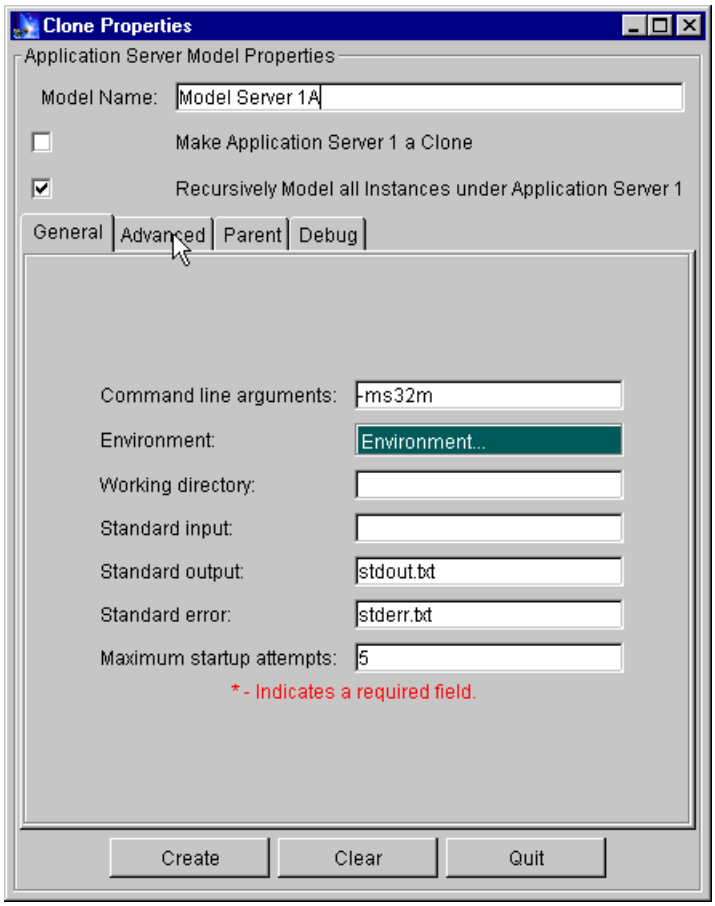

Figure 366. Defining the model properties

At this time you may wish to change the workload management selection policy. To do this, click the **Advanced** tab and you will be presented with this option, as shown in Figure 367 on page 437. For our example, we can leave

this as the default Round Robin Prefer Local, which is the preferred policy for this solution as all clones accessed will exist on a single AS/400.

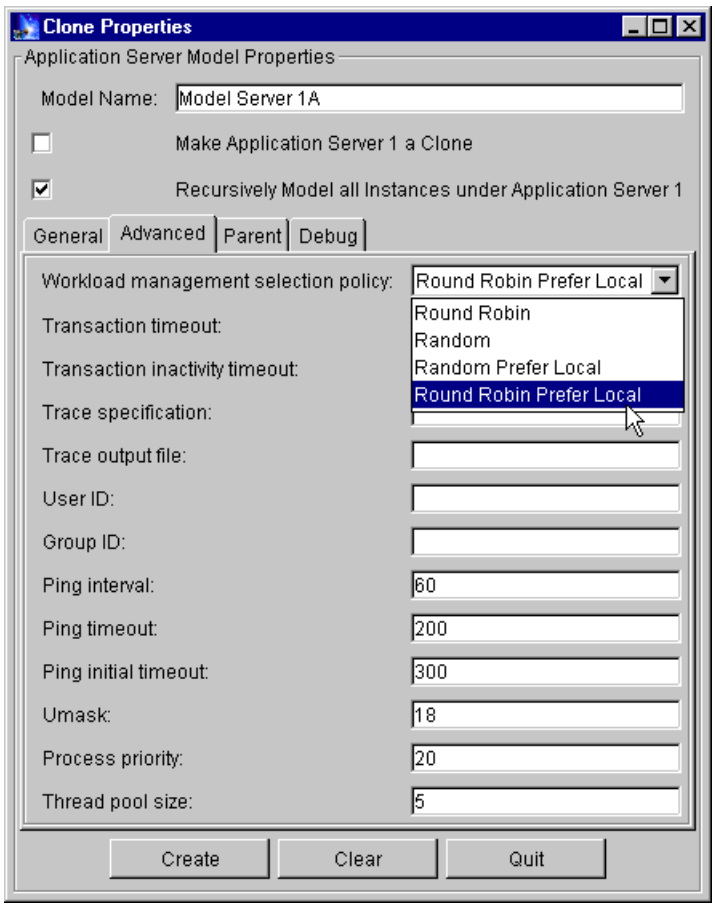

Figure 367. Changing the workload management selection policy

Note

Currently, with V3.02 any changes you make to the workload management selection policy from this tab are not reflected in the generated model.

To change the workload management selection policy, you should change the properties of the resultant model.

After completing any changes you wish to make, click the **Create** button to generate the model.

When the model has been created, you will see the following message.

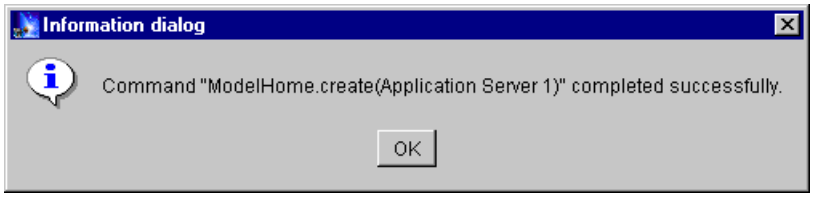

Figure 368. Model created message

You should now repeat this process for the second Web application. After completing your configuration, you should see your models in the Topology tree surrounded by the cyan square, used to identify models.

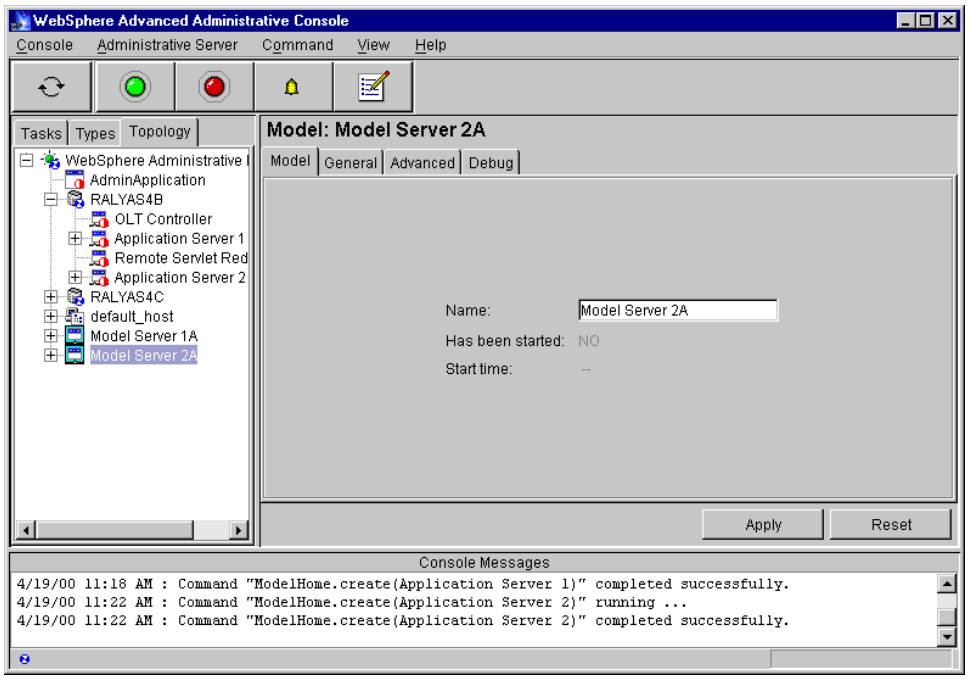

Figure 369. WebSphere topology showing models

16.8 Changing the WLM configuration

Once the Web application models have been created, you can change the workload management selection policy to suite your requirements. Select the model in the Topology pane of the Administrative Console and then click the **Advanced** tab. The default is Round Robin Prefer Local which is the preferred policy for this solution as all clones exist on a single AS/400. If you wish to change the policy, select one from the drop down list and then click **Apply** as shown in Figure 370.

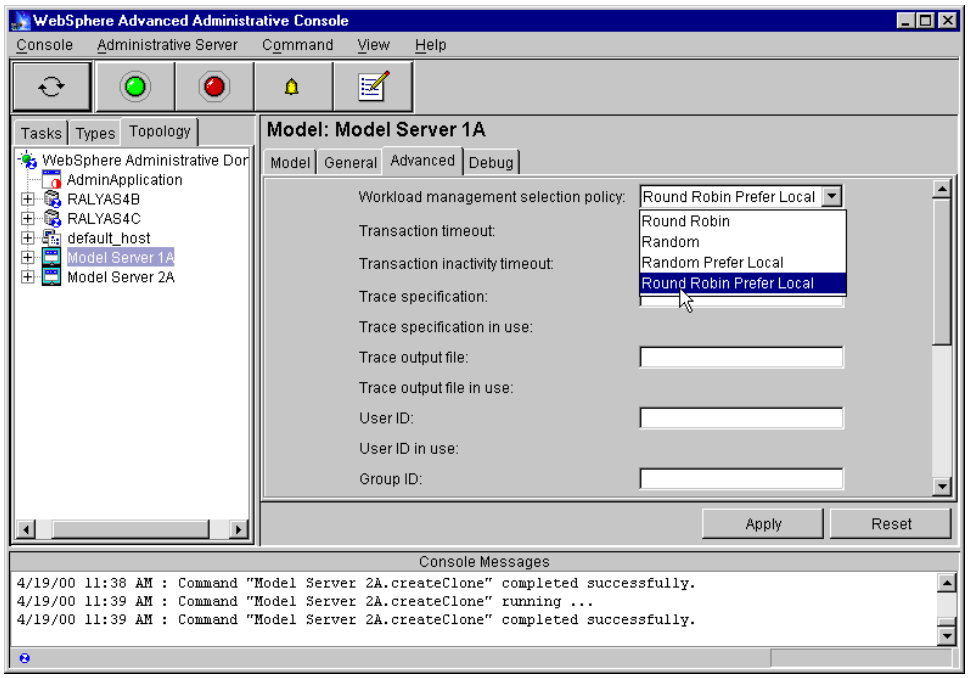

Figure 370. Changing the workload management selection policy

16.9 Creation of the clones

Note

Creating a clone of a model is identical across all platforms.

Within this scenario, we are creating two clones of each model we generated above on each node. For brevity, we have shown this process for the first clone only on each node.

> **Note**
>
> If you ticked the option to make the parent application server a clone of the model when you generated the model, you will only need to create a single clone for each model, as the first clone will be your original server.

16.9.1 Creating the clones for Machine A

From the Topology pane of the WebSphere Administrative Console, select the first model you wish to clone with the context menu button (right click) and select **Create->Clone** as shown in Figure 371.

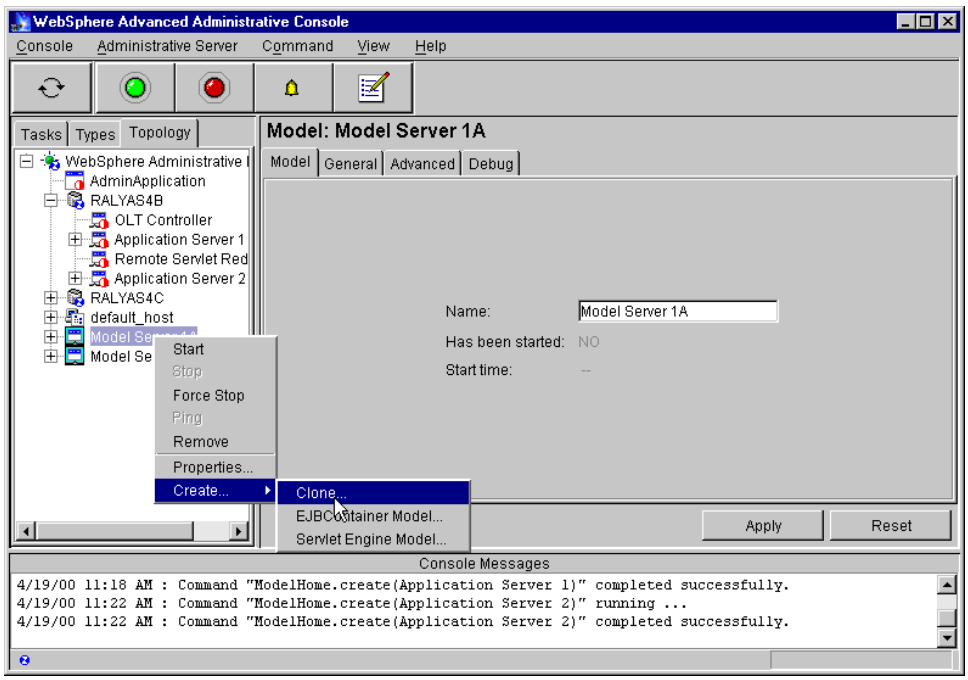

Figure 371. Creating a clone of the model

In the Clone Parent dialog box that is displayed, enter your chosen clone name. Select the node on which you require the clone to be installed, by expanding the Nodes tree and clicking on the relevant node with the context

menu button as shown in Figure 372. Click the **Create** button to generate the clone.

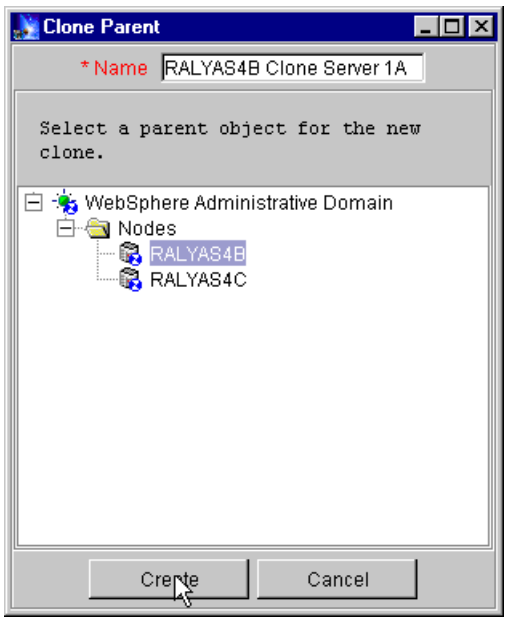

Figure 372. Defining the clone

When the clone has been created, you will see the following message.

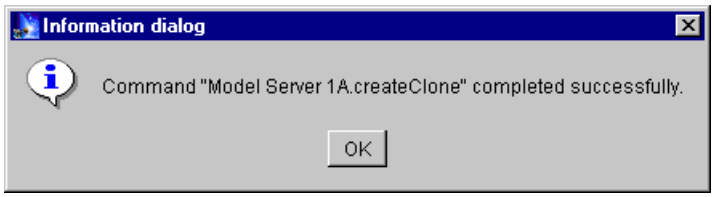

Figure 373. Clone created message

You should now repeat this process for the remaining clones for machine A.

16.9.2 Creating the clones for Machine B

From the Topology pane of the WebSphere Administration Console, select the first model you wish to clone with the context menu button and select **Create->Clone** as shown in Figure 374 on page 442.

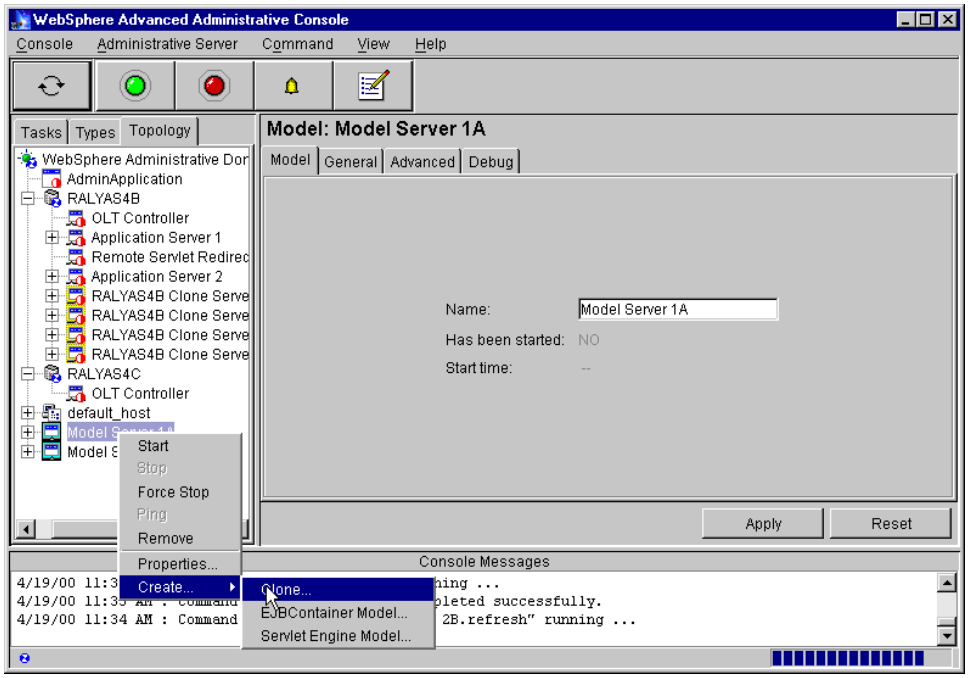

Figure 374. Creating a clone of the model

In the Clone Parent dialog box that is displayed, enter your chosen clone name. Select the node on which you require the clone to be installed, by expanding the Nodes tree and clicking on the relevant node as shown in Figure 375 on page 443. Click the **Create** button to generate the clone.

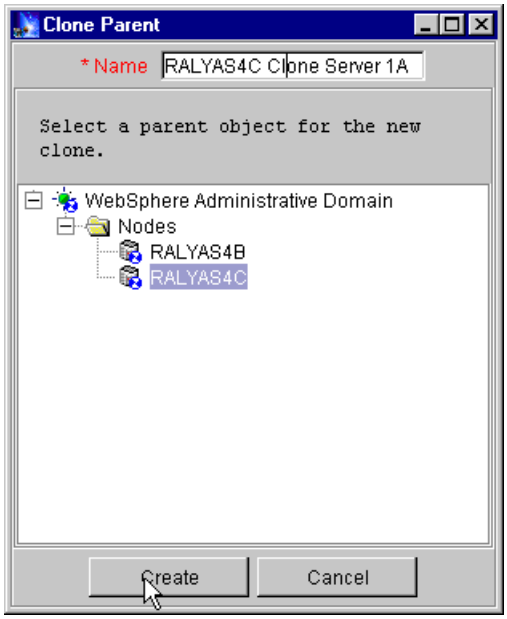

Figure 375. Defining the clone

When the clone has been created, you will see the following message.

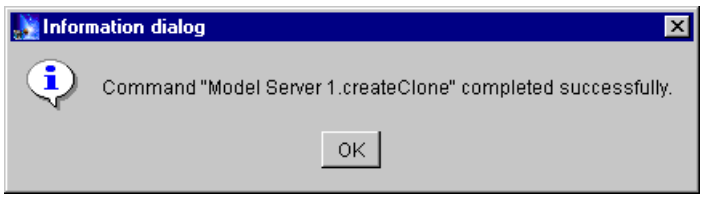

Figure 376. Clone created message

You should now repeat this process for the remaining clones for Machine B.

After completing your configuration, you should see your clones in the Topology tree surrounded by the yellow square, used to identify clones as shown in Figure 377 on page 444.

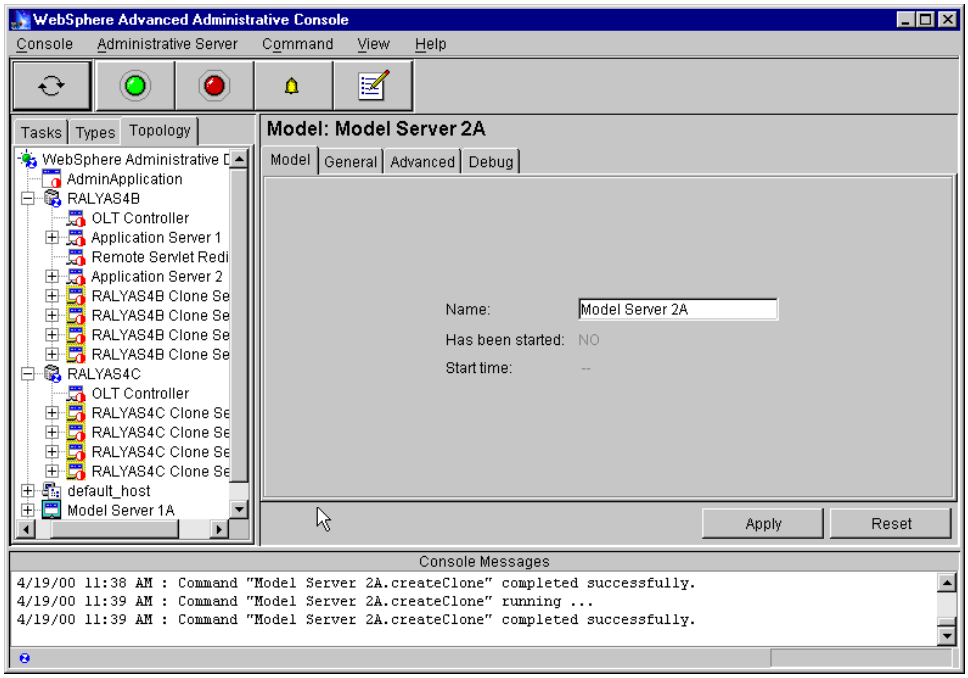

Figure 377. WebSphere topology showing clones

16.10 Configuring the HTTP server

When you configure the HTTP server instance you will use for the WebSphere Application Server, you must ensure that any WebSphere Application Server V2.xx references are removed from your configuration file. If you suspect that the HTTP server instance was previously used for WAS V2 Standard Edition support then you should remove the V2 Standard edition configuration information

16.11 Starting the servers

Now you are ready to start the models and Web servers.

16.11.1 Starting the models

Start the first model by clicking on the model name, with the context menu button (right click) in the Topology Tree and select **Start** as shown in Figure 378 on page 445.

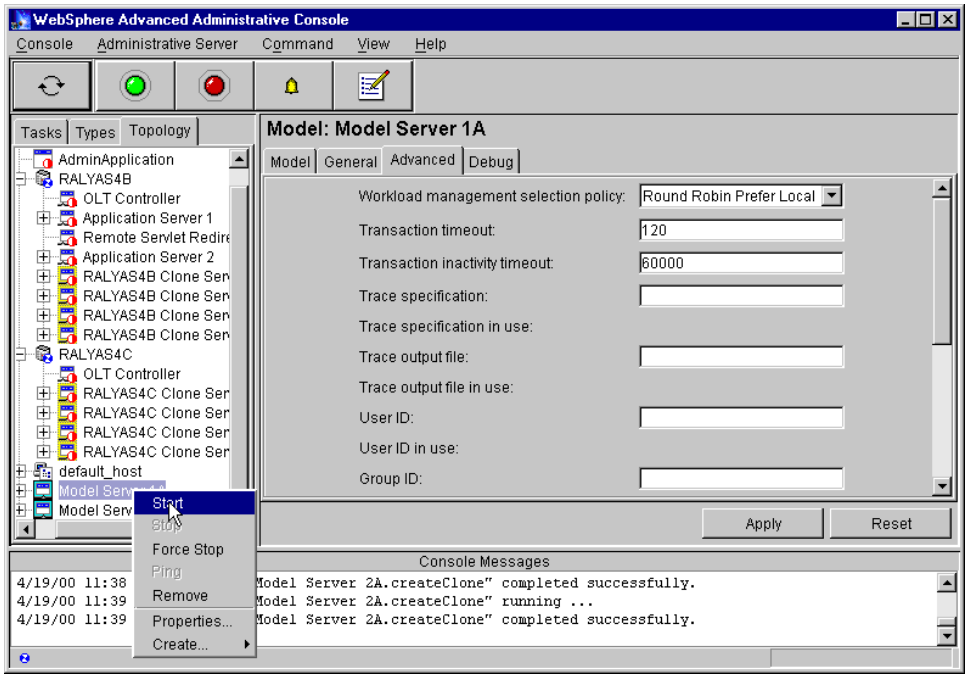

Figure 378. Starting the model

Once the model has started you will see a message telling you that your model has started successfully.

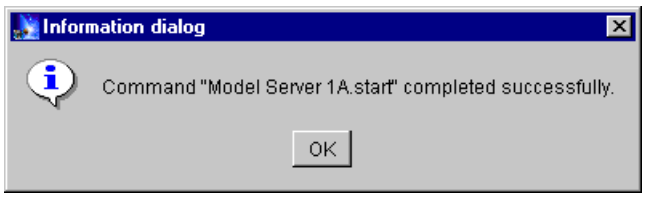

Figure 379. Model started information dialog

Repeat the process to start the second model.

The clones for the models you started, should now show as started in the Topology pane as shown in Figure 380 on page 446.

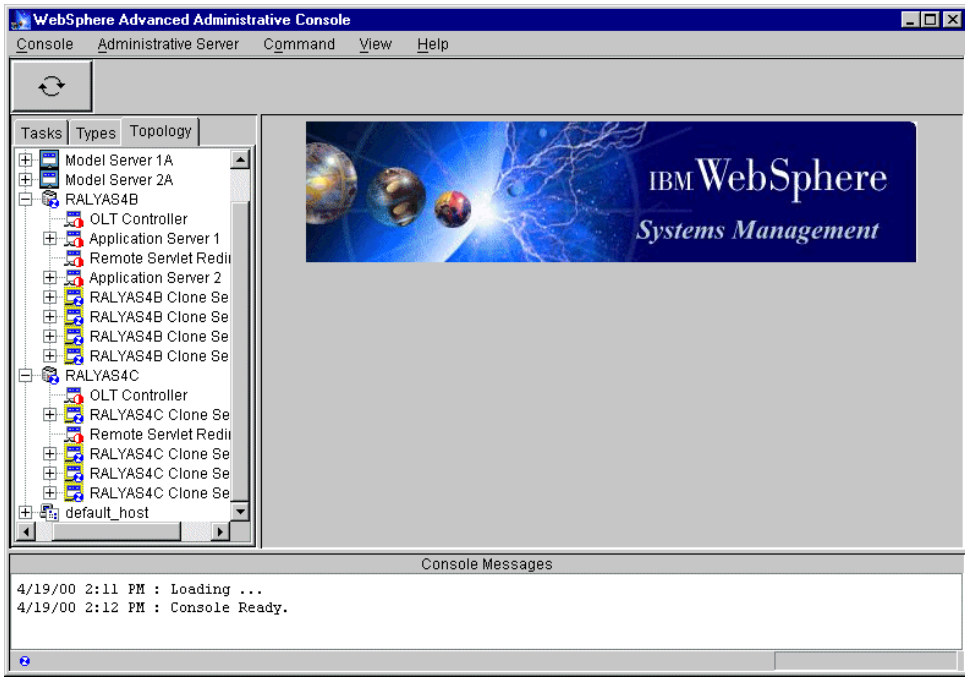

Figure 380. Topology pane, showing started clones

Note

At the time of writing, issues with the Administrative Console cause the clone icons to not always display as started. We have found that refreshing the node subtree will only occasionally correct the icons. Stop and restart the Administrative Console to display the correct information.

Repeat the process to start the second model.

16.11.2 Starting the HTTP server(s)

Two methods for starting the HTTP server exist on the AS/400. The first method, as shown in Figure 381 on page 447 is from a 5250 command line.

```
MAIN                          AS/400 Main Menu
                                                          System:   AS4B
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. Client Access/400 tasks

    90. Sign off

Selection or command
===> STRTCPSVR SERVER(*HTTP) HTTPSVR(WASASCN25)

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
```

Figure 381. Starting the HTTP server from command line

Alternatively, you can start your HTTP server instance from the Work with Server instance option of the HTTP Configuration and Administration Web pages, as shown in Figure 382 on page 448.

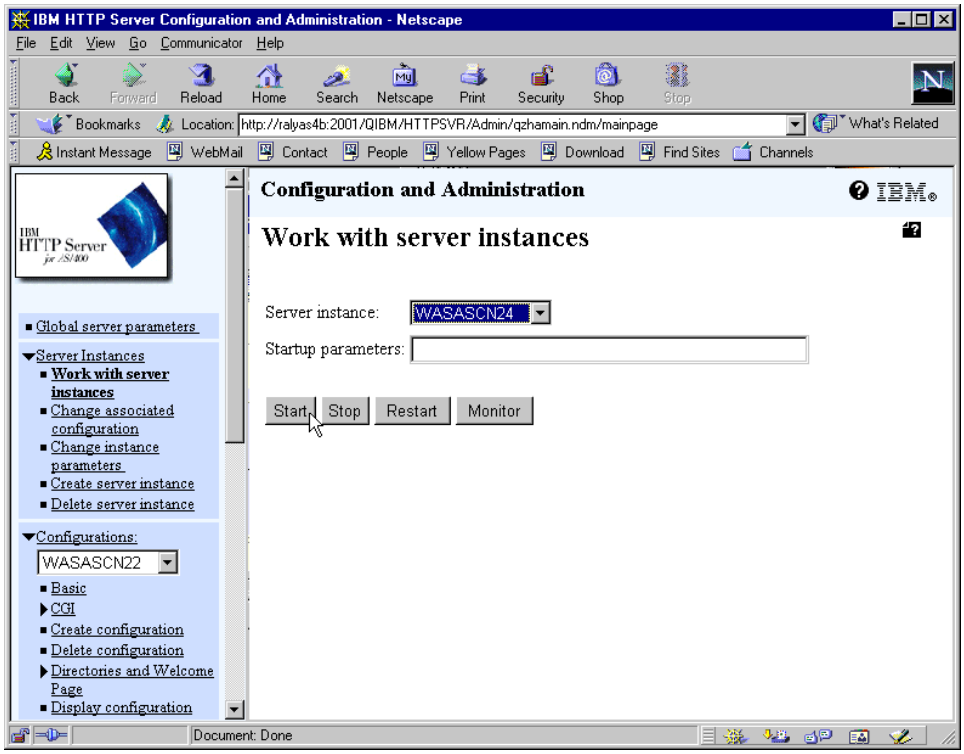

Figure 382. Starting the HTTP server instance from the HTTP configuration and administration Web page.

For brevity, we have only included the information to start the Web server

16.12 Test the servers

Now that the Network Dispatcher, the HTTP server, and WebSphere are running, we need to test the servers to make sure that everything is configured properly.

Start up an instance of your preferred browser, in our examples we use Netscape, and enter the following URI (www.itso.ral.ibm.com is the cluster address of ND) and press Enter:

```
http://www.itso.ral.ibm.com/webapp/AS1default/showCfg
```

As the initial load balancing is undertaken by the *IBM Network Dispatcher*, you cannot guarantee which HTTP server instance the request will be forwarded to. It is possible however to predict how requests are processed

when they reach a particular WebSphere instance based on the workload selection policy. For our scenario, the policy was round robin prefer local and our testing confirmed this.

Note

If the AS/400s utilized are different models or the same model with different processor features then you may see a pronounced preference for one AS/400 in the distribution of requests from the *IBM Network Dispatcher*. This is the load balancing algorithm detecting the performance difference between the AS/400s.

16.13 Related topologies

Using the *IBM Network Dispatcher* allows us to perform intelligent load balancing and horizontal scaling for HTTP servers in our scenarios.

16.13.1 Variation 1: OSE remote scenario

Two AS/400s acting as the HTTP servers (Machines A and C) will use OSE Remote to route requests to nodes on Machines B and C respectively. Each node will be running a clone or clones of one or more application servers. The nodes will share a repository stored on Machine B.

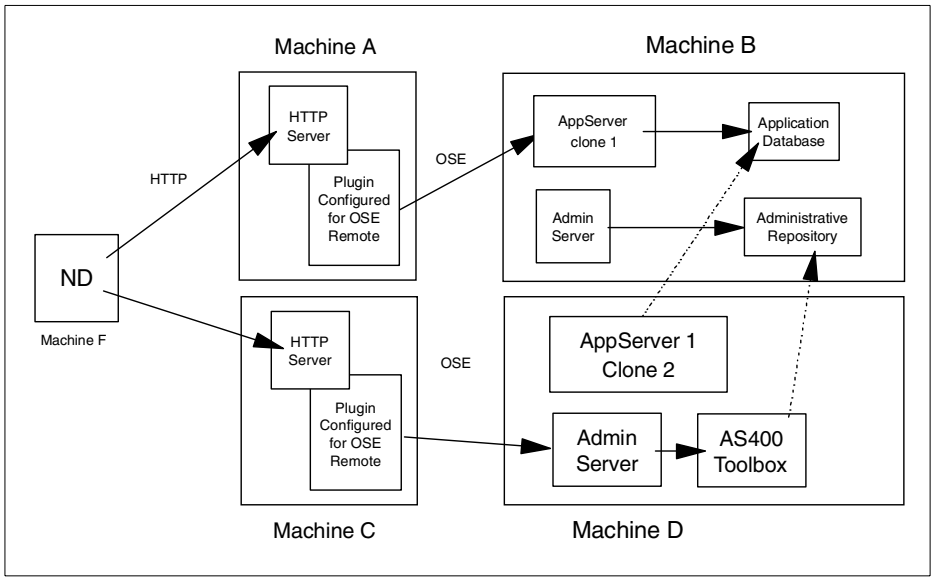

Figure 383. OSE Remote with Network Dispatcher

Note

Although Figure 370, shows the node on Machine D running as a full Administrative Server, this node could be run as an Administrative agent. Refer to 11.8.1, “High Availability Considerations when using an Administrative Agent” on page 299 for additional information.

16.13.2 Variation 2: OSE Remote with high availability

This is a variation of the scenario in 16.13.1, “Variation 1: OSE remote scenario” on page 449. Two AS/400s acting as the HTTP servers (Machines A and C) will use OSE Remote to route requests to nodes on Machines B and C. Each node will be running a clone or clones of one or more application servers. The nodes will share a repository stored on Machine B.

This topology has the advantage over the previous topology in that we have failover should either WebSphere Administrative Server on Machine B or Machine D fail or be taken down for service.

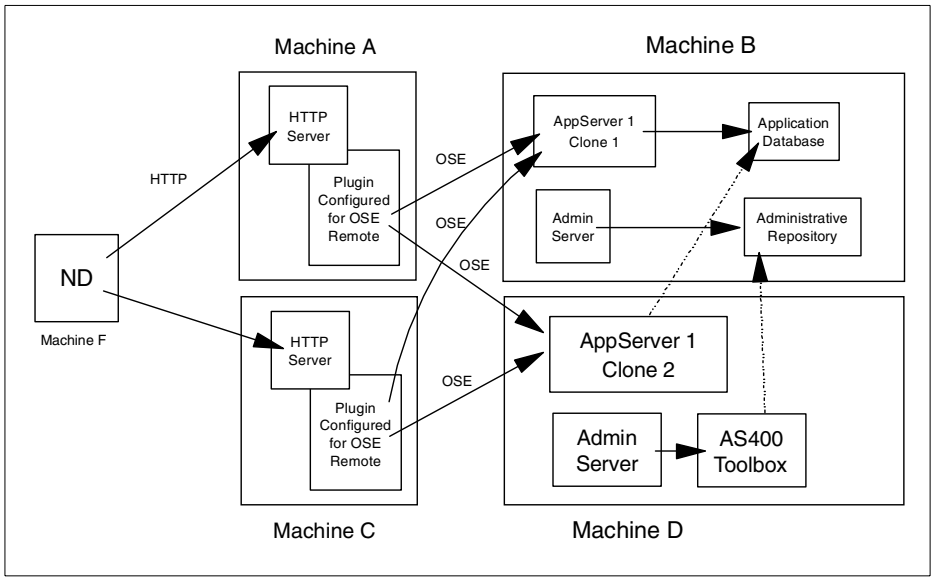

Figure 384. OSE Remote with high availability

Note

Although Figure 384, shows the node on Machine D running as a full Administrative Server, this node could be run as an Administrative agent. Refer to 11.8.1, “High availability considerations when using an Admin agent” on page 325 for additional information.

16.13.3 Variation 3: multiple distinct domains

This section describes how we can utilize distinct WebSphere domains and the *IBM Network Dispatcher* to create highly available environments. As each domain uses a separate repository on a different machine, we can create a cluster of WebSphere domains to support our environment. Although this type of scenario provides a greater level of high availability, it does so with an increased level of complexity for maintaining and synchronizing the environments.

Two AS/400s are used to provide high availability via horizontal scaling.

Each machine runs the same application environment to its own Administrative Repository. Should Machine A fail or be taken down for

maintenance then all requests will be automatically sent to Machine B for processing.

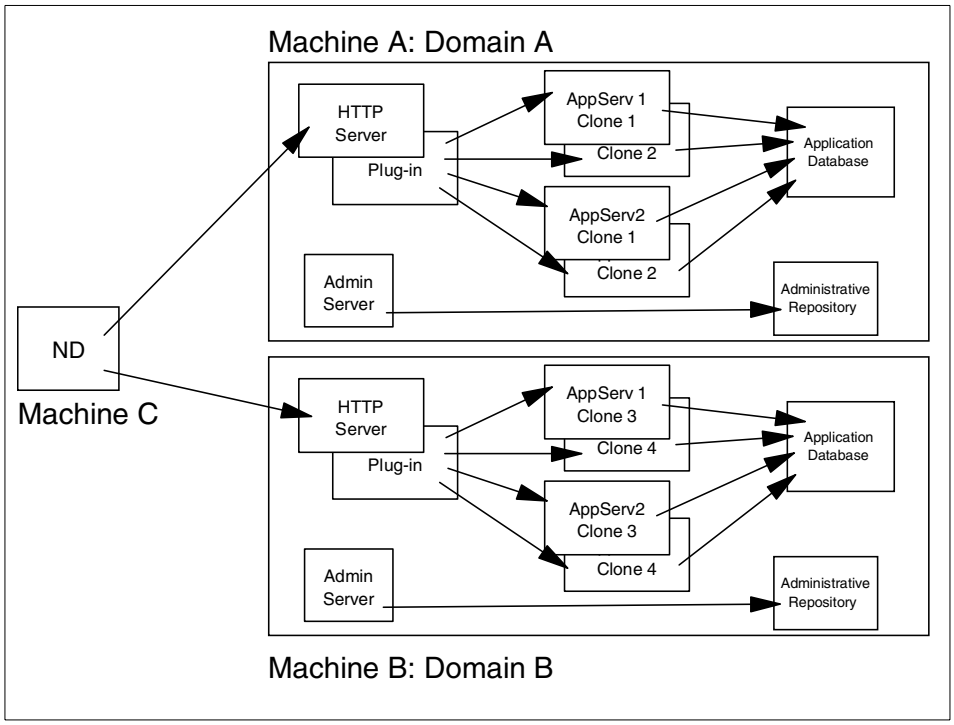

Figure 385. Multiple distinct domains with Network Dispatcher

Chapter 17. Three-tier topologies

Our definition of a three-tier environment is one where the environment is split, separating the HTTP server, servlets and Enterprise JavaBeans onto distinct machines.

Each of these components can be scaled and workload managed independently or in conjunction with each other. Both vertical and horizontal scaling of each component can be used to achieve this, however only horizontal scaling will provide a high availability environment.

This chapter provides instructions for the administrative tasks required to configure the distinct components on individual AS/400's.

We will discuss the following steps for setting up this configuration:

1. Configuration overview
2. Installation summary
3. Configuring the WebSphere Application Server environments
4. Configuring the EJB node
5. Configuring the servlet node
6. Configuring the plugin for OSE Remote
7. Testing the configuration

17.1 Overview of the configuration

This is essentially the topology from Chapter 13, "OSE Remote" on page 371. We have simply moved our EJBs and repository onto a separate machine.

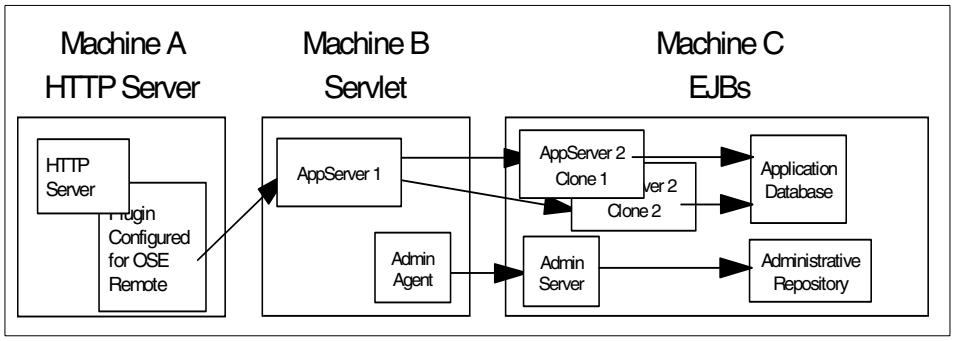

Figure 386. Three-tier OSE remote environment using Administrative agents

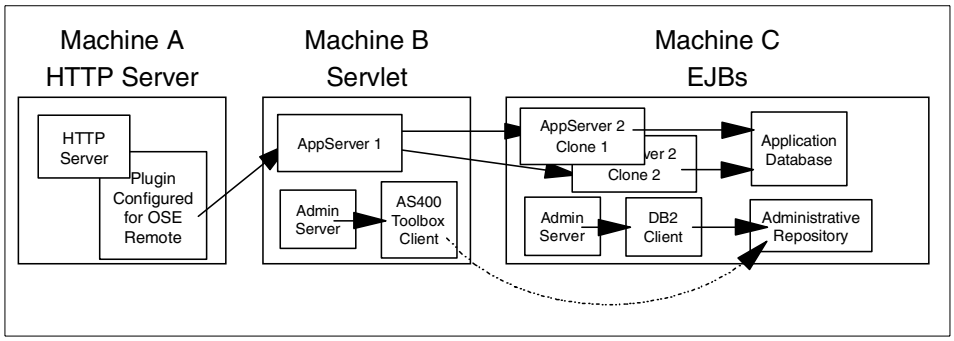

Figure 387. Three-tier OSE remote environment with full Administrative Servers

All HTTP requests received by Machine A are redirected, or forwarded, to the application server on Machine B. The servlets running on Machine B use Enterprise JavaBeans running on Machine C to read, change, update or delete information from the application database.

This is a typical scenario where only Machine A would be the Internet DMZ.

17.2 Installation summary

Table 44 lists the products we need to install on the three AS/400s before we can begin.

Table 44. Product Installation

Machine A	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and option 2
	5769DG1 IBM HTTP Server for AS/400
	5733WA2 / 5733WA3 WebSphere *BASE and option 1
Machine B	OS/400 V4R4 or above, option 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and option 2
	5769JC1 AS/400 Toolbox for Java 5769JC1
	5733WA2 / 5733WA3 WebSphere *BASE and option 1
Machine C	OS/400 V4R4 or above, options 12 and 30 must be installed
	5769TC1 TCP/IP Connectivity Utilities for AS/400
	5769JV1 AS/400 Developer Kit for Java *BASE and option 2
	5733WA2 / 5733WA3 WebSphere *BASE and option 1

Please refer to the following for further information:

- *AS/400 Software Installation,* SC41-5120
- `www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html`
- *Building AS/400 Appls for WebSphere Advanced 3.0,* SG24-5691(available as a “redpiece” at http://www.redbooks.**ibm.com**/)

17.3 Configuring WebSphere Administrative Servers and environment

The WebSphere Application Server instance utilized for this scenario is an additional instance. The base topology of the node is imported from an XML template and is not the default topology. Additionally our testing servlet and EJB are not generally available outside of IBM.

Our chosen configuration and instance options follow:

17.3.1 Machine C: EJB WebSphere Administrative Server

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 45.

Table 45. Properties for RALYAS4B WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/Trade
mntr.admin.name	TRADEADMIN
install.initial.config	false
admin.dbSchema	EJSSCN25
ose.srvgrp.ibmappserve.clone1.port	8899
admin.bootstrapPort	7799
admin.lsdPort	9999

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports.

Additionally, if Machine B is to be run as a full Adminstrative Server instance, we need to undertake the additional setup describe in 11.10.4, “Configuration changes for the remote AS/400” on page 332.

17.3.2 Machine B: Servlet WebSphere Administrative Server

17.3.2.1 Configuring as an Administrative agent

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 46.

Table 46. Properties for RALYAS4C WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/trade
mntr.admin.name	TRADEADMIN
install.initial.config	false
admin.bootstrapHost	RALYAS4B
admin.bootstrapPort	7799
admin.lsdPort	9999
ose.srvgrp.ibmappserve.clone1.port	8899

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports. As we are importing our configuration via XML, we changed our WebSphere Administrative Server instance to disable the creation of the default application server, as described in 11.6, “Configuring Admin Server instance to not install the default application server” on page 324.

17.3.2.2 Configuring as a full Adminstrative Server

This AS/400 was configured as an additional instance, and our configuration template is shown in Table 47.

Table 47. Properties for RALYAS4C WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/Trade
mntr.admin.name	TRADEADMIN
admin.dbSchema	EJSSCN25
install.initial.config	false
admin.bootstrapPort	7799
admin.lsdPort	9999
ose.srvgrp.ibmappserve.clone1.port	8899

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports. As we have multiple instances of the HTTP server and we are not binding these to a specific IP address, we need to choose a distinct TCP/IP port for our HTTP server instance.

As we intend to connect to a remote Adminstrative Repository collection, we need to undertake the additional setup described in 11.10.5, “Configuration changes on additional Admin Server” on page 333. We chose to synchronize the password for both QEJB profiles.

17.3.3 Machine A: OSE Remote WebSphere Administrative Server

This AS/400 has the WebSphere Adminstrative Server environment installed for OSE Remote support. It was configured as an additional instance, and our configuration template is shown in Table 48.

Table 48. Properties for RALYAS4B WebSphere instance

Configuration options	Value
Instance directory	/QIBM/UserData/WebAsAdv/Trade
mntr.admin.name	TRADEADMIN
install.initial.config	false
admin.dbSchema	EJSSCN25
ose.srvgrp.ibmappserve.clone1.port	8899
admin.bootstrapPort	7799
admin.lsdPort	9999

We updated the three property files to reflect our new directory structure, repository name, admin server name and chosen TCP/IP ports.

As we are only utilizing OSE Remote, only the bootstrap.properties file will be used by the HTTP plug-in.

17.4 Configure the EJB node

At first, you configure the EJB node. You should do the following steps to configure it.

1. Place the deployed JAR
2. Start the Administrative Server
3. Start the Administrative Console
4. Create an application server
5. Create a container
6. Create an enterprise bean
7. Start EJB application server

17.4.1 Place the EJB JAR file

You have to place your EJB JAR on Machine C.

17.4.2 Start the Administrative Server

Start the WebSphere Administrative Server on Machine C. If this is the default WebSphere Administrative Server, start the QEJBSBS subsystem in the QEJB library. If this is an additional WebSphere Administrative Server instance, refer to 11.5.3, “Starting the new instance and connecting an Admin Console” on page 322 for further information. When the server has started, we start an Administrative Console on a workstation.

17.4.3 Start the Administrative Console

When the server has started, we start an Administrative Console on a workstation.

17.4.4 Create an application server

In the console, highlight and right click Machine C.

Select **Create** and **Application Server** and then you will see the Create Application Server window.

You specify your application server name in the Application Server Name field then click **OK**.

17.4.5 Create a container

Create an EJB container for your EJB application. To do so, highlight the application server which you created in the previous step and right click it.

Select **Create** and **EJBContainer** then you will get a Create EJBContainer window.

You specify the name of your container then click **OK**.

17.4.6 Create an enterprise bean

Install/create your enterprise bean into the new EJB container just created.

Highlight the container which you created and right click it.

Select **Create** and **EnterpriseBean** then you will get a Create EnterpriseBean window.

Click the **Browse..** button and you will see the Open window.

Then select the file (EJB application) which you want to install.

You may get the following Confirm dialog depending on your JAR file.

We clicked Yes for our sample application.

Figure 388. Confirm dialog

Then click **OK** to create the EnterpriseBean.

Now, you can see the EnterpriseBean in the Administrative Console.

17.4.7 Start EJB application server

You are ready to start your EJB application server. Select your application server and click Start button.

17.5 Configure the servlet node

Next, you configure the servlet node. You should do the following steps to configure it.

1. Place the deployed JAR
2. Start the Administrative Server
3. Refresh the Administrative Console
4. Add host aliases
5. Create an application server
6. Create a servlet engine
7. Update transport type for OSE Remote
8. Create a web application
9. Create a servlet
10. Start servlet application server

17.5.1 Place the EJB JAR file

You have to place your EJB JAR file (at least EJB client application) on Machine B.

17.5.2 Start the Administrative Server

Start the WebSphere Administrative Server on Machine B. If this is the default WebSphere Administrative Server, start the QEJBSBS subsystem in the QEJB library. If this is an additional WebSphere Administrative Server instance, refer to 11.5.3, “Starting the new instance and connecting an Admin Console” on page 322 for further information.

If you are running this machine as an Administrative agent, then you must start the full Administrative Server instance on Machine A first.

17.5.3 Refresh the Administrative console

After the Administrative Server on Machine B starts completely, you should refresh your Administrative Console to see it.

17.5.4 Add host aliases

Using the Administrative Console, configure the host aliases for Machine A. You should add the IP address, the host name and fully qualified host.

17.5.5 Create an application server

Select Machine B and right click.

Select **Create** and **Application Server** then you will get the Create Application Server window.

You specify your application server name.

17.5.6 Adding a classpath

In the Command Line Arguments field, we specified the classpath with the EJB JAR file which contains a EJB client stub then click **OK**.

17.5.7 Create a servlet engine

Select Machine B and right click.

Select **Create** and **Servlet Engine** then you will get the Create Servlet Engine window.

You specify your servlet engine name then click **OK**.

17.5.8 Update transport type for OSE Remote

The servlet engine needs to be configured to use sockets instead of local pipes. You need to specify INET sockets for the transport type of OSE queue.

Select the servlet engine of the Application Server under Machine B.

Select the **Advanced** tab inside the Servlet Engine panel, and choose OSE for the Queue Type and click **Settings**.

You will get the Edit Servlet Engine Transport window. Then choose **INET Sockets** for Transport Type.

Then click **OK**. You will be returned to the Administrative Console.

Click the **Apply** button. The servlet engine is now updated.

17.5.9 Create a Web application

Create the Web application in the servlet engine which you created in the previous step.

Select the servlet engine you just created, and right click.

Select **Create** and **Web Application** then you will get the Create Web Application window.

You specify your Web application name the click **OK**.

17.5.10 Create a servlet

Now, you can create your servlet on Machine B.

Select the Web application which you just created and right click.

Select **Create** and **Servlet** then you will get the Create Servlet window.

You specify your servlet name and servlet class name.

Then click **Add** to specify the Servlet Web Path.

You will get the Add Web Path to Servlet window.

Then you specify your servlet path and click **OK**.

You will get back to the Create Servlet window.

Please verify that Servlet Name, Web Application, Servlet Class Name and Servlet Web Path List are specified.

Then click **OK**.

Now you can see the servlet which you just created in the Administrative Console.

17.5.11 Start servlet application server

You are ready to start the servlet application server. Select the servlet application server and click Start button.

17.6 Configure the plug-in for OSE Remote

You configured and started your EJB application server on Machine C and servlet application server on Machine B.

For the next step, you need to configure the plug-in for OSE Remote.

Configuring the OSE Remote environment for this three-tier topology is identical to configuring OSE Remote for a two-tier environment. OSE Remote only forwards servlet requests from the HTTP plug-in to the second-tier machine running the servlets.

Refer to 13.3.2, “Configure the transport type” on page 374 and 13.4, “Configure the plug-in for OSE Remote on Machine A” on page 376 in Chapter 13, “OSE Remote” on page 371 for the necessary information.

17.7 Start the HTTP server

Now you need to start the HTTP server.

When you configure the HTTP server instance that you will use for the WebSphere Application Server, you must ensure that any WebSphere Application Server V2.xx references are removed from your configuration file. If you suspect that the HTTP server instance was previously used for WAS V2 Standard Edition support then you should remove the V2 Standard edition.

17.8 Confirming the scenario is working

Now, you are ready to test your WebSphere three-tier configuration.

Start up an instance of your preferred browser, in our examples we use Netscape, enter the following URI and press Enter.

```
http://BRS2AM/trade/beanthere?count=5
```

You should see an HTML page similar to Figure 389. This indicates that the servlet request was redirected to the WebSphere Application Server on Machine B.

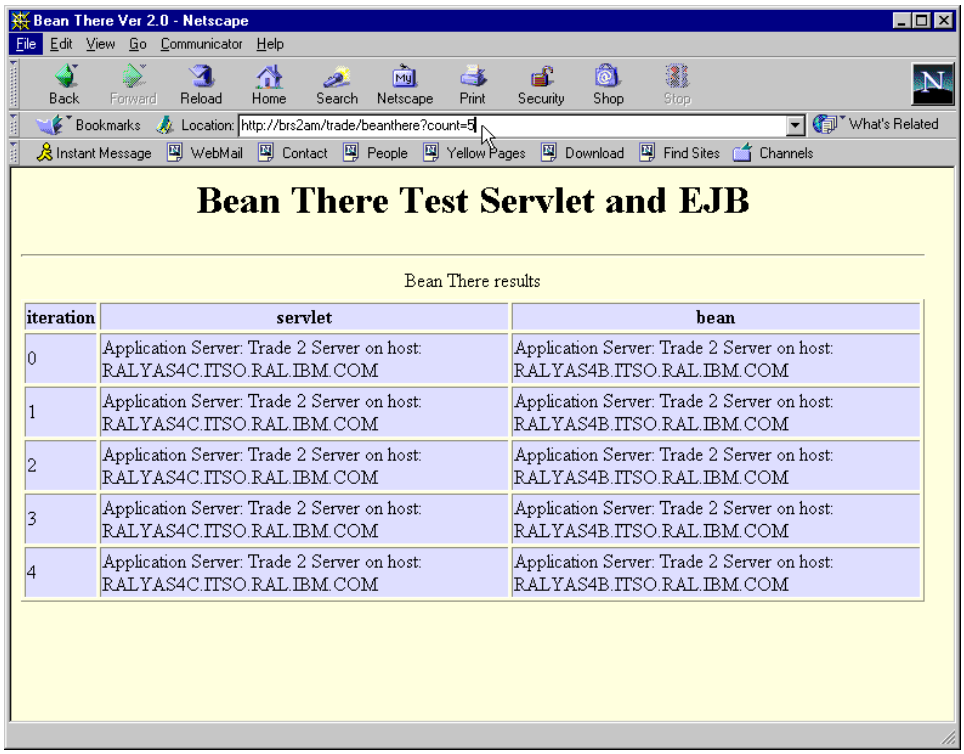

Figure 389. HTML page returned by HTTP request

Appendix A. Firewall considerations

This appendix examines the options when running WebSphere Advanced V3.02x and V3.5 in a DMZ configuration. In this configuration a firewall is configured between the Web server and the WebSphere Application Server. This configuration raises several issues.

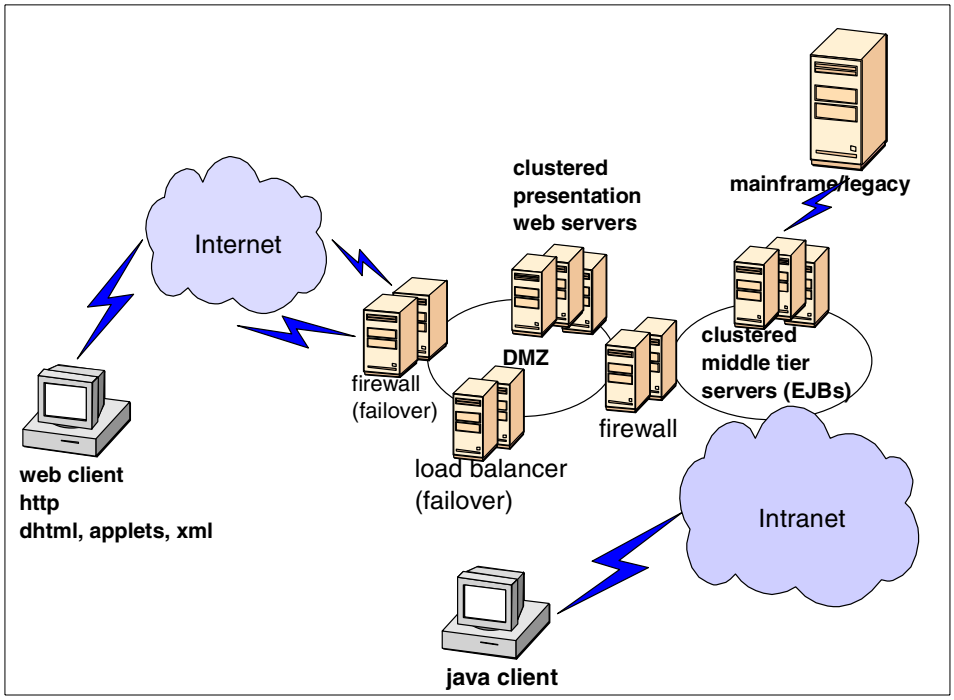

Figure 390. WebSphere and DMZ

We will also discuss the configuration which has a firewall between two WebSphere Application Servers; one is for the servlets and the other is for EJBs.

A.1 Database access

One of the main purposes of a DMZ configuration is to protect the business logic and data in the environment from unauthorized access. Since a WebSphere Administrative Server needs access to a database for its configuration information, it is often not a viable solution to run an Administrative Server in the DMZ.

A.2 Network address translation

A firewall which runs Network Address Translation (NAT) receives packets for one IP address, and translates the headers of the packet to forward the packet to a second IP address. This causes a problem with some complex protocols, such as RMI/IIOP, where IP addresses are embedded in the body of the IP packet. These IP addressees are not translated, thus breaking the conversation.

A.3 Configuration selections

There are five selections which we will discuss:

1. OSE Remote
2. Thick Servlet Redirector
3. Thin Servlet Redirector
4. Admin-agent Thick Servlet Redirector
5. Reverse proxy / IP forwarding

A.3.1 OSE Remote

The OSE protocol is used to communicate from the WebSphere plug-in to a local servlet engine or a local Servlet Redirector. OSE was originally developed to over a network but it was not originally supported to run in that configuration. With some slight configuration modifications, the OSE protocol can be used to forward requests to a remote servlet engine directly. Because OSE is a simple protocol, it does not have the embedded IP address problems that RMI/IIOP has.

For each clone that OSE will access through the firewall, a port must be opened to allow traffic through. This is the port that the servlet engine listens to for requests. You can verify the port number of each servlet engine on the Administrative Console as shown in Figure 60 on page 101.

If you want to use WebSphere security for resources which are on a Web server, you need to open another port which is specified in the <was_dir>/properties/bootstrap.properties as described in 5.6, “Configure bootstrap.properties for security” on page 109.

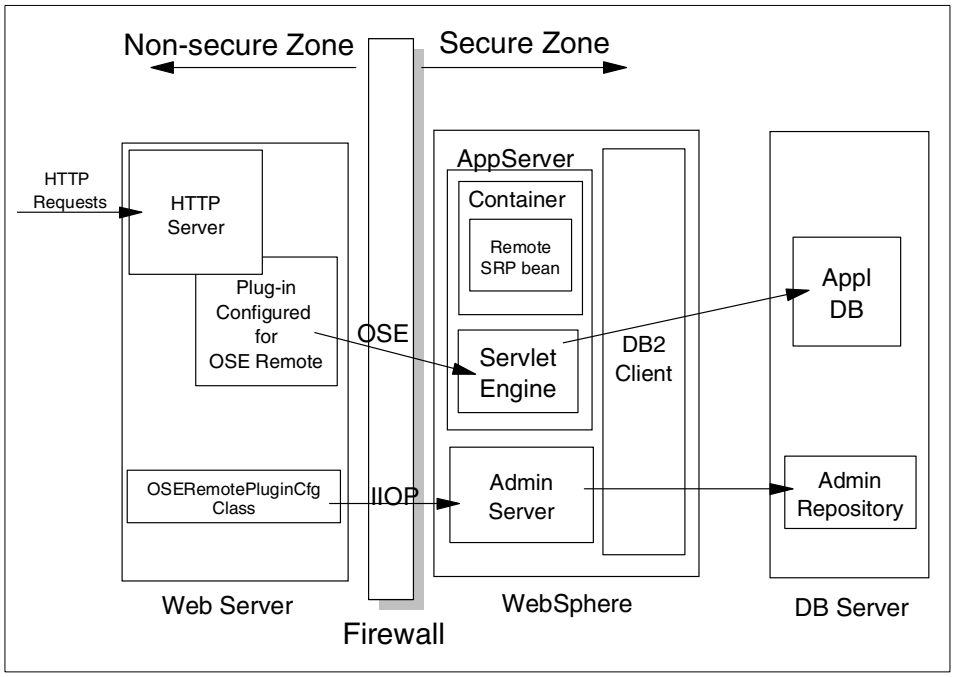

Figure 391. OSE Remote

If you want to use a script which includes the com.ibm.servlet.engine.oselistener.systemsmgmt.OSERemotePluginCfg class for OSE Remote configuration, you will be required to open ports for the IIOP traffic in the firewall. This configuration is not required if you configured OSE Remote by hand.

Note

You do not need to configure the plug-in files by hand if you do not want to run OSERemoteConfig from the Web server machine. You can run the script from the application server machine copy the files to the Web server machine. In this case, you don't need to open ports on the firewall for the IIOP trafic to run OSERemotePluginCfg.

For the IIOP traffic, you need to open ports for the following on the firewall:

1. Name Service Port: 900 (by default)

   In our example, we specify 900 as shown in Figure 67 on page 108

2. Location Service Daemon: 9000 (by default)

3. The CORBA Listener Port for the Administrative Server (no default value)

   In our example, we specify 33000 as described in , “CORBA Listener Port for the Administrative Server” on page 107.

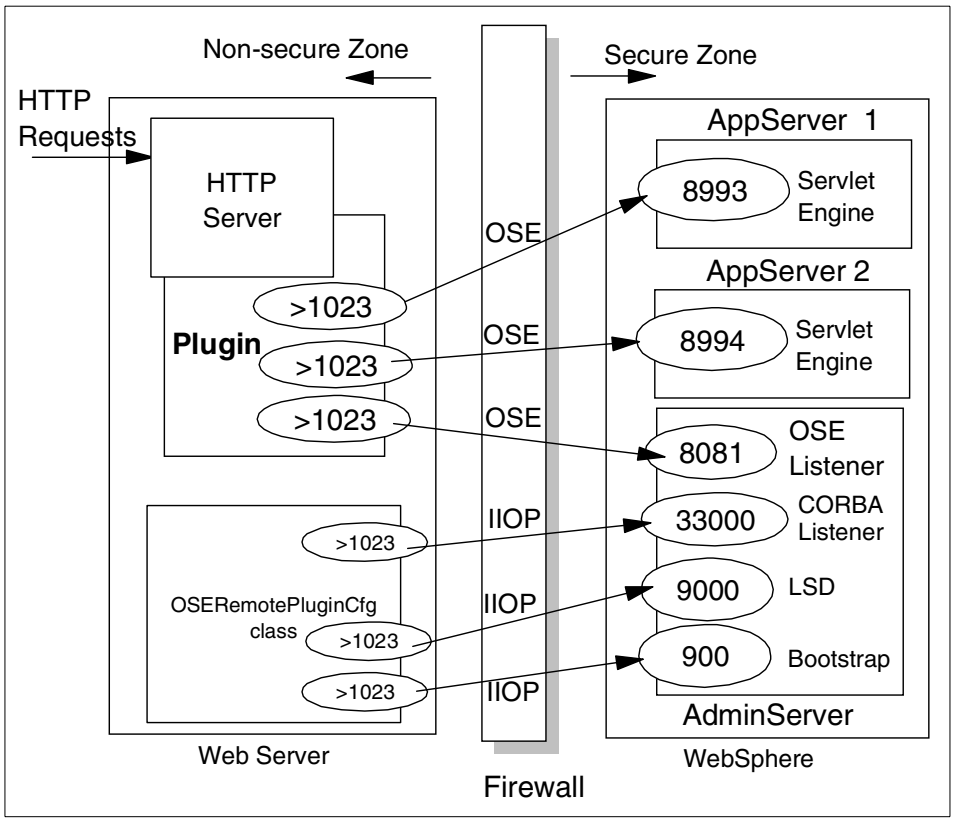

Figure 392. IIOP communication between the OSE plug-in configuration class and WebSphere

A.3.2 Servlet Redirector

A Servlet Redirector is a process which accepts requests from WebSphere's HTTP server plug-in and forwards them to servlets on machines remote to the Web server. The Servlet Redirector communicates with the plug-in via an internal protocol called OSE, and uses Remote Method Invocation (RMI) over the Internet Inter-ORB Protocol (IIOP) to distribute the servlet requests. The Servlet Redirector can be configured in three different ways, each with its own advantages and disadvantages.

A.3.3 Thick Servlet Redirector

In the thick Servlet Redirector configuration, the Web server machine runs a full instance of WebSphere Administrative Server, which shares an Administrative Repository with the WebSphere instance behind the firewall. In the WebSphere instance on the Web server machine, we configure a Servlet Redirector. This Servlet Redirector retrieves its configuration information from its local administrative server.

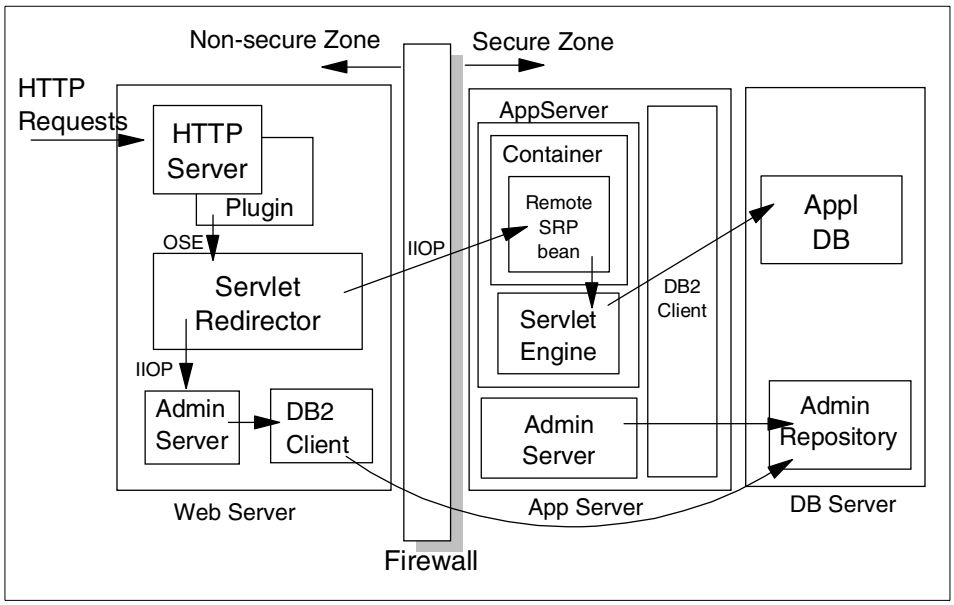

Figure 393. Thick Servlet Redirector

For the IIOP traffic, you need to open ports for the following on the firewall:

1. Location Service Daemon: 9000 (by default)
2. The CORBA Listener Port for the administrative server (no default value)

   In our example, we specify 33000.
3. The CORBA Listener Port for the application server (no default value)

   In our example. we specify 35000.
4. The CORBA Listener Port for the Servlet Redirector (no default value)

   In our example, we specify 40000.

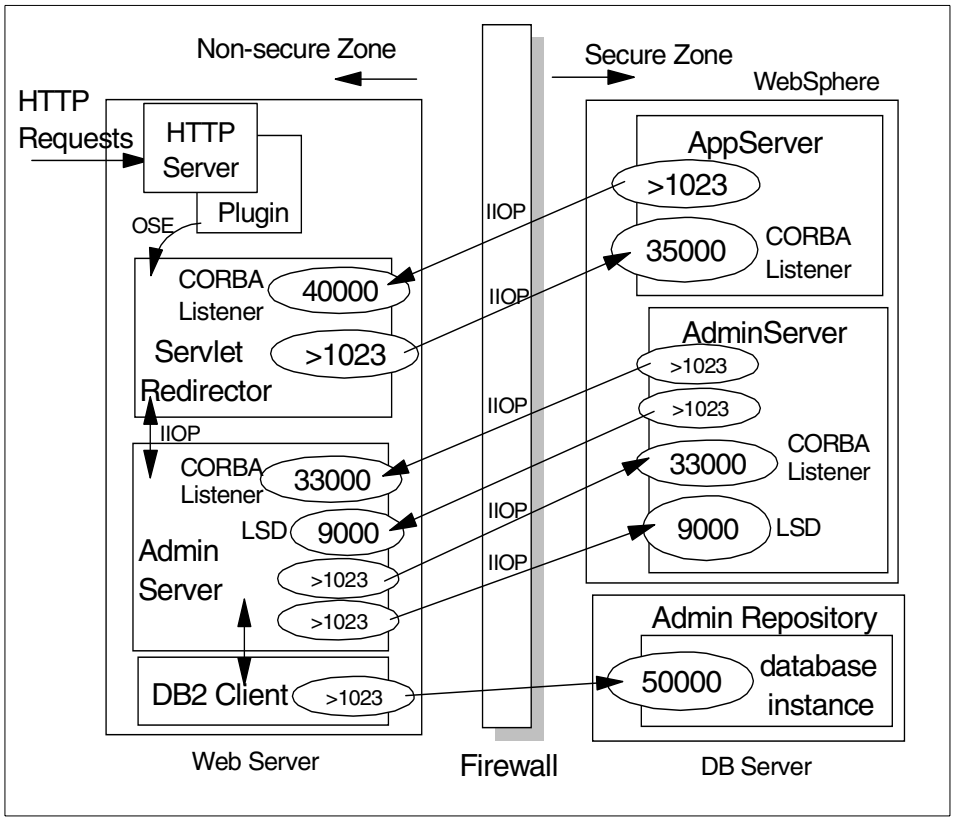

Figure 394. IIOP communication between the thick Servlet Redirector and WebSphere

To specify the CORBA Listener Port for the Administrative Server, you need to update <was_dir>/bin/admin.config as we described in 6.3, “Configure the Administrative Server CORBA listener port” on page 163.

To specify the CORBA Listener Port for the application server, you need to add the parameter in the command line arguments for the application server as we described in 6.5, “Add the CORBA listener port for the application server” on page 166.

The CORBA Listener Port for the Servlet Redirector is specified in the Command line args field of the Servlet Redirector as we described in 6.6.1, “Create the Servlet Redirector” on page 168.

A.3.4 Thin Servlet Redirector

In the thin Servlet Redirector configuration, the Web server machine runs a standalone version of the Servlet Redirector. This Servlet Redirector uses RMI/IIOP to communicate through the firewall with a WebSphere Administrative Server to retrieve its configuration information.

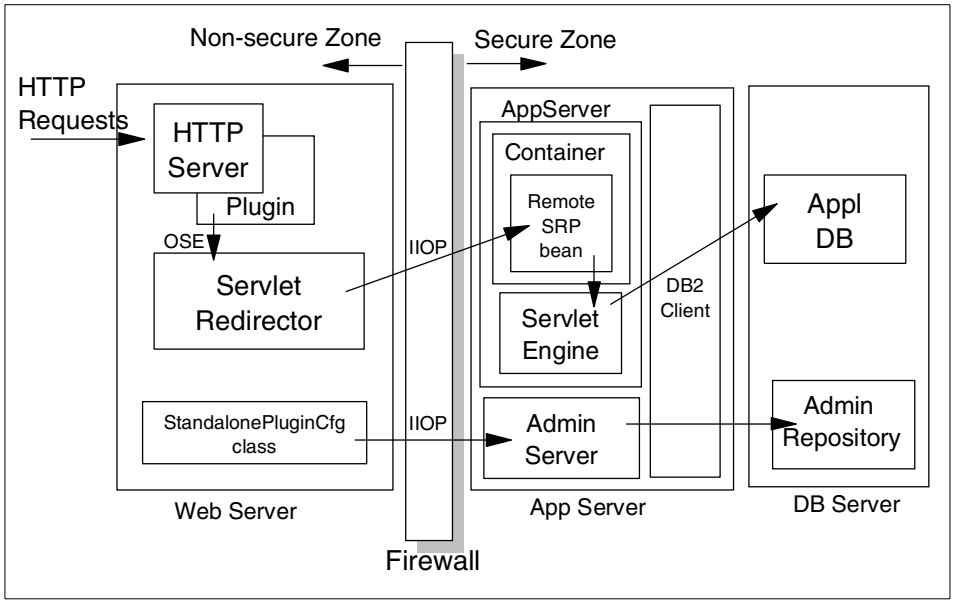

Figure 395. Thin Servlet Redirector

For the IIOP traffic, you need to open ports for the following on the firewall:

1. Bootstrap: 900 (by default)
2. Location Service Daemon: 9000 (by default)
3. The CORBA Listener Port for the Administrative Server (no default value)

   In our example, we specify 33000.
4. The CORBA Listener Port for the application Server (no default value)

   In our example. we specify 35000.
5. The CORBA Listener Port for the Servlet Redirector (no default value)

   In our example, we specify 40000.

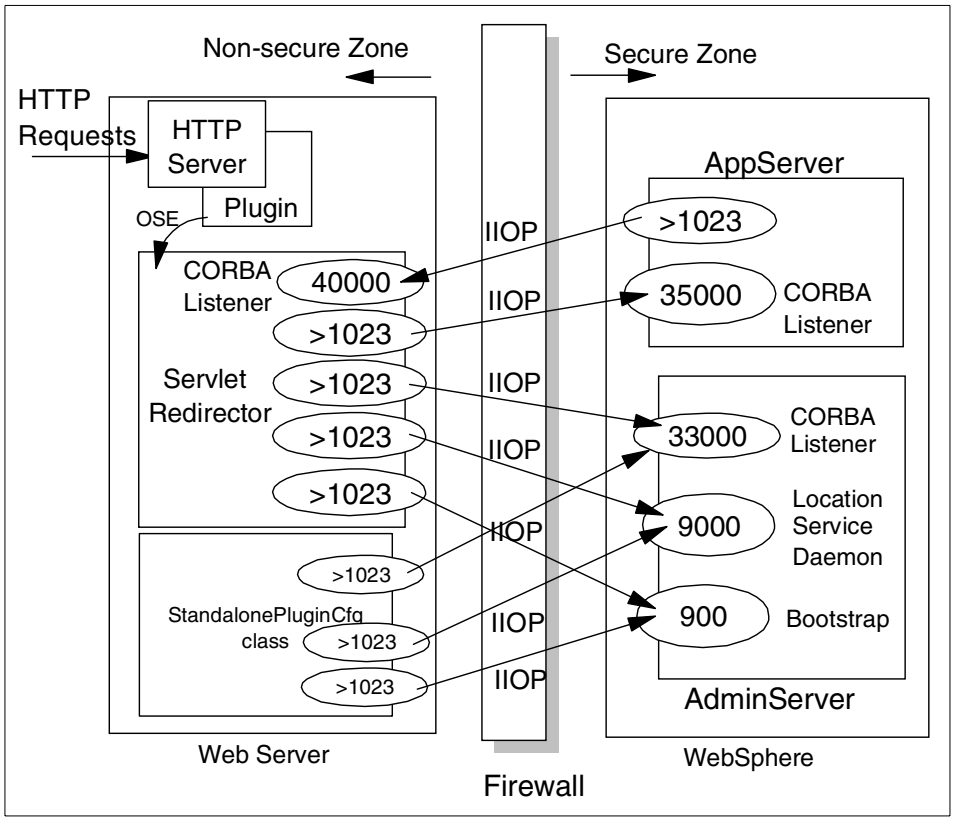

Figure 396. IIOP communication between the thin Servlet Redirector and WebSphere

To specify the CORBA Listener Port for the Administrative Server, you need to update <was_dir>/bin/admin.config as described in 8.3, “Configure the Administrative Server CORBA listener port” on page 203.

To specify the CORBA Listener Port for the application server, you need to add the parameter in the command line arguments for the application server as described in 8.5.2, “Add the CORBA listener port for the application server” on page 206.

The CORBA Listener Port for the Servlet Redirector is specified in the shell script (or bat file) as described in, “CORBA listener port” on page 219.

A.3.5 Admin agent thick Servlet Redirector

In the Admin agent Servlet Redirector configuration, the Web server machine runs an administrative-agent instead of a full Administrative Server. The

Administrative agent communicates via RMI/IIOP through the firewall with an Administrative Server on a backend machine to retrieve its configuration information. In the Administrative agent, we configure a Servlet Redirector. This Servlet Redirector retrieves its configuration information from the local Administrative agent.

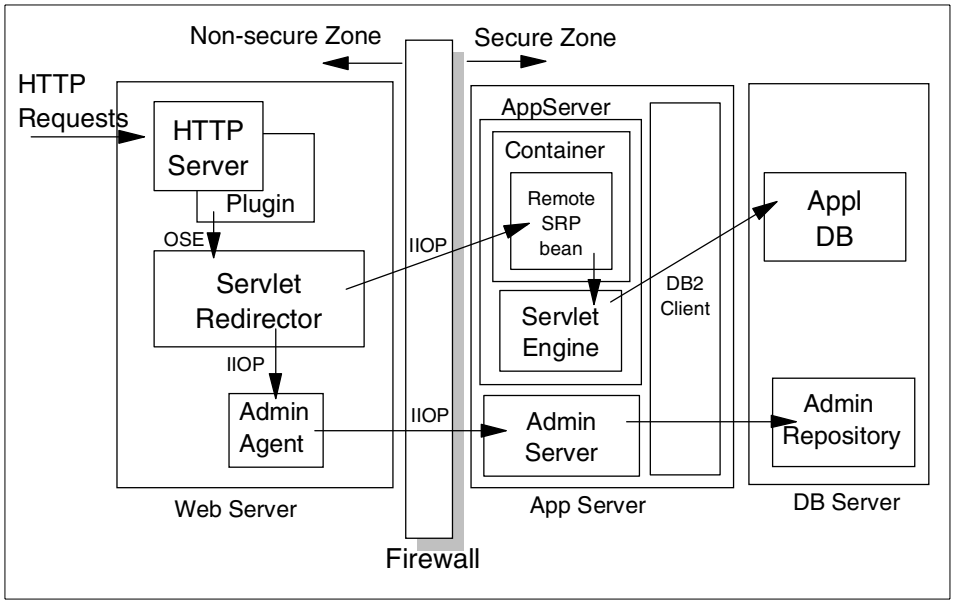

Figure 397. Servlet Redirector with Administrative agent

For the IIOP traffic, you need to open ports for the following on the firewall:

1. Bootstrap: 900 (by default)
2. Location Service Daemon: 9000 (by default)
3. The CORBA Listener Port for the Administrative Server (no default value)
   In our example, we specify 33000.
4. The CORBA Listener Port for the application server (no default value)
   In our example, we specify 35000.
5. The CORBA Listener Port for the Servlet Redirector (no default value)
   In our example, we specify 40000.

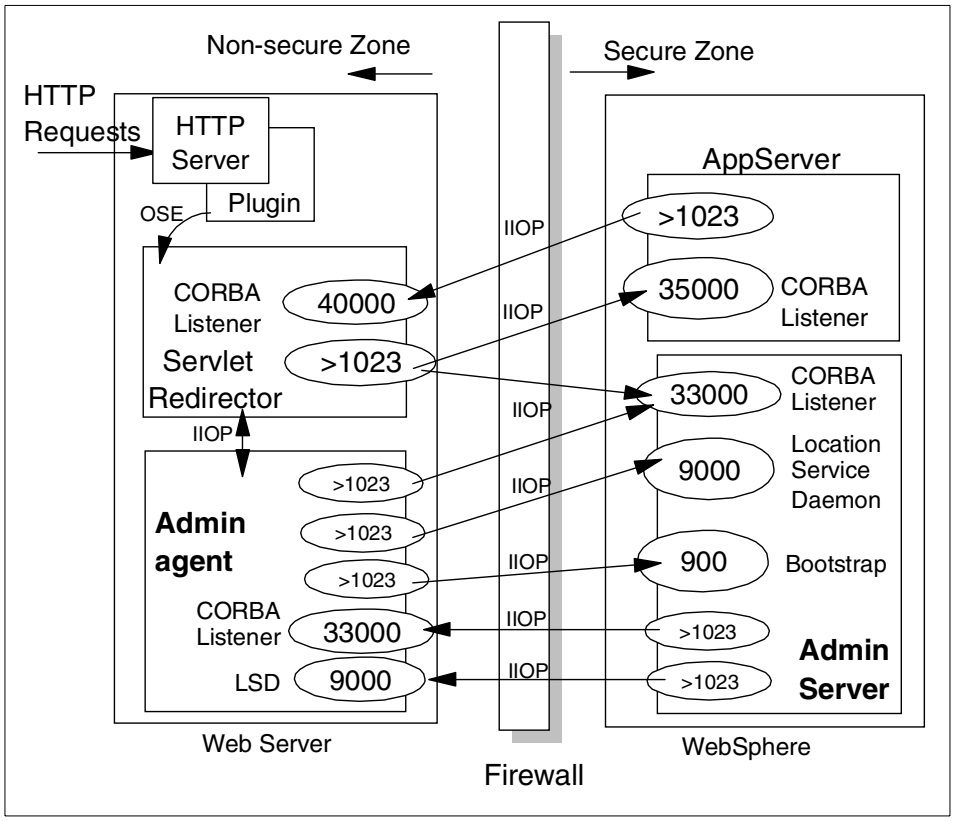

Figure 398. IIOP communication between the Admin agent Servlet Redirector and WebSphere

To specify the CORBA Listener Port for the Administrative Server, you need to update <was_dir>/bin/admin.config on both the Web server and WebSphere machine as described in 7.4, “Configure the Administrative Server CORBA listener port” on page 186.

To specify the CORBA Listener Port for the application server, you need to add the parameter in the command line arguments for the application server as we described in 7.6, “Adding the CORBA listener port for the application server” on page 188.

The CORBA Listener Port for the Servlet Redirector is specified in the command line args field of the Servlet Redirector as described in 7.7.1, “Create the Servlet Redirector” on page 190.

A.3.6 Reverse proxy/IP forwarding

No additional WebSphere administration is required when setting up a reverse proxy configuration; the implementation details are determined by the reverse proxy. In this configuration a reverse proxy that resides in the DMZ listens on port 80 for requests that have a certain format. It then forwards those requests to an HTTP server that resides on the same machine as WebSphere. The requests are then fulfilled and passed back through the reverse proxy to the client, hiding the originating Web server.

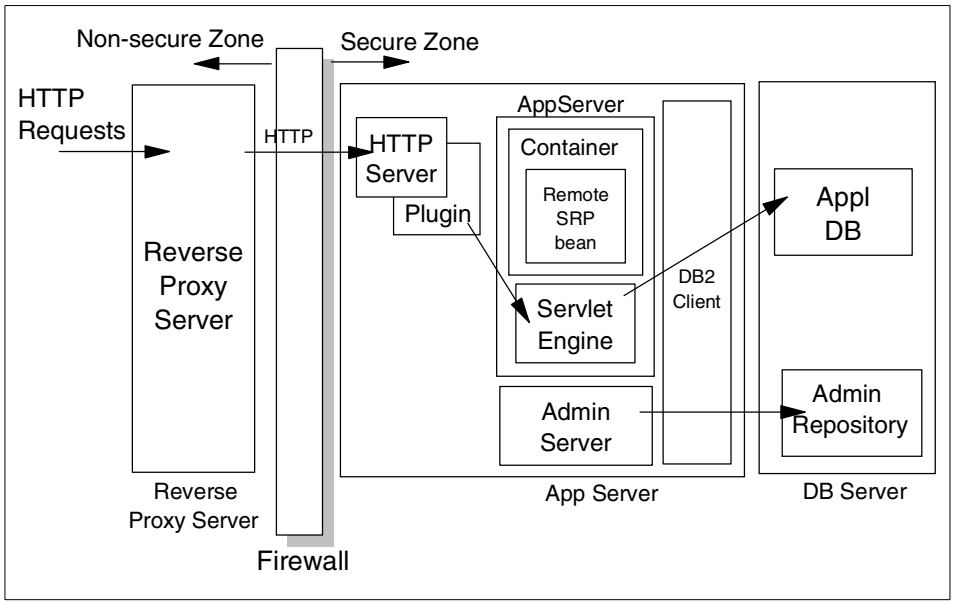

Figure 399. Reverse proxy/IP forwarding

A.3.7 Selection summary

Here is the summary of configuration options.

Configuration	WAS Security Support	No DB Access	NAT Support	Minimal Firewall Holes	DMZ Protocol Switch
OSE Remote	X	X	X	X	X
Thick Servlet Redirector	X				X

Configuration	WAS Security Support	No DB Access	NAT Support	Minimal Firewall Holes	DMZ Protocol Switch
Thin Servlet Redirector		X			X
Admin-Agent Servlet Redirector	X	X			X
Reverse Proxy	X	X	X	X	

A.4 IIOP tunneling

Because of the need to control Internet or intranet access to production WebSphere Application Server installations, many application servers will be placed behind a firewall. Such placement becomes problematic when attempting to flow a request from a client to the application server.

Typically, firewalls allow only HTTP requests to flow through them. Therefore, communication through a firewall requires a method for passing (tunneling) the requests through the firewall. The IIOP tunneling feature enables CORBA IIOP packets to pass through a firewall.

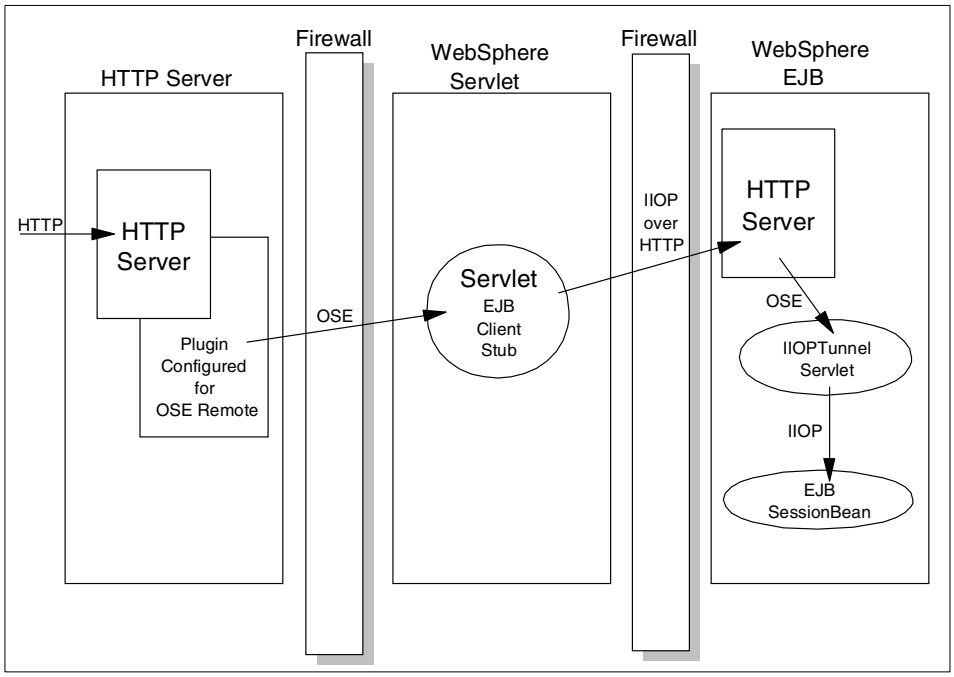

Figure 400. IIOP tunneling and firewalls

CORBA IIOP is the protocol used when enterprise bean remote objects are accessed between a client and an EJB server. The need for tunneling is dependent on the environment of your distributed application. The client may be an applet, Java application, or a servlet. In the scenario where the client contains enterprise bean (RMI/IIOP) remote objects and there is a firewall between the client and the EJB server, you will need to use the IIOP tunneling feature, unless your firewall allows IIOP requests to pass through.

The IIOP tunneling feature involves:

- The Object Request Broker (ORB) in the client wraps the IIOP packet into an HTTP request.
- The HTTP request's URL points to a tunneling servlet on the WebSphere Application Server.
- The tunneling servlet unwraps the IIOP packet from the HTTP request, establishes a socket connection to the target EJB server, and sends the IIOP request to the target EJB server.
- The servlet waits for the server's reply and sends the reply to the client.

There are some tradeoffs for using this feature. One tradeoff is performance. The HTTP protocol is not a guaranteed persistent connection, as is the case for a TCP (socket) connection. Therefore, a new HTTP connection is established for each request. This results in slower performance than sending IIOP packets over a persistent TCP connection. Some Web browsers and Web servers support a feature called keep alive, which attempts to keep the connection open longer than one request. However, the keep alive feature does not guarantee the length of time the connection will remain open.

A.4.1 Configuring IIOP tunneling

Figure 401 shows our IIOP tunneling test environment.

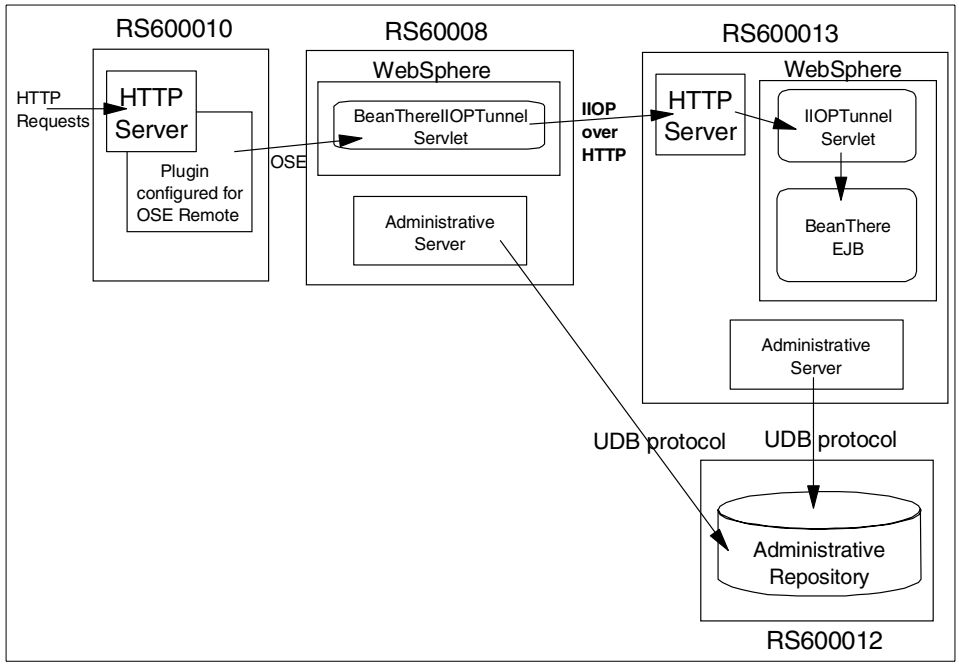

Figure 401. IIOP tunneling configuration

A.4.1.1 Modifying your servlet for IIOP tunneling

To configure tunneling, additional properties need to be set when the client creates the JNDI InitialContext. Below is a snippet of our sample code to show the properties needed for tunneling:

```
protected beanThere.BeanThereHome getHome() throws
javax.naming.NamingException, java.rmi.RemoteException {
   if (beanThereHome == null) {
      javax.naming.Context ctx = new javax.naming.InitialContext();
      beanThereHome =
(beanThere.BeanThereHome)javax.rmi.PortableRemoteObject.narrow(ctx.look
up("BeanThere"),
         beanThere.BeanThereHome.class);
   }

   return beanThereHome;
}
```

Figure 402. IIOP tunneling sample program (before modification)

```
protected beanThere.BeanThereHome getHome() throws
javax.naming.NamingException, java.rmi.RemoteException {
   if (beanThereHome == null) {
           // For IIOP tunneling

      Properties ppty = new Properties();

      ppty.put("com.ibm.CORBA.ForceTunnel", "always");
      ppty.put("com.ibm.CORBA.TunnelAgentURL",
"http://9.24.104.119/IIOPTunnel");
      ppty.put("com.ibm.CORBA.BootstrapHost", "9.24.104.119");
      ppty.put(javax.naming.Context.PROVIDER_URL,
"iiop://9.24.104.119");
      ppty.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.ejs.ns.jndi.CNInitialContextFactory");

      javax.naming.Context ctx = new javax.naming.InitialContext(ppty);
      beanThereHome =
(beanThere.BeanThereHome)javax.rmi.PortableRemoteObject.narrow(ctx.look
up("BeanThere"),
         beanThere.BeanThereHome.class);
   }

   return beanThereHome;
}
```

Figure 403. IIOP tunneling sample program (after modification)

com.ibm.CORBA.ForceTunnel=value
where value could be set to one of the following values:

a. whenrequired: This is the default value. The ORB will attempt a TCP connection directly to the target object server. If that succeeds, tunneling will not be attempted. If the TCP connection fails, the ORB will attempt to connect to the servlet using HTTP. If the HTTP connection fails, an org.omg.CORBA.COMM_FAILURE system exception is thrown to the client.

b. never: This value tells the client-side ORB to never attempt an HTTP connection. This value disables the tunneling feature.

c. always: This value directs the client-side ORB to attempt only HTTP connections. IIOP connections are not attempted.

com.ibm.CORBA.TunnelAgentURL=URL_string
where URL_string specifies the URL where the tunneling servlet resides.

com.ibm.CORBA.BootstrapHost=Server_host_name
where Server_host_name specifies the host name of the remote bootstrap server. This property is not related specifically to tunneling, but it is needed to tell the client ORB which server the tunneling servlet should go to for the JNDI InitialContext. This server might not be the same server that contains the tunneling servlet.

See the online manual for more detailed information.

A.4.1.2 Deploying your servlet and client EJB stub

After you modify your servlet to use IIOP tunneling, you need to deploy the servlet and your EJB client stub on the servlet machine.

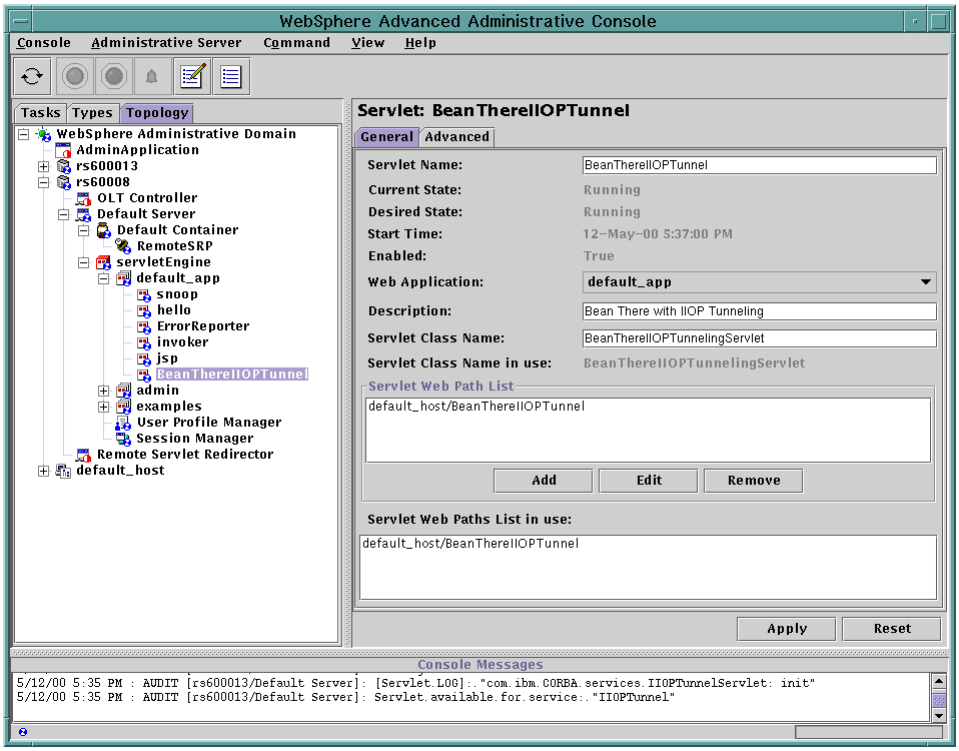

Figure 404. Add your servlet to your servlet machine

A.4.1.3 Adding the IIOPTunnel servlet to the EJB machine

The best way to enable the tunneling servlet named IIOPTunnelServlet.class is to add the servlet to a Web application.

Use the Administrative Console to create a servlet for the Web application. Specify a name for the servlet, such as IIOPTunnel. For the servlet class name, specify the full package name (com.ibm.CORBA.services.IIOPTunnelServlet).

If you named the servlet IIOPTunnel, your com.ibm.CORBA.TunnelAgentURL property would be:

```
http://host:port/servlet/IIOPTunnel
```

The following is our sample configuration. In our case, com.ibm.CORBA.TunnelAgentURL property is:

```
http://9.24.104.119/IIOPTunnel
```

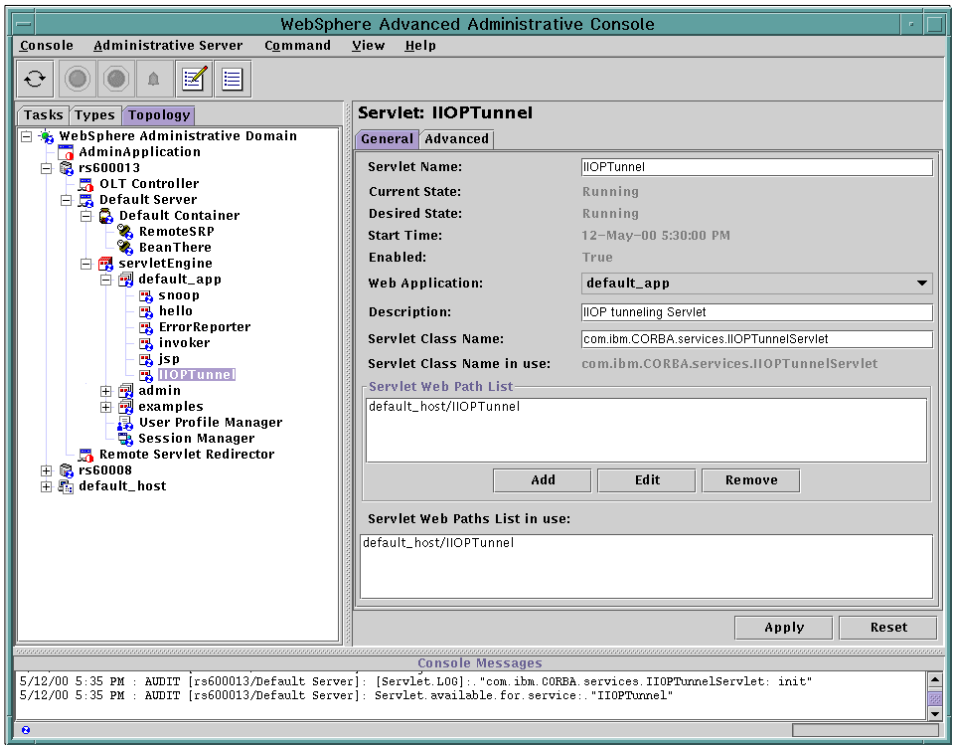

Figure 405. Add IIOPTunnel servlet on EJB machine

A.4.1.4 Deploying your EJB

You deploy your EJBs as normal EJBs. There is nothing special for IIOP tunneling.

A.4.1.5 Configure OSE Remote

You now configure OSE Remote to run the HTTP server and servlet machine on separate machines. See Chapter 5, “OSE Remote” on page 95 for more information.

A.4.1.6 Start servers

You need to start all servers before testing.

On the EJB server machine (rs600013):

1. Start application server (IIOPTunnelServlet and EJBs)
2. Start the HTTP server

On the servlet machine (rs60008):

- Start application server (Servlet)

On the HTTP server machine (rs600010):

- Start the HTTP server

A.4.1.7 Testing the IIOP tunneling

Now, you are ready to test your IIOP tunneling configuration.

The following is the result of our IIOP tunneling test.

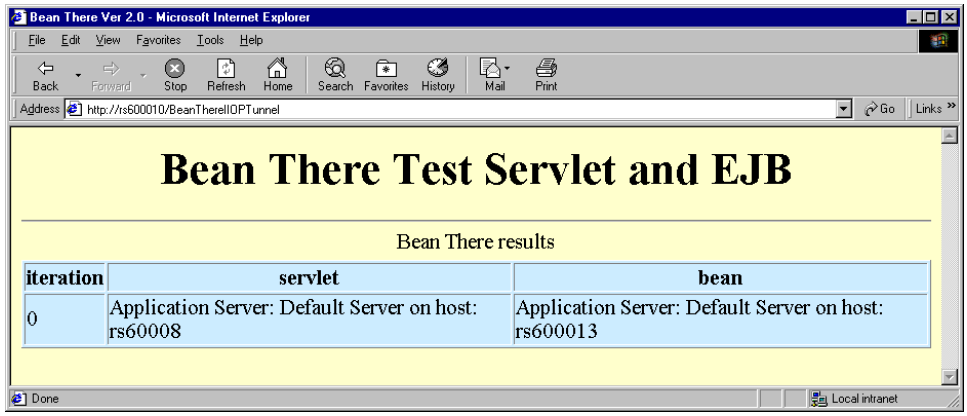

Figure 406. Output on the browser

This is the output of an IP network trace.

As you can see, the HTTP protocol was used to forward the request from the servlet machine (rs60008; 9.24.104.30) to the EJB machine (rs600013; 9.24.104.119). And also, the second packet shows you that the IIOP packet is encapsulated in the HTTP packet. This is called IIOP over HTTP.

```
[ src = 00:20:35:7c:6f:6e, dst = 00:20:35:7c:cb:f3]
802.2 LLC header:
dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP header breakdown:
   < SRC =     9.24.104.30 >
   < DST =    9.24.104.119 >
   ip_v=4, ip_hl=20, ip_tos=0, ip_len=308, ip_id=58227, ip_off=0
   ip_ttl=60, ip_sum=b78b, ip_p = 6 (TCP)
TCP header breakdown:
   <source port=32951, destination port=80(www) >
   th_seq=85d09ac6, th_ack=6e6a40ba
   th_off=5, flags<PUSH | ACK>
   th_win=15972, th_sum=ff43, th_urp=0
00000000     504f5354 202f4949 4f505475 6e6e656c     |POST /IIOPTunnel|
00000010     3f686f73 743d392e 32342e31 30342e31     |?host=9.24.104.1|
00000020     31392670 6f72743d 39303026 6f6e6577     |19&port=900&onew|
00000030     61793d66 616c7365 20485454 502f312e     |ay=false HTTP/1.|
00000040     300d0a55 7365722d 4167656e 743a204a     |0..User-Agent: J|
00000050     61766131 2e312e38 0d0a486f 73743a20     |ava1.1.8..Host: |
00000060     392e3234 2e313034 2e313139 0d0a4163     |9.24.104.119..Ac|
00000070     63657074 3a207465 78742f68 746d6c2c     |cept: text/html,|
00000080     20696d61 67652f67 69662c20 696d6167     | image/gif, imag|
00000090     652f6a70 65672c20 2a3b2071 3d2e322c     |e/jpeg, *; q=.2,|
000000a0     202a2f2a 3b20713d 2e320d0a 436f6e6e     | */*; q=.2..Conn|
000000b0     65637469 6f6e3a20 6b656570 2d616c69     |ection: keep-ali|
000000c0     76650d0a 436f6e74 656e742d 74797065     |ve..Content-type|
000000d0     3a206170 706c6963 6174696f 6e2f782d     |: application/x-|
000000e0     7777772d 666f726d 2d75726c 656e636f     |www-form-urlenco|
000000f0     6465640d 0a436f6e 74656e74 2d6c656e     |ded..Content-len|
00000100     6774683a 20323736 0d0a0d0a              |gth: 276....    |
```

Figure 407. Packet from the servlet machine to the EJB machine (1/2)

```
802.2 LLC header:
dsap aa, ssap aa, ctrl 3, proto 0:0:0, type 800 (IP)
IP header breakdown:
   < SRC =     9.24.104.30 >
   < DST =    9.24.104.119 >
   ip_v=4, ip_hl=20, ip_tos=0, ip_len=316, ip_id=58228, ip_off=0
   ip_ttl=60, ip_sum=b782, ip_p = 6 (TCP)
TCP header breakdown:
   <source port=32951, destination port=80(www) >
   th_seq=85d09bd2, th_ack=6e6a40ba
   th_off=5, flags<PUSH | ACK>
   th_win=15972, th_sum=6cdf, th_urp=0
00000000     47494f50 01000000 00000108 00000002     |GIOP............|
00000010     00000006 000000a4 00000000 00000028     |...............(|
00000020     49444c3a 6f6d672e 6f72672f 53656e64     |IDL:omg.org/Send|
00000030     696e6743 6f6e7465 78742f43 6f646542     |ingContext/CodeB|
00000040     6173653a 312e3000 00000001 00000000     |ase:1.0.........|
00000050     00000068 00010100 0000000c 392e3234     |...h........9.24|
00000060     2e313034 2e333000 80ad0000 0000002c     |.104.30........,|
00000070     4a4d4249 00000010 42f65a47 63313030     |JMBI....B.ZGc100|
00000080     30303030 30303030 30303030 00000024     |000000000000...$|
00000090     00000008 00000000 00000000 00000001     |................|
000000a0     00000001 00000014 00000000 00010001     |................|
000000b0     00000000 00010100 00000000 49424d12     |............IBM.|
000000c0     00000008 00000000 00000000 00000005     |................|
000000d0     01000000 00000004 494e4954 00000004     |........INIT....|
000000e0     67657400 0000001c 49424d44 3a000000     |get.....IBMD:...|
000000f0     0000000c 392e3234 2e313034 2e333000     |....9.24.104.30.|
00000100     00000000 0000000c 4e616d65 53657276     |........NameServ|
00000110     69636500                                |ice.            |
```

Figure 408. The packet from the servlet machine to the EJB machine (2/2)

Appendix B. WLM enabling EJBs

In this appendix, we describe work load management for EJBs. We will discuss the following subjects:

- Overview: enabling workload management for enterprise beans
- How to set up and run wlmjar
- Output from running wlmjar
- Command syntax and arguments for wlmjar
- Troubleshooting wlmjar
- Example and discussion

B.1 Overview: enabling workload management for enterprise beans

Enabling workload management for EJBs (WLM) requires two steps:

1. Create an application server model and clones.
2. Create a client side JAR file containing WLM-enabled stubs, as documented below.

After you have deployed an EJB JAR file that contains one or more enterprise beans used by EJB clients, you must create a workload-managed version of that JAR file so that the EJB clients that access the enterprise beans do so through the workload management service.

The workload management service improves application server availability, ensures that client requests are distributed evenly over the available servers, and ensures that transaction affinity is maintained.

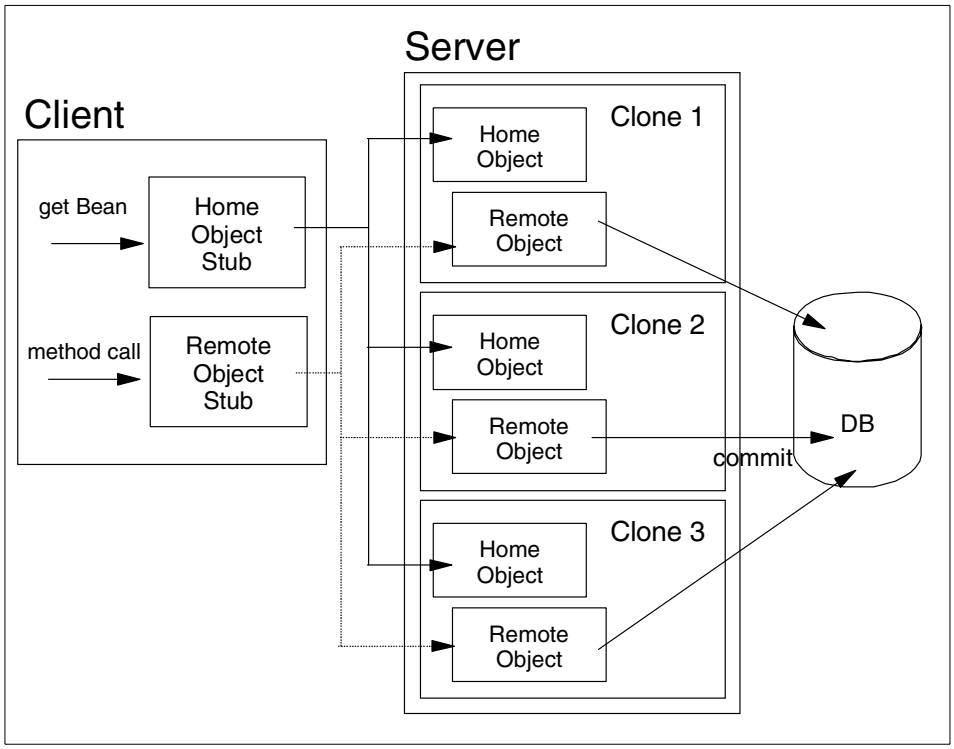

Figure 409. WLM for EJBs

EJB JAR files that are deployed in VisualAge for Java are automatically enabled for workload management, and you do not need to perform this task on those JAR files.

Note: WebSphere 3.5

With WebSphere 3.5 Administrative Console, when you create enterprise beans which are not deployed in VisualAge for Java, you will be asked if you want to make them WLM-enabled JARs or not.

B.2 How to set up and run the wlmjar command

To create a workload-managed JAR file, you must process the deployed JAR by using the `wlmjar` shell command. Follow these steps to set up and run the command:

1. Ensure the WAS_HOME and JAVA_HOME environment variables are defined correctly.
   - WAS_HOME refers to the root directory of your product installation (for example, D:\WebSphere\AppServer\ on Windows).
   - JAVA_HOME refers to the root directory of the JDK (for example: D:\jdk1.1.n on Windows). See the *Getting Started* book for a list of supported JDK levels.
2. Change to deployedEJBs subdirectory of WebSphere Application Server installation, where your deployed JAR file resides.
3. Issue the command:

```
wlmjar -J DeployedInc.jar -IF Inc -P com.transarc.jmon.examples.Inc
-V
```

where DeployedInc.jar is the name of your deployed JAR file, Inc is the name of your enterprise bean, and the -P is the package name for your enterprise bean code.

On AIX and Solaris, type `wlmjar.sh` instead of `wlmjar`.

B.3 Output from running wlmjar

The output of this command is a file named _wlm_deployedJARFile, which is the workload-managed JAR file.

Note: WebSphere 3.5

If you make the WLM-enabled JAR when you create enterprise beans with WebSphere 3.5 Administrative Console, the file named _wlm_deployedJARFile, which is the workload-managed JAR file, will be created.

If you specified the `-M` option with the command, you must overwrite the original deployed JAR file with the new workload-managed JAR file.

Every EJB client that needs to access one or more of the enterprise beans contained in the workload-managed JAR file must identify the location of that JAR file in the client's CLASSPATH.

B.4 Command syntax and arguments for wlmjar

Run the `wlmjar` command as shown here:

```
# wlmjar -J deployedJARFile -IF remoteInterface -IF homeInterface -P
javaPackage -M -V
```

On AIX and Solaris, type `wlmjar.sh` instead of `wlmjar`.

Run the command in the directory containing the deployed JAR file containing the enterprise bean that you want to enable for workload management. Run the command one time for each Java package in the deployed JAR file.

In the command, the arguments are as follows (you can also type `wlmjar` at a prompt to get a list):

- -J deployedJARFile: Identifies the deployed JAR file to be enabled for workload management.
- -P javaPackage: Identifies the Java package of which each target enterprise bean is a part. All of the remote interfaces specified using the -IF parameter must belong to this package.
- -IF remoteInterface: Identifies the remote interface of each enterprise bean in the deployed JAR file contained in the Java package identified by the -P option. The remote interface can be the bean or home interface.

  Both the option and the argument must be specified at least one time. The most common usage is to specify them multiple times, once for each of the remote bean and home interfaces in the JAR file.
- -M: Reuse manifest. Directs the command to create one workload-managed JAR file that can be used by both an EJB server and an EJB client. It is strongly recommended that this option be used for deployed JAR files.
  - If you use this option, you can later overwrite the original JAR file with the new workload-managed JAR file. The new file can be used by clients and servers.
  - If you do not use this option, you must ensure that the EJB server uses the original deployed JAR file and the EJB client uses the workload-managed JAR file.

This option MUST be used if the server in which beans from the jar file are installed is itself a client of any of those beans.

- -V: Directs the command to provide verbose information about what it is going on during JAR file processing. This option is not required.

B.5 Troubleshooting wlmjar

If you encounter problems with wlmjar, it is likely that the tool cannot find the files it needs. The wlmjar program uses the JAVA_HOME and WAS_HOME environment variables to locate these files:

- ujc.jar
- iioprt.jar
- iioptools.jar

The files reside in <was_dir>/lib where was_dir is the root directory of your application server product installation and should match the WAS_HOME value.

Also, ensure that the WebSphere Application Server files precede the JDK files in your CLASSPATH environment variable.

Finally, check that your PATH environment variable points to the directory containing the `rmic` command that comes with the Advanced Application Server (installed in <install_root>/bin).

Ensure that one of these two cases is true:

- Either your PATH environment variable does not point to the directory containing the `rmic` command that comes with the Java Development Kit.
- Or, in your PATH environment variable, the directory that contains the `rmic` command that comes with the Advanced Edition Application Server precedes the directory that contains the `rmic` command that comes with the JDK.

(For example, on Windows: set PATH=C:\WebSphere\AppServer\bin;C:\JDK\bin;...)

rmic

In JDK 1.1.*, the rmic shipped with the JDK did not have the -iiop option for generating IIOP Stubs (rather than RMI Stubs). A separated rmic was shipped with Websphere which had this capability , and this was the rmic needed by wlmjar. In JDK 1.2.*, the rmic in the JDK does support the -iiop option, so it can be used.

B.6 Example and discussion

Multiple beans may reside in the JAR file. Generally, all need to be workload managed. These two cases are treated differently:

- A bean on the server accesses another bean on the server (that is, the server is its own client).
- Or, in contrast, the only accesses to the beans are from external clients.

This section walks through an example that illustrates some related wlmjar considerations.

Consider a small accounting application. The application uses three enterprise beans to do its work:

- The Account entity bean provides persistence for accounts.
- The AccountFinder session bean provides a query capability for obtaining indirect access to Account entity beans.
- The AccountManager session bean provides an interface to manipulate one or more Account beans.

Each type of bean also has a corresponding home interface, called AccountHome, AccountFinderHome, and AccountManagerHome, respectively. All the beans exist in the Account package and are packaged in the Account.jar file.

If a bean on the server accesses another bean on the server, as in the above example (AccountManager accesses Account), the Account.jar file must contain the deployed code before running wlmjar. If the only accesses to the beans are through external clients, the pre-deployed JAR file may be used.

Here, the client directly accesses the AccountFinder and AccountManager beans. The AccountManager bean directly accesses the Account beans, which are indirectly accessed by the client.

To obtain the benefits of workload management, you would run wlmjar on each of the beans' local and remote interfaces as follows: `wlmjar -p Account -if Account -if AccountFinder -if AccountManager -if AccountHome -if AccountFinderHome -if AccountManagerHome -j Account.jar -m`

On AIX and Solaris, you would type `wlmjar.sh` instead of `wlmjar`.

The -m (reuse manifest) option needs to be specified since one of the beans on the server (AccountManager) is the client of another bean on the server (Account). You can omit it if only external clients access the beans.

You would then delete the old Account.jar file, and rename the newly generated _wlm_Account.jar file to Account.jar. You would add it to the classpath for the client code accessing the beans, and deploy it within the server (in place of the original Account.jar file).

The old file must be overwritten, since one of the beans on the server (AccountManager) is the client of another bean on the server (Account). If only external clients access the beans, the server can reuse the old Account.jar, while the new _wlm_Account.jar should be placed in the classpath of the clients.

Appendix C. WLM tuning properties

While in general it is not necessary, it is possible to tune some properties that can affect the behavior of the WLM runtime. These values should ONLY be set in order to tune the WLM runtime in response to problems that you might encounter in your environment. If WLM is functioning correctly there is nothing to be gained from setting these properties and in fact may produce undesirable results.

C.1 Client Properties

The following properties are specific to the WLM client runtime. Note that the Servlet Redirector is a WLM-enabled EJB client. As such they are set as a command line argument for the JVM that the WLM client is running in. In many cases, such as where a servlet is a client to an EJB, this means that these are specified as part of the command line arguments for the application server where the servlet is running.

-Dcom.ibm.CORBA.requestTimeout

-Dcom.ibm.ejs.wlm.MaxCommFailures

-Dcom.ibm.ejs.wlm.UnusableInterval

C.1.1 -Dcom.ibm.CORBA.requestTimeout

You can set this value in the Command Line Arguments field, for example:

-Dcom.ibm.CORBA.requestTimeout=30

which specifies that the request to timeout after 30 seconds. If your network is subject to extreme latency it might be desirable to specify a fairly large number here to prevent timeouts. Note if the value which you specify is too small, your EJB client application will timeout before it receives the response from the EJB server application or backend servers. Please be very careful when you specify this parameter, there is no recommended value for this property, and it should only be set if your application is experiencing timeouts. This parameter applies to both V3.02x and V3.5.

C.1.2 -Dcom.ibm.ejs.wlm.MaxCommFailures

You can set this value in the Command Line Arguments field, for example:

-Dcom.ibm.ejs.wlm.MaxCommFailures=0

The wlm client runtime will not mark a server as bad unless a certain number of attempts to access it have failed. This is to allow for the possibility of transient errors. This number is governed by the property com.ibm.ejs.wlm.maxFailures. The default is 0 in V3.5 and is 2 in V3.02. Thus, in V3.02, the client will attempt to access the server until 2 attempts fail, and sometimes these failures can be propagated to the application. If this value is set to 0, the wlm runtime will not attempt to use the server after the first failure, and this will eliminate the possibility of further failures to the server.

C.1.3 -Dcom.ibm.ejs.wlm.UnusableInterval

You can set this value in the Command Line Arguments field, for example:

-Dcom.ibm.ejs.wlm.UnusableInterval=900

After the wlm client runtime has marked a server as bad, it will not attempt to reuse it for the time interval specified by this property. The default for this property is 900 seconds in V3.5 and 300 seconds in V3.02. If this is set to a large value, the server will be marked bad for a long period of time. This precludes the wlm refresh protocol from refreshing the WLM state on the client before this time runs out, when it would have otherwise done so.

C.2 Server properties

The following property is either a client or server property, depending on the version of WebSphere you are running. As such the mechanism for setting it will vary.

C.2.1 -Dcom.ibm.ejs.wlm.RefreshInterval

The value for this property is specified in the following form:

-Dcom.ibm.ejs.wlm.RefreshInterval=xxx

Where xxx is the number of seconds that elapse between the Adminserver updates of the model information to the application servers. In V3.5 this information is a "push" (a message from the adminserver to the application server). As a result this property is appended to the arguments for the com.ibm.ejs.sm.util.process.Nanny.adminServerJvmArgs entry in this file. In versions less than 3.5, such as V3.02.x, this value was propagated to the application servers based on a "pull architecture" (request from the application server to the adminserver). As such it is set as a command line

argument for each application server. The default value for this property is 300 (seconds).

Appendix D. Basic WebSphere operation

In this is appendix, we will discuss some basic WebSphere operations:

1. Definition and configuration of an application server
2. Configuration of a servlet engine
3. Configuration of a Web application
4. Servlet configuration
5. Consideration of multiple HTTP servers

D.1 Definition and configuration of an application server

An application server is defined from the Topology tab of the Administrative Console. Select the node the application server will reside on. Right click it and select **Create->Application Server** as shown in Figure 410.

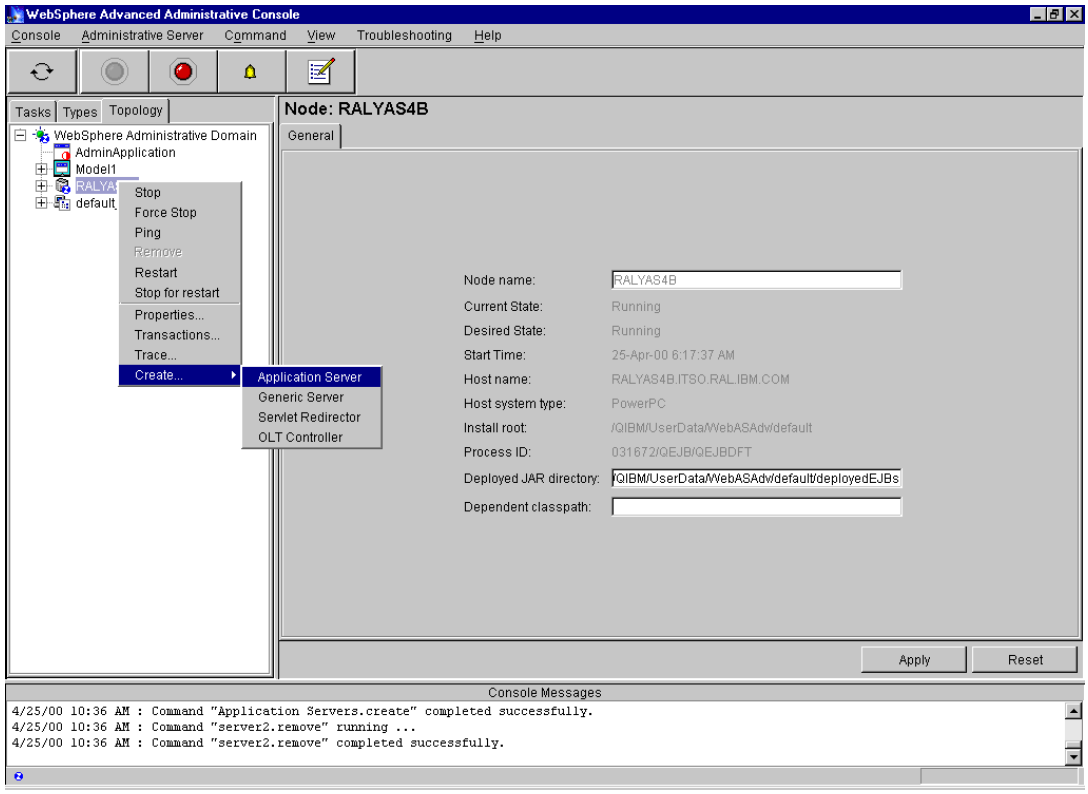

Figure 410. Create application server

The Application Server Properties window will appear. Enter a name for the server as shown in Figure 411 and click the **Create** button.

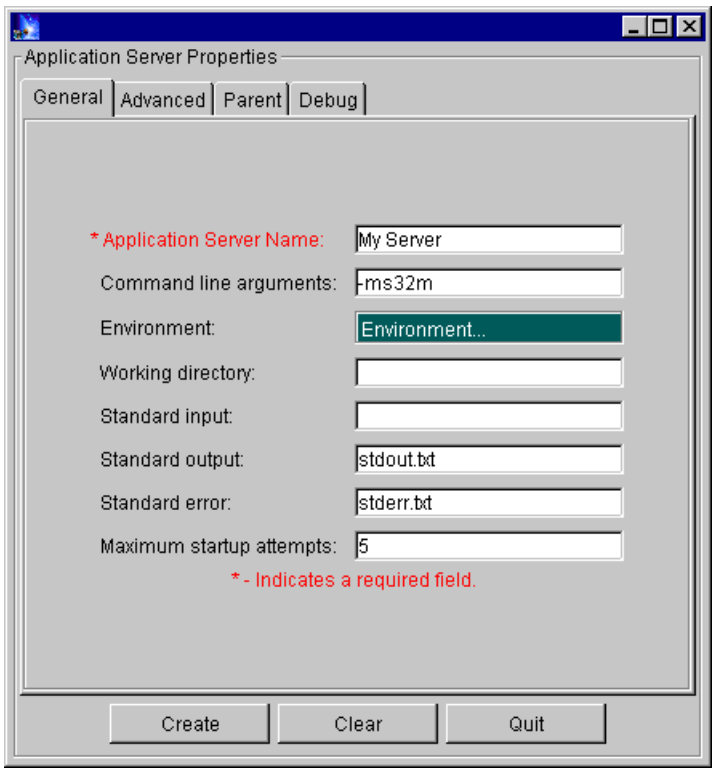

Figure 411. Application Server Properties

When the server is created you will see the following message.

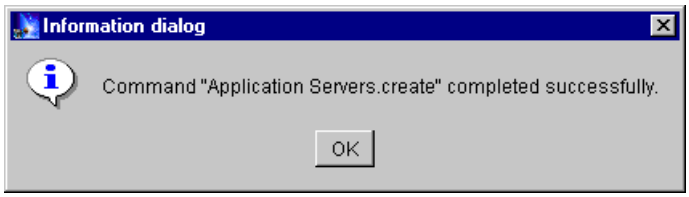

Figure 412. Application server created message

The application server will now appear in the Topology tab.

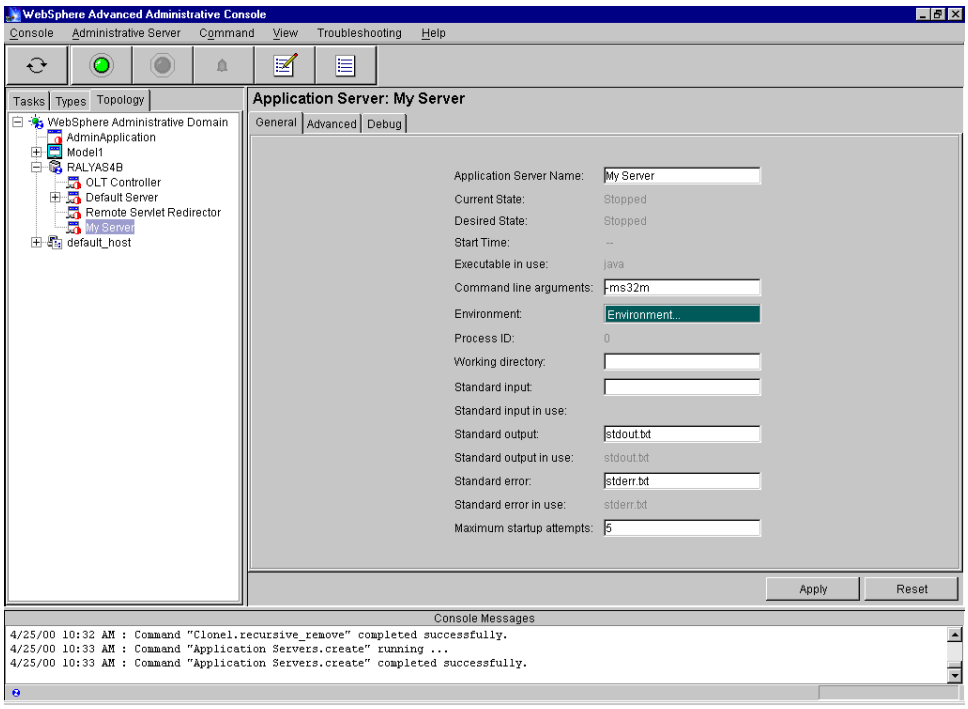

Figure 413. Topology with application server

D.2 Configuration of a servlet engine

A servlet engine is also defined from the Topology tab of the Administrative Console. Select the application server the servlet engine will reside on. Right click it and select **Create->Servlet Engine** as shown in Figure 414.

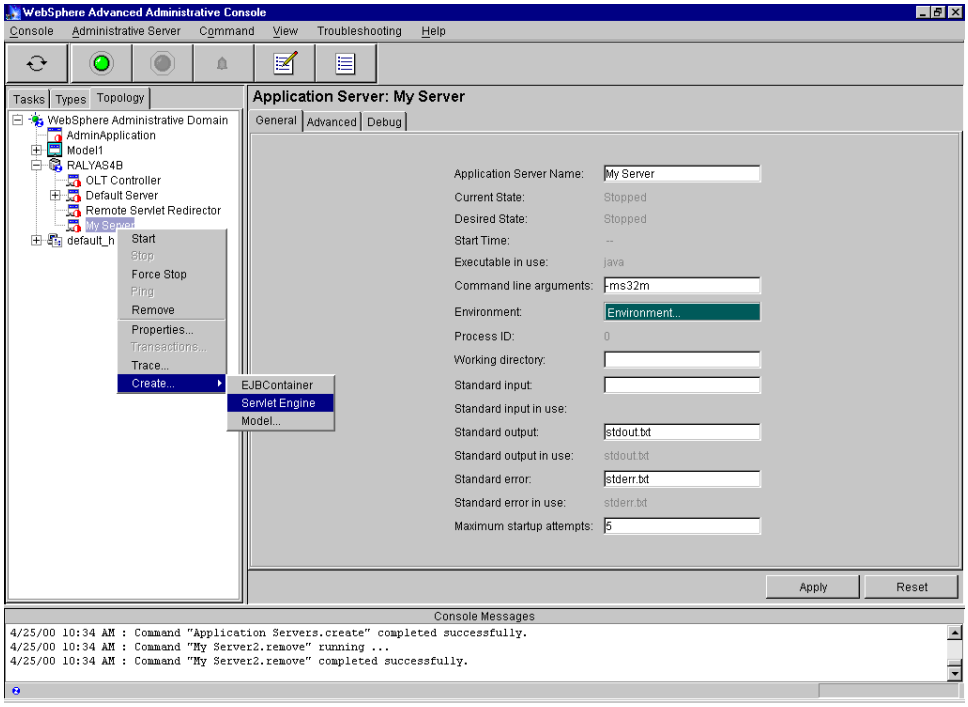

Figure 414. Create servlet engine

The Create Servlet Engine window will appear. Enter a name for the servlet engine as shown in Figure 415 and click the **Create** button.

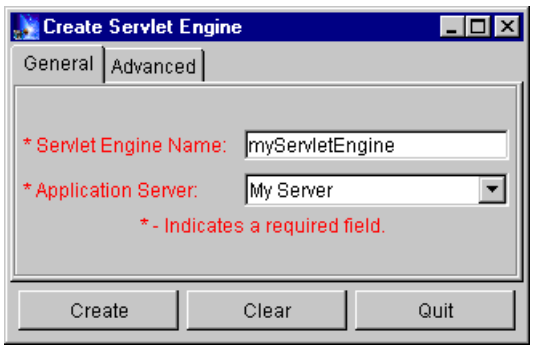

Figure 415. Create Servlet Engine

When the servlet engine has been created, you will see the following message.

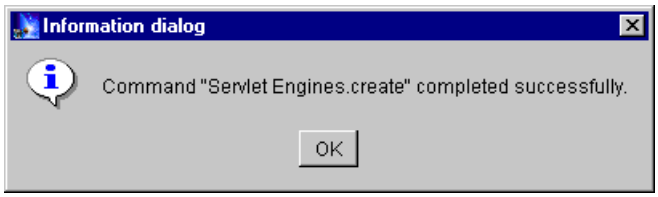

Figure 416. Servlet engine created message

The servlet engine will now appear in the Topology tab.

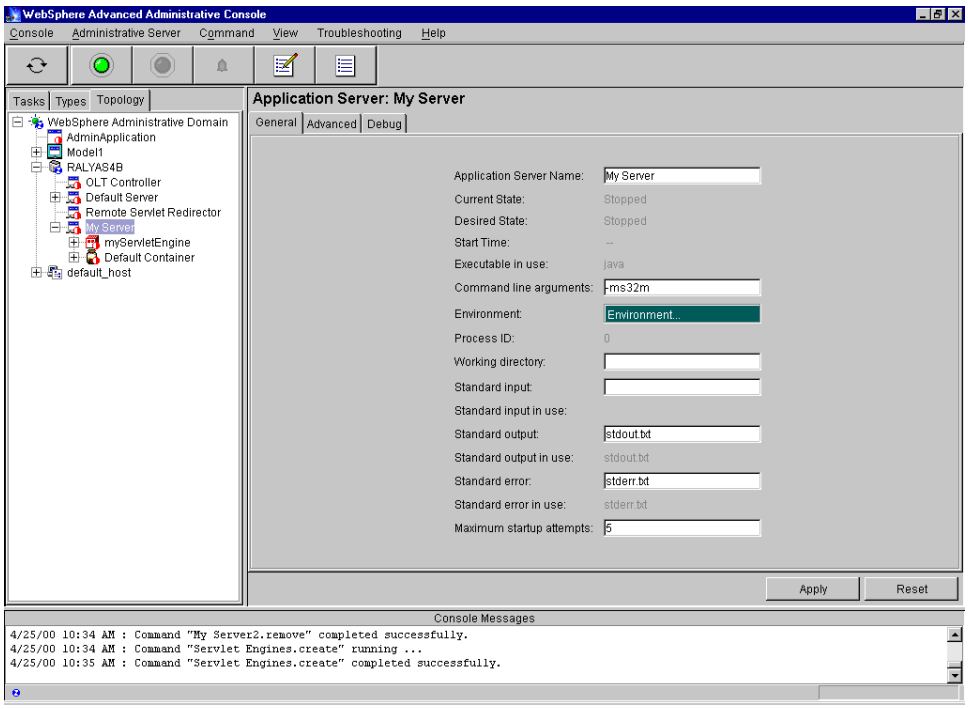

Figure 417. Topology with servlet engine

D.3 Configuration of a Web application

A Web application is defined from the Topology tab of the Administrative Console. Select the servlet engine the Web application will reside on. Right click it and select **Create->Web Application** as shown in Figure 418.

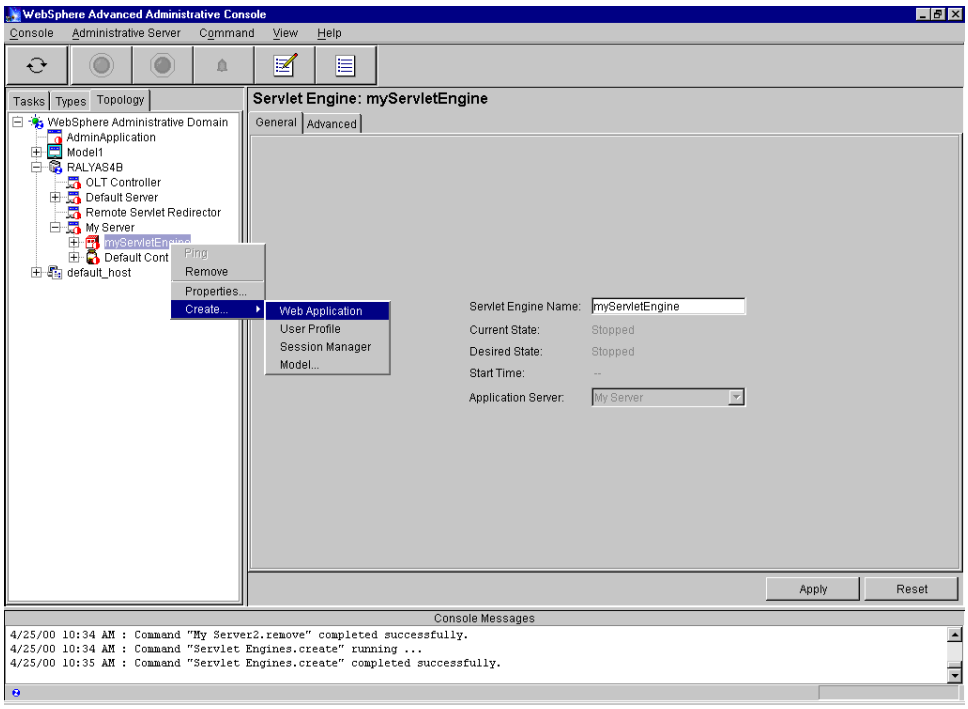

Figure 418. Create Web application

The Create Web Application window will appear. Enter a name for the Web application and click the **Create** button.

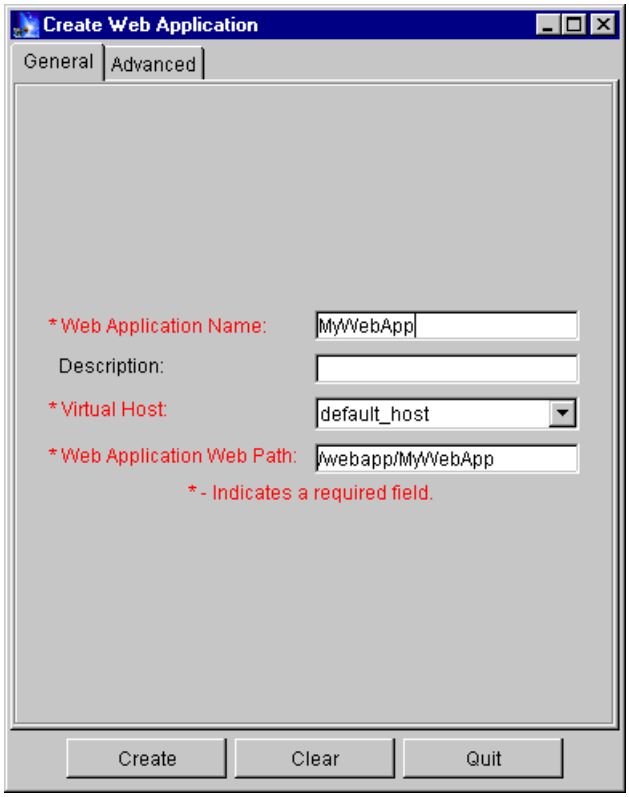

Figure 419. Create Web Application

When the Web application has been created, you will receive the following message.

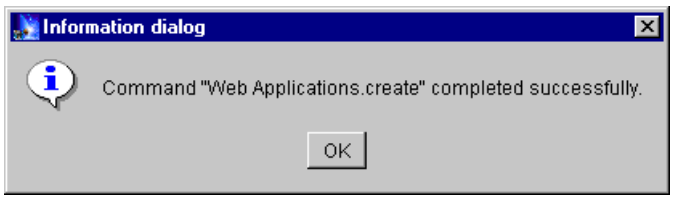

Figure 420. Web application created message

The Web application will now appear in the Topology tab.

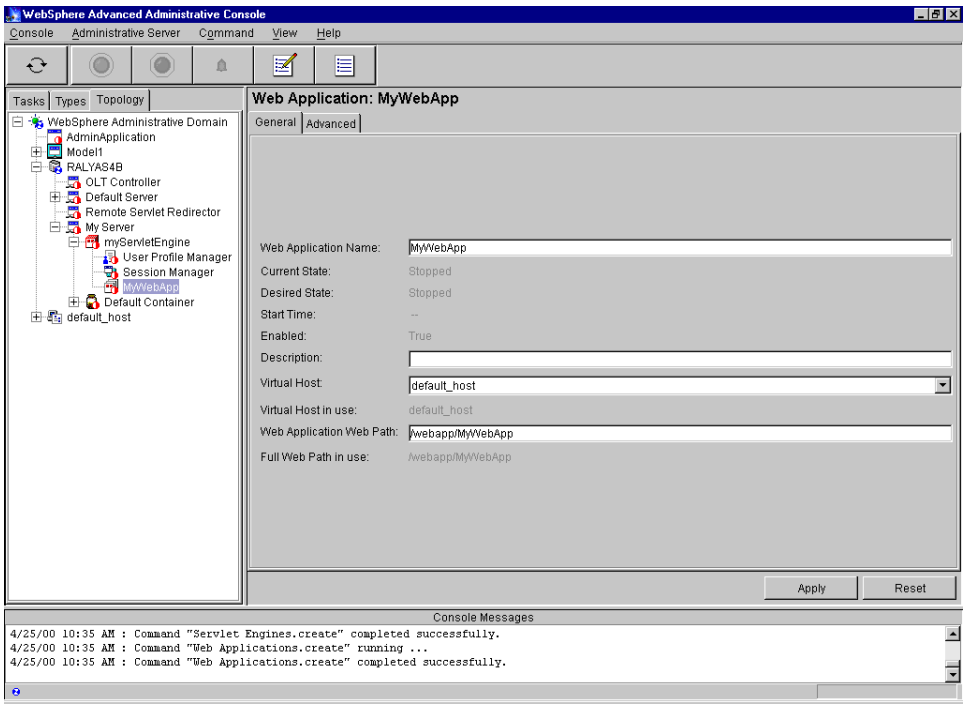

Figure 421. Topology with Web application

D.4 Servlet configuration

A servlet is configured from the Topology tab of the Administrative Console. Select the Web application the servlet will reside on. Right click it and select **Create->Servlet** as shown in Figure 422.

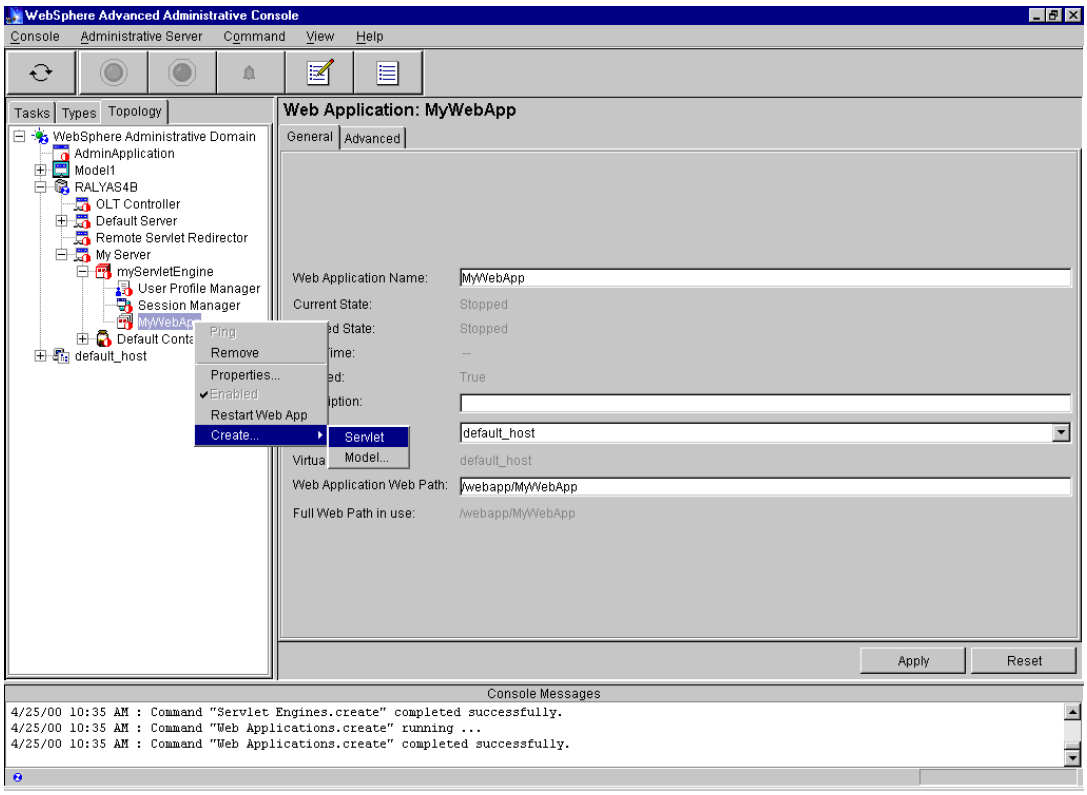

Figure 422. Create servlet

The Create Servlet window will appear. Enter the servlet name and class as shown in Figure 423, then click the **Add** button.

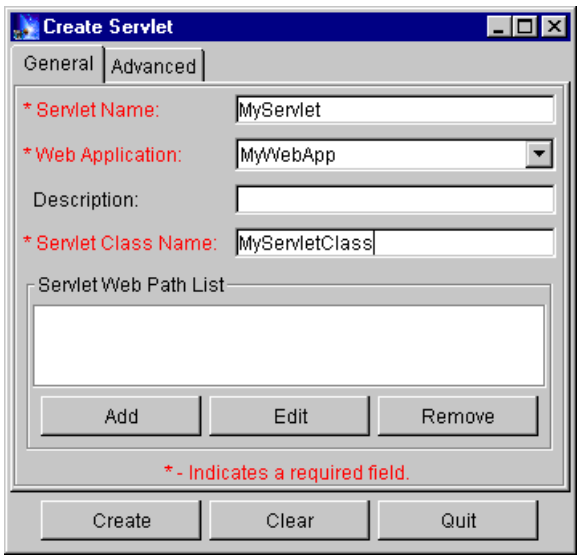

Figure 423. Create Servlet

The Add Web Path to Servlet window will appear. This will define the URI used to call the servlet as shown in Figure 424. Enter the servlet name to the end of the path and click the **OK** button.

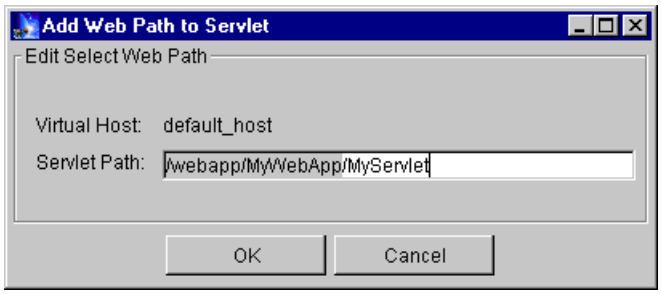

Figure 424. Servlet Web path

You will be returned to the Create Servlet Window as shown in Figure 425. Click the **Create** button.

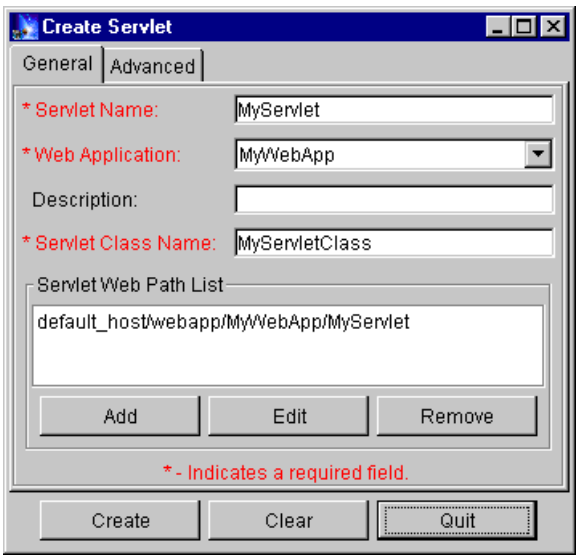

Figure 425. Create Servlet

When the servlet has been created, you will receive the following message.

Figure 426. Servlet created message

The servlet will now appear in the Topology tab.

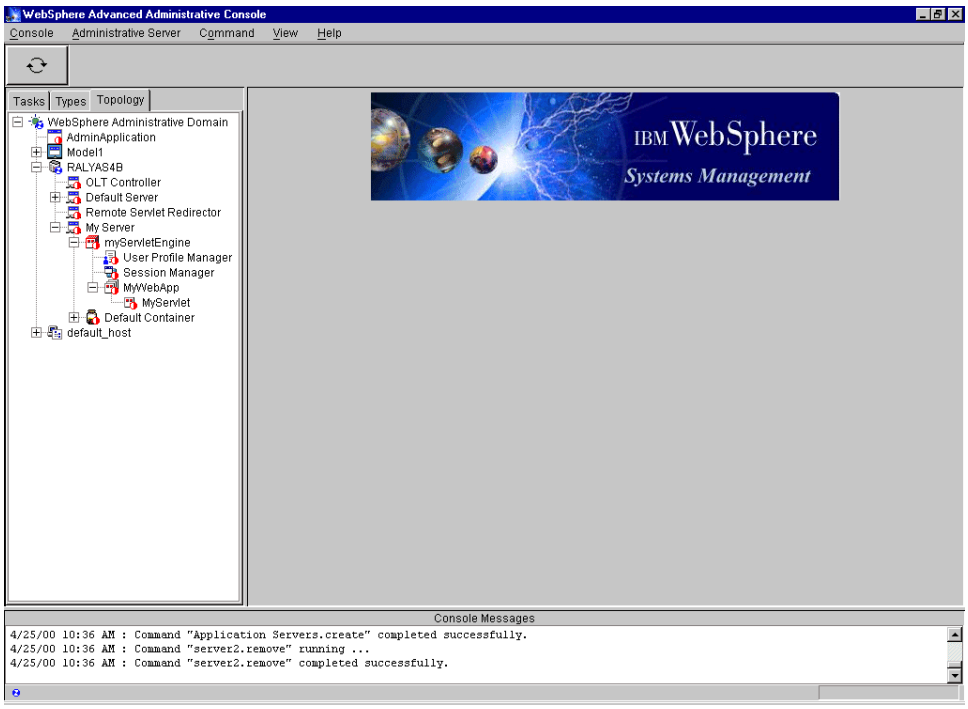

Figure 427. Topology with servlet

D.5 Consideration of multiple HTTP server instances

Regardless of the HTTP server that you choose to use (Lotus Domino, Netscape Enterprise Server, IHS, IIS), if choose to run multiple HTTP server instances you will need to install the WebSphere plugin, as appropriate for your HTTP server, for each instance. For example, with Netscape Enterprise Server this requires modification of the obj.conf file for each HTTP server instance, IIS requires that the ISAPI filter "sePlugin" be configured for each instance.

You will also need to insure that entries corresponding to the HTTP server hostname (and port number if other than the default of 80) be made to the host aliases for the appropriate virtual host (default host -> advanced tab).

Appendix E. Multi-instance template for AS/400

Answer the following questions and then place your answers into Table 49 on page 514. Then using the completed table, update the required property file values in E.2, “Property file changes” on page 515. These values can then be used to modify your new instance property files as described in 11.5, “Multi-instance support for WebSphere V3 Advanced” on page 316

We have included an example of this process in E.3, “Multi-instance configuration example” on page 517.

E.1 Questionnaire

1. What is your chosen WebSphere instance IFS directory?

/QIBM/UserData/WebASAdv/_____

Place this value in the Value section for Q.1, Table 49 on page 514

2. What will you use for the Administrative Server name?

_____

Place this value in the Value section for Q.2, Table 49 on page 514

3. Do you require the default application server to be installed?

_____

If you answered yes, then place true in the Value section for Q 3, Table 49 on page 514. Otherwise enter false.

4. What AS/400 library name would you like to use for the repository collection?

_____

Place this value in the Value section for Q.4, Table 49 on page 514

5. What unique and unused TCP/IP port number will you use for the Administrative Server bootstrap port?

_____

Place this value in the Value section for Q.5, Table 49

6. What unique and unused TCP/IP port number will you use for the Administrative Server lsd port?

_____

Place this value in the Value section for Q.6, Table 49

7. What unique and unused TCP/IP port number will you use for the ose.srvgrp.ibmappserve.clone1 port?

_____

Place this value in the Value section for Q.7, Table 49

Table 49. Properties for WebSphere instance

Q. #	Configuration options	Value
1	Instance directory	
2	mntr.admin.name	
3	install.initial.config	
4	admin.dbSchema	
5	admin.bootstrapPort	
6	admin.lsdPort	
7	ose.srvgrp.ibmappserve.clone1.port	

E.2 Property file changes

Using the values you specified in Table 49 on page 514, specify the new instance values for the bootstrap.properties, admin.properties and sas.server.props files in the tables below.

The values specified, should then be used to modify your instance property files.

E.2.1 bootstrap.properties file

The bootstrap.properties file contains the information pertinent to the HTTP plug-in. The following four properties must be changed.

Table 50. bootstrap.properties file properties requiring changes

Property	Description
server.root	This should be your chosen <was_instance_root> from question 1
ose.tmp.dir	This is the temp subdirectory of the <was_instance_root> from question 1
ose.logs.dir	This is the logs subdirectory of the <was_instance_root> from question 1
ose.srvgrp.ibmappserve.clone1.port	This must be the unique unused TCP/IP port number from question 7

Write your answers in the space provided.

Table 51. bootstrap.properties file changes

Property to edit	New value
server.root	_____
ose.tmp.dir	_____/temp
ose.logs.dir	_____/logs
ose.srvgrp.ibmappserve.clone1.port	_____

E.2.2 admin.properties

The admin.properties file contains the information pertinent to the WebSphere Administrative Server instance. The following properties must be changed.

Table 52. admin.properties file properties requiring changes

Property to edit	New Value
mntr.admin.name	This is the value from question 2.
install.initial.config	This is the value from question 3.
admin.dbSchema	This is the value from question 4.
admin.bootstrapPort	This must be the unique unused TCP/IP port number from question 5.
admin.lsdPort	This must be the unique unused TCP/IP port number from question 6.
admin.classpath	This is the classpath used by the WebSphere Administrative Server when it starts. All references to the <was_default_instance_root> must be substituted with the new <was_instance_root> from question.
admin.instance.root	This should be your chosen <was_instance_root> from question 1.
java.properties	All references to the <was_default_instance_root> must be substituted with the new <was_instance_root> from question 1.

Write your answers in the space provided.

Table 53. admin.properties file changes

Property to edit	New Value
mntr.admin.name	_____
install.initial.config	_____
admin.dbSchema	_____
admin.bootstrapPort	_____
admin.lsdPort	_____

Property to edit	New Value
admin.classpath	Substitute the reference to the <was_default_instance_root> with _____
admin.instance.root	_____
java.properties	Substitute the reference to the <was_default_instance_root> with _____

E.2.3 sas.server.props

The sas.server.props properties file is primarily used for WebSphere security. The following two properties must be changed.

Table 54. sas.server.props file properties requiring changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root> from question 1.
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root> from question 1.

Write your answers in the space provided.

Table 55. sas.server.props file changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with _____
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with _____

E.3 Multi-instance configuration example

The answers to these questions should be entered into Table 56 on page 519.

E.3.1 Questionnaire answers

1. What is your chosen WebSphere instance IFS directory?

/QIBM/UserData/WebASAdv/New_____

Place this value in the Value section for Q.1

2. What will you use for the Administrative Server name?

NEWADMIN_____

Place this value in the Value section for Q.2

3. Do you require the default application server to be installed?

yes_____

If you answered yes, then place true in the Value section for Q.3

Otherwise enter false.

4. What AS/400 library name would you like to use for the repository collection?

EJSNEW_____

Place this value in the Value section for Q.4

5. What unique and unused TCP/IP port number will you use for the Administrative Server bootstrap port?

7799_____

Place this value in the Value section for Q.5

6. What unique and unused TCP/IP port number will you use for the Administrative Server lsd port?

9999_____

Place this value in the Value section for Q.6

7. What unique and unused TCP/IP port number will you use for the ose.srvgrp.ibmappserve.clone1.port?

8899_____

Place this value in the Value section for Q.7

Table 56. Properties for WebSphere instance

Q. #	Configuration options	Value
1	Instance directory	/QIBM/UserData/WebAsAdv/New
2	mntr.admin.name	NEWADMIN
3	install.initial.config	true
4	admin.dbSchema	EJSNew
5	admin.bootstrapPort	7799
6	admin.lsdPort	9999
7	ose.srvgrp.ibmappserve.clone1.port	8899

E.3.2 Property file changes

Using these values we can build our property file changes.

E.3.2.1 Bootstrap.properties file

We edited the bootstrap.properties file and replaced the default values for our example instance as shown.

Table 57. bootstrap.properties file changes

Property to edit	New value
server.root	/QIBM/UserData/WebAsAdv/New
ose.tmp.dir	/QIBM/UserData/WebAsAdv/New/temp
ose.logs.dir	/QIBM/UserData/WebAsAdv/New/logs
ose.srvgrp.ibmappserve.clone1.port	8899

E.3.2.2 Admin.properties

We edited the admin.properties file and replaced the default values as shown.

Table 58. admin.properties file changes

Property to edit	New Value
mntr.admin.name	NEWADMIN
install.initial.config	true
admin.dbSchema	EJSNEW
admin.bootstrapPort	7799
admin.lsdPort	9999
admin.classpath	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New
admin.instance.root	/QIBM/UserData/WebAsAdv/New
java.properties	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New

E.3.2.3 sas.server.props

We edited the file and replaced the default values as shown.

Table 59. sas.server.props file changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New

E.3.3 Property file contents

We have included the three modify property files based on the example questionnaire.

E.3.3.1 bootstrap.properties file

```
#####################################################################
# @(#)bootstrap.properties1.0 98/08/21
#
# Configuration properties for JVM and plugin initialization
#

#####################################################################
# --------------------WARNING----------------------------------
# All properties in this file must start with the following prefixes:
#       server.
#       ose.
#####################################################################

#####################################################################
# System Properties                                                 #
#
# Installation Root Directory
# WARNING!!! server.root must contain forward slashes '/'
#
server.root=/QIBM/UserData/WebASAdv/New

#####################################################################
# OSE library setup                                                 #
#####################################################################

##
# Full path to the location of the OSE libraries.
#
ose.library.path=/QSYS.LIB/QEJB.LIB

#####################################################
# Names of the OSE Libraries to use for the different supported jvm Types and
# engine operation modes.
# The property is specified as "ose.library.<mode>.<jvmtype>=<Library Name>"
# The library must be in the directory specified by ose.library.path.

##
# Out-of-Process Support Libraries
#
ose.library.out=/QSYS.LIB/QEJB.LIB

##
# Single thread version
#
ose.library.outst=/QSYS.LIB/QEJB.LIB
```

Figure 428. Modified bootstrap.properties file - Part 1

```
##
# Common services support library.
#
ose.commonserv.lib=/QSYS.LIB/QEJB.LIB

##
# Single thread version
#
ose.commonservst.lib=/QSYS.LIB/QEJB.LIB


###############################################################################################
## Thread Usage Parameter
##   The following parameter determines how the application server interacts with the
##   underlying webserver.  For Netscape and Domino Go Webservers, the response to an
##   HTTP request must be made on the same thread as the request.  The Application Server
##   provides a mechanism that will ensure this behavior is satisfied. However, this insurance
##   will negatively impact performance.  Therefore, the following property allows you to
##   enable or disable the insurance, depending on the types of servlets you are running in
##   the application server.  If you are running single threaded servlets, or your multithreaded
##   servlets ALWAYS write to the response object on the original request thread, leave this
##   property set to TRUE.  If you are using chaing/filtering or you have multithreaded servlets
##   that write to the response object on a thread other than the request thread, set this property
##   to FALSE.  If you are not running Netscape or Domino Go, set this property to TRUE in all
##   cases.  If you are running the application server in a separate process from the webserver
##   (out-of-process mode), this property is ignored and always set to TRUE
#
ose.single.thread=true

##
# Temp ose directory.
#
ose.tmp.dir=/QIBM/UserData/WebASAdv/New/temp

##
# Logs OSE directory
# Directory in which native logs are written
#
ose.logs.dir=/QIBM/UserData/WebASAdv/New/logs

##
# Refresh interval for OSE properties in seconds.
# Larger values optimize performance in a production environment
#
ose.refresh.interval=20
```

Figure 429. Modified bootstrap.properties file - Part 2

```
##
# Set the native web-server plugins log level
# values of ose.native.log.leve may be a combination of
# TRACE INFORM ERROR WARNING
#
ose.native.log.level=ERROR|WARNING

##
# Set the plugin specific native web-server plugins log level
# values of ose.native.log.leve may be a combination of
# TRACE INFORM ERROR WARNING
#
ose.plugin.log.level=ERROR|WARNING

##
# Set ose.security.enabled to true to make ose
# secure your site.
#
ose.security.enabled=false

################################################################################################
###
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.concurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8899
ose.srvgrp.ibmappserve.clone1.type=local
ose.srvgrp.ibmappserve.clone1.host=localhost
```

Figure 430. Modified bootstrap.properties file - Part 3

E.3.3.2 admin.properties file

```
##############################################################################
#
#   Websphere properties file
#      #,;  - comments
#
##############################################################################
#
#   Monitor Properties:
#
#      mntr.retryValue - admin server recovery retry count
#      mntr.retryTimeout - wait time (seconds) between retry attempts
#      mntr.admin.name  - name used to start Admin server job
#
##############################################################################

mntr.retryValue=10
mntr.retryTimeout=60
mntr.admin.name=NEWADMIN

##############################################################################
#
#   Install Initial Configuration Property:
#
#      install.initial.config - when set to true, a Default (sample) environment
#                               will be setup when the Administrative Server
#                               starts
#
##############################################################################

install.initial.config=true

##############################################################################
#
#   Administrative Server properties:
#
#      admin.dbSchema - name of Administrative repository (AS/400 collection)
#      admin.dbUrl - URL location of the administrative database (local default)
#      admin.dbDriver - qualified class name of the database driver
#      admin.dbUser - optional user id to access administrative databases
#      admin.dbPassword - optional password to access administrative databases
#      admin.bootstrapPort - TCP/IP port number of the admin server
#      admin.lsdPort - TCP/IP port number of location server daemon
#      admin.traceString - trace options (see documentation for options)
#      admin.traceOutput - trace output
#      admin.jarFile - admin server jar files
#      admin.nameServiceJar - name server jar file
#      admin.initializer - initializer classes (multiple initializers allowed)
#      admin.classpath - Websphere classpath (=+ to concatenate multiple lines)
#      admin.instance.root - server root and admin/app server default current
#                            working directory
#
```

Figure 431. Modified admin.properties file - Part 1

```
#
################################################################################

admin.dbSchema=ejsnew
admin.dbUrl=jdbc:db2:*local
admin.dbDriver=com.ibm.db2.jdbc.app.DB2Driver
;admin.dbUser=
;admin.dbPassword=
admin.bootstrapPort=7799
admin.lsdPort=9999
;admin.bootstrapHost=
;admin.nodeName=
admin.nameServiceJar=/QIBM/ProdData/WebASAdv/lib/server/ns.jar
admin.jarFile=/QIBM/ProdData/WebASAdv/lib/server/repository.jar
admin.jarFile=/QIBM/ProdData/WebASAdv/lib/server/tasks.jar
admin.traceString=com.ibm.*=all=disabled
admin.traceOutput=logs/tracefile
admin.logFile=tranlog/tranlog1,tranlog/tranlog2
admin.initializer=com.ibm.ejs.security.Initializer
admin.initializer=com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer
admin.initializer=com.ibm.servlet.config.AS400SetupInitializer

admin.classpath=/QIBM/ProdData/WebASAdv/lib/wsa400.jar:
admin.classpath=+/QIBM/UserData/WebASAdv/New/properties:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ibmwebas.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/servlet.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/webtlsrn.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ns.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/ejs.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/ujc.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/repository.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/admin.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/tasks.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/x509v1.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/databeans.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ejscp.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/lotusxsl.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/xml4j.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/dertrjrt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/sslight.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/iioprt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/iioptools.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/deployTool.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/vaprt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/console.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/swingall.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ibmjndi.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/jsp10.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/properties

admin.instance.root=/QIBM/UserData/WebASAdv/New
```

Figure 432. Modified admin.properties file - Part 2

```
################################################################################
#
#   Application Server properties:
#
#      ejbsvr.java.compiler - value of application server's JVM system
#                             property 'java.compiler'
#
################################################################################

ejbsvr.java.compiler=jitc_de

################################################################################
#
#   Java properties:
#
#      java.verbose - turn on verbose mode
#      java.heap_minSize -  initial heap size
#      java.heap_maxSize -  maximum heap size
#      java.gc_verbose - message when garbage collection occurs
#      java.gc_class - class garbage collection
#      java.native_stackSize - maximum native stack size for any thread
#      java.stackSize - maximum Java stack size for any thread
#      java.properties - java properties <name>=<value> (one or more lines)
#
################################################################################

java.verbose=0
java.heap_minSize=67108864
;java.heap_maxSize=0
;java.gc_verbose=0
;java.gc_class=1
;java.native_stackSize=0
;java.stackSize=0
java.properties=com.ibm.CORBA.ConfigURL=file:///QIBM/UserData/WebASAdv/New/properti
es/sas.server.props
java.properties=os400.defineClass.optLevel=0
java.properties=jdbc.db2.cli.nojoblog=true
java.properties=com.ibm.ejs.sm.adminServer.qualifyHomeName=true
java.properties=com.ibm.ejs.sm.server.StartNotificationListenerClass=com.ibm.ejs.sm
.server.AS400AdminStartedListener
java.properties=com.ibm.ejs.dbm.FileWaitTime=32766
java.properties=com.ibm.ejs.dbm.RecordWaitTime=60

################################################################################
```

Figure 433. Modified admin.properties file - Part 3

E.3.3.3 sas.server.props file

```
##SAS Properties - Editable
#Mon Apr 17 17:19:37 GMT+00:00 2000
com.ibm.CORBA.loginUserid=
com.ibm.CORBA.principalName=RALYAS4B.ITSO.RAL.IBM.COM/
com.ibm.CORBA.loginPassword=
com.ibm.CORBA.securityEnabled=false
com.ibm.CORBA.authenticationTarget=LOCALOS
#SAS Properties - DO NOT EDIT
#Mon Apr 17 17:19:37 GMT+00:00 2000
LOCALOS.server.id=
com.ibm.CORBA.delegateCredentials=methodDefined
com.ibm.CORBA.SSLKeyRingPassword=WebAS
com.ibm.CORBA.SSLClientKeyRing=com.ibm.websphere.DummyKeyring
com.ibm.CORBA.SSLClientKeyRingPassword=WebAS
com.ibm.CORBA.securityDebug=no
com.ibm.CORBA.standardClaimQOPModels=integrity
com.ibm.CORBA.SSLV3SessionTimeout=9600
com.ibm.CORBA.LTPAServerAssociationEnabled=true
com.ibm.CORBA.securityTraceLevel=none
com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
com.ibm.CORBA.keytabFileName=/QIBM/UserData/WebASAdv/New/etc/keytab5
com.ibm.CORBA.SSLTypeIClientAssociationEnabled=true
com.ibm.CORBA.disableSecurityDuringBootstrap=false
com.ibm.CORBA.LTPAClientAssociationEnabled=false
com.ibm.CORBA.standardPerformQOPModels=confidentiality
com.ibm.CORBA.loginTimeout=30
com.ibm.CORBA.bootstrapRepositoryLocation=/QIBM/UserData/WebASAdv/New/etc/secbootstrap
LOCALOS.server.pwd=
com.ibm.CORBA.loginSource=properties
com.ibm.CORBA.SSLKeyRing=com.ibm.websphere.DummyKeyring
```

Figure 434. Modified sas.server.props file

Appendix F. Multi-instance template for AS/400 Admin-agent mode

Answer the following questions and then place your answers into Table 60 on page 530. Then using the completed table, update the required property file values in F.2, “Property file changes” on page 531 as described in 11.5, “Multi-instance support for WebSphere V3 Advanced” on page 316. These values can then be used to modify your new instance property files.

We have included an example of this process in F.3, “Multi-instance configuration example” on page 533.

F.1 Questionnaire

1. What is your chosen WebSphere instance IFS directory? This should be the same as the instance IFS directory of the full Administrative Server.

/QIBM/UserData/WebASAdv/_____

Place this value in the Value section for Q.1, Table 60 on page 530.

2. What will you use for the Administrative Server name? If possible use the same value as specified in the Full administrative instance.

_____

Place this value in the Value section for Q.2, Table 60 on page 530.

3. Do you require the default application server to be installed?

_____

If you answered yes, then place true in the Value section for Q.3, Table 60 on page 530. Otherwise enter false.

4. What is the host name of the full Administrative Server?

_____

Place this value in the Value section for Q. 4, Table 60 on page 530.

5. What TCP/IP port number is used for the full Administrative Server bootstrap port?

_____

Place this value in the Value section for Q.5, Table 60.

6. What unique and unused TCP/IP port number will you use for the Administrative Server lsd port?

_____

Place this value in the Value section for Q.6, Table 60.

7. What unique and unused TCP/IP port number will you use for the ose.srvgrp.ibmappserve.clone1.port?

_____

Place this value in the Value section for Q.7, Table 60.

Table 60. Properties for WebSphere instance

Q. #	Configuration options	Value
1	Instance directory	
2	mntr.admin.name	
3	install.initial.config	
4	admin.bootstrapHost	
5	admin.bootstrapPort	
6	admin.lsdPort	
7	ose.srvgrp.ibmappserve.clone1.port	

F.2 Property file changes

Using the values you specified in Table 60 on page 530, specify the new instance values for the bootstrap.properties, admin.properties and sas.server.props files in the tables below.

The values specified, should then be used to modify your instance property files.

F.2.1 bootstrap.properties file

The bootstrap.properties file contains the information pertinent to the HTTP plug-in. The following four properties must be changed.

Table 61. bootstrap.properties file properties requiring changes

Property	Description
server.root	This should be your chosen <was_instance_root> from question 1.
ose.tmp.dir	This is the temp subdirectory of the <was_instance_root> from question 1.
ose.logs.dir	This is the logs subdirectory of the <was_instance_root> from question 1.
ose.srvgrp.ibmappserve.clone1.port	This must be the unique unused TCP/IP port number from question 7.

Write your answers in the space provided.

Table 62. bootstrap.properties file changes

Property to edit	New value
server.root	_____
ose.tmp.dir	_____/temp
ose.logs.dir	_____/logs
ose.srvgrp.ibmappserve.clone1.port	_____

F.2.2 admin.properties

The admin.properties file contains the information pertinent to the WebSphere Administrative Server instance. The following properties must be changed.

Table 63. admin.properties file properties requiring changes

Property to edit	New Value
mntr.admin.name	This is the value from question 2.
install.initial.config	This is the value from question 3.
admin.bootstrapHost	This is the value from question 4.
admin.bootstrapPort	This is the TCP/IP port number from question 5.
admin.lsdPort	This must be the unique unused TCP/IP port number from question 6.
admin.classpath	This is the classpath used by the WebSphere Administrative Server when it starts. All references to the <was_default_instance_root> must be substituted with the new <was_instance_root> from question 1.
admin.instance.root	This should be your chosen <was_instance_root> from question 1.
java.properties	All references to the <was_default_instance_root> must be substituted with the new <was_instance_root> from question 1.

Write your answers in the space provided.

Table 64. admin.properties file changes

Property to edit	New Value
mntr.admin.name	_____
install.initial.config	_____
admin.dbSchema	_____
admin.bootstrapPort	_____
admin.lsdPort	_____
admin.classpath	Substitute the reference to the <was_default_instance_root> with _____
admin.instance.root	_____

Property to edit	New Value
java.properties	Substitute the reference to the <was_default_instance_root> with _____

F.2.3 sas.server.props

The sas.server.props properties file is primarily used for WebSphere security. The following two properties must be changed.

Table 65. sas.server.props file properties requiring changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root> from question 1.
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with the new <was_instance_root> from question 1.

Write your answers in the space provided.

Table 66. sas.server.props file changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with _____
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with _____

F.3 Multi-instance configuration example

The answers to these questions should be entered into Table 67 on page 535.

F.3.1 Questionnaire answers

1. What is your chosen WebSphere instance IFS directory? This should be the same as the instance IFS directory of the full Administrative Server.

/QIBM/UserData/WebASAdv/New_____

Place this value in the Value section for Q.1

2. What will you use for the Administrative Server name? If possible use the same value as specified in the full Administrative instance.

NEWADMIN_____

Place this value in the Value section for Q.2

3. Do you require the default application server to be installed?

yes_____

If you answered yes, then place true in the Value section for Q.3.

Otherwise enter false.

4. What is the host name of the full Administrative Server?

RALYAS4B_____

Place this value in the Value section for Q.4

5. What TCP/IP port number is used for the Administrative Server bootstrap port?

7799_____

Place this value in the Value section for Q.5

6. What unique and unused TCP/IP port number will you use for the Administrative Server lsd port?

9999_____

Place this value in the Value section for Q.6

7. What unique and unused TCP/IP port number will you use for the ose.srvgrp.ibmappserve.clone1.port?

8899_____

Place this value in the Value section for Q.7

Table 67. Properties for WebSphere instance

Q. #	Configuration options	Value
1	Instance directory	/QIBM/UserData/WebAsAdv/New
2	mntr.admin.name	NEWADMIN
3	install.initial.config	true
4	admin.bootstrapHost	RALYAS4B
5	admin.bootstrapPort	7799
6	admin.lsdPort	9999
7	ose.srvgrp.ibmappserve.clone1.port	8899

F.3.2 Property file changes

Using these values we can build our property file changes.

F.3.2.1 Bootstrap.properties file

We edited the bootstrap.properties file and replaced the default values for our example instance as shown.

Table 68. bootstrap.properties file changes

Property to edit	New value
server.root	/QIBM/UserData/WebAsAdv/New
ose.tmp.dir	/QIBM/UserData/WebAsAdv/New/temp
ose.logs.dir	/QIBM/UserData/WebAsAdv/New/logs
ose.srvgrp.ibmappserve.clone1.port	8899

F.3.2.2 Admin.properties

We edited the admin.properties file and replaced the default values as shown.

Table 69. admin.properties file changes

Property to edit	New Value
mntr.admin.name	NEWADMIN
install.initial.config	true

Property to edit	New Value
admin.bootstrapHost	RALYAS4B
admin.bootstrapPort	7799
admin.lsdPort	9999
admin.classpath	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New
admin.instance.root	/QIBM/UserData/WebAsAdv/New
java.properties	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New

F.3.2.3 sas.server.props

We edited the file and replaced the default values as shown.

Table 70. sas.server.props file changes

Property to edit	New value
com.ibm.CORBA.keytabFileName	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New
com.ibm.CORBA.botstrapRepository Location	Substitute the reference to the <was_default_instance_root> with /QIBM/UserData/WebAsAdv/New

F.3.3 Property file contents

We have included the three modified property files based on the example questionnaire.

F.3.3.1 bootstrap.properties file

```
#####################################################################
# @(#)bootstrap.properties1.0 98/08/21
#
# Configuration properties for JVM and plugin initialization
#

#####################################################################
# --------------------WARNING-----------------------------------
# All properties in this file must start with the following prefixes:
#       server.
#       ose.
#####################################################################

#####################################################################
# System Properties                                                 #
#
# Installation Root Directory
# WARNING!!! server.root must contain forward slashes '/'
#
server.root=/QIBM/UserData/WebASAdv/New

#####################################################################
# OSE library setup                                                 #
#####################################################################

##
# Full path to the location of the OSE libraries.
#
ose.library.path=/QSYS.LIB/QEJB.LIB

#####################################################
# Names of the OSE Libraries to use for the different supported jvm Types and
# engine operation modes.
# The property is specified as "ose.library.<mode>.<jvmtype>=<Library Name>"
# The library must be in the directory specified by ose.library.path.

##
# Out-of-Process Support Libraries
#
ose.library.out=/QSYS.LIB/QEJB.LIB

##
# Single thread version
#
ose.library.outst=/QSYS.LIB/QEJB.LIB
```

Figure 435. Modified bootstrap.properties file - Part 1

```
##
# Common services support library.
#
ose.commonserv.lib=/QSYS.LIB/QEJB.LIB

##
# Single thread version
#
ose.commonservst.lib=/QSYS.LIB/QEJB.LIB


###############################################################################################
## Thread Usage Parameter
##   The following parameter determines how the application server interacts with the
##   underlying webserver.  For Netscape and Domino Go Webservers, the response to an
##   HTTP request must be made on the same thread as the request.  The Application Server
##   provides a mechanism that will ensure this behavior is satisfied. However, this insurance
##   will negatively impact performance.  Therefore, the following property allows you to
##   enable or disable the insurance, depending on the types of servlets you are running in
##   the application server.  If you are running single threaded servlets, or your multithreaded
##   servlets ALWAYS write to the response object on the original request thread, leave this
##   property set to TRUE.  If you are using chaing/filtering or you have multithreaded servlets
##   that write to the response object on a thread other than the request thread, set this property
##   to FALSE.  If you are not running Netscape or Domino Go, set this property to TRUE in all
##   cases.  If you are running the application server in a separate process from the webserver
##   (out-of-process mode), this property is ignored and always set to TRUE
#
ose.single.thread=true

##
# Temp ose directory.
#
ose.tmp.dir=/QIBM/UserData/WebASAdv/New/temp

##
# Logs OSE directory
# Directory in which native logs are written
#
ose.logs.dir=/QIBM/UserData/WebASAdv/New/logs

##
# Refresh interval for OSE properties in seconds.
# Larger values optimize performance in a production environment
#
ose.refresh.interval=20
```

Figure 436. Modified bootstrap.properties file - Part 2

```
##
# Set the native web-server plugins log level
# values of ose.native.log.leve may be a combination of
# TRACE INFORM ERROR WARNING
#
ose.native.log.level=ERROR|WARNING

##
# Set the plugin specific native web-server plugins log level
# values of ose.native.log.leve may be a combination of
# TRACE INFORM ERROR WARNING
#
ose.plugin.log.level=ERROR|WARNING

##
# Set ose.security.enabled to true to make ose
# secure your site.
#
ose.security.enabled=false

################################################################################
# Admin Server Properties
#
ose.adminqueue=ibmappserve
ose.max.concurrency=1
ose.srvgrp.ibmappserve.type=FASTLINK
ose.srvgrp.ibmappserve.clonescount=1
ose.srvgrp.ibmappserve.clone1.port=8899
ose.srvgrp.ibmappserve.clone1.type=local
ose.srvgrp.ibmappserve.clone1.host=localhost
```

Figure 437. Modified bootstrap.properties file - Part 3

F.3.3.2 admin.properties file

```
##############################################################################
#
#   Websphere properties file
#      #,;  - comments
#
##############################################################################
#
#   Monitor Properties:
#
#      mntr.retryValue - admin server recovery retry count
#      mntr.retryTimeout - wait time (seconds) between retry attempts
#      mntr.admin.name  - name used to start Admin server job
#
##############################################################################

mntr.retryValue=10
mntr.retryTimeout=60
mntr.admin.name=NEWADMIN

##############################################################################
#
#   Install Initial Configuration Property:
#
#      install.initial.config - when set to true, a Default (sample) environment
#                               will be setup when the Administrative Server
#                               starts
#
##############################################################################

install.initial.config=true

##############################################################################
#
#   Administrative Server properties:
#
#      admin.dbSchema - name of Administrative repository (AS/400 collection)
#      admin.dbUrl - URL location of the administrative database (local default)
#      admin.dbDriver - qualified class name of the database driver
#      admin.dbUser - optional user id to access administrative databases
#      admin.dbPassword - optional password to access administrative databases
#      admin.bootstrapPort - TCP/IP port number of the admin server
#      admin.lsdPort - TCP/IP port number of location server daemon
#      admin.traceString - trace options (see documentation for options)
#      admin.traceOutput - trace output
#      admin.jarFile - admin server jar files
#      admin.nameServiceJar - name server jar file
#      admin.initializer - initializer classes (multiple initializers allowed)
#      admin.classpath - Websphere classpath (=+ to concatenate multiple lines)
#      admin.instance.root - server root and admin/app server default current
#                            working directory
#
```

Figure 438. Modified admin.properties file - Part 1

```
#
################################################################################

;admin.dbSchema=ejsadmin
;admin.dbUrl=jdbc:db2:*local
;admin.dbDriver=com.ibm.db2.jdbc.app.DB2Driver
;admin.dbUser=
;admin.dbPassword=
admin.bootstrapPort=7799
admin.lsdPort=9999
admin.bootstrapHost=RALYAS4B
;admin.nodeName=
;admin.nameServiceJar=/QIBM/ProdData/WebASAdv/lib/server/ns.jar
;admin.jarFile=/QIBM/ProdData/WebASAdv/lib/server/repository.jar
;admin.jarFile=/QIBM/ProdData/WebASAdv/lib/server/tasks.jar
admin.traceString=com.ibm.*=all=disabled
admin.traceOutput=logs/tracefile
admin.logFile=tranlog/tranlog1,tranlog/tranlog2
admin.initializer=com.ibm.ejs.security.Initializer
admin.initializer=com.ibm.servlet.engine.ejs.ServletEngineAdminInitializer
admin.initializer=com.ibm.servlet.config.AS400SetupInitializer


admin.classpath=/QIBM/ProdData/WebASAdv/lib/wsa400.jar:
admin.classpath=+/QIBM/UserData/WebASAdv/New/properties:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ibmwebas.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/servlet.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/webtlsrn.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ns.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/ejs.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/ujc.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/repository.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/admin.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/tasks.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/x509v1.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/databeans.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ejscp.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/lotusxsl.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/xml4j.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/dertrjrt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/sslight.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/iioprt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/iioptools.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/deployTool.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/vaprt.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/console.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/swingall.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/ibmjndi.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/lib/server/jsp10.jar:
admin.classpath=+/QIBM/ProdData/WebASAdv/properties
```

Figure 439. Modified admin.properties file - Part 2

```
################################################################################
#
#   Application Server properties:
#
#      ejbsvr.java.compiler - value of application server's JVM system
#                             property 'java.compiler'
#
################################################################################

ejbsvr.java.compiler=jitc_de

################################################################################
#
#   Java properties:
#
#      java.verbose - turn on verbose mode
#      java.heap_minSize -  initial heap size
#      java.heap_maxSize -  maximum heap size
#      java.gc_verbose - message when garbage collection occurs
#      java.gc_class - class garbage collection
#      java.native_stackSize - maximum native stack size for any thread
#      java.stackSize - maximum Java stack size for any thread
#      java.properties - java properties <name>=<value> (one or more lines)
#
################################################################################

java.verbose=0
java.heap_minSize=67108864
;java.heap_maxSize=0
;java.gc_verbose=0
;java.gc_class=1
;java.native_stackSize=0
;java.stackSize=0
java.properties=com.ibm.CORBA.ConfigURL=file:///QIBM/UserData/WebASAdv/New/properti
es/sas.server.props
java.properties=os400.defineClass.optLevel=0
java.properties=jdbc.db2.cli.nojoblog=true
java.properties=com.ibm.ejs.sm.adminServer.qualifyHomeName=true
java.properties=com.ibm.ejs.sm.server.StartNotificationListenerClass=com.ibm.ejs.sm
.server.AS400AdminStartedListener
java.properties=com.ibm.ejs.dbm.FileWaitTime=32766
java.properties=com.ibm.ejs.dbm.RecordWaitTime=60

################################################################################
```

Figure 440. Modified admin.properties file - Part 3

F.3.3.3 sas.server.props file

```
##SAS Properties - Editable
#Mon Apr 17 17:19:37 GMT+00:00 2000
com.ibm.CORBA.loginUserid=
com.ibm.CORBA.principalName=RALYAS4B.ITSO.RAL.IBM.COM/
com.ibm.CORBA.loginPassword=
com.ibm.CORBA.securityEnabled=false
com.ibm.CORBA.authenticationTarget=LOCALOS
#SAS Properties - DO NOT EDIT
#Mon Apr 17 17:19:37 GMT+00:00 2000
LOCALOS.server.id=
com.ibm.CORBA.delegateCredentials=methodDefined
com.ibm.CORBA.SSLKeyRingPassword=WebAS
com.ibm.CORBA.SSLClientKeyRing=com.ibm.websphere.DummyKeyring
com.ibm.CORBA.SSLClientKeyRingPassword=WebAS
com.ibm.CORBA.securityDebug=no
com.ibm.CORBA.standardClaimQOPModels=integrity
com.ibm.CORBA.SSLV3SessionTimeout=9600
com.ibm.CORBA.LTPAServerAssociationEnabled=true
com.ibm.CORBA.securityTraceLevel=none
com.ibm.CORBA.SSLTypeIServerAssociationEnabled=true
com.ibm.CORBA.keytabFileName=/QIBM/UserData/WebASAdv/New/etc/keytab5
com.ibm.CORBA.SSLTypeIClientAssociationEnabled=true
com.ibm.CORBA.disableSecurityDuringBootstrap=false
com.ibm.CORBA.LTPAClientAssociationEnabled=false
com.ibm.CORBA.standardPerformQOPModels=confidentiality
com.ibm.CORBA.loginTimeout=30
com.ibm.CORBA.bootstrapRepositoryLocation=/QIBM/UserData/WebASAdv/New/etc/secbootstrap
LOCALOS.server.pwd=
com.ibm.CORBA.loginSource=properties
com.ibm.CORBA.SSLKeyRing=com.ibm.websphere.DummyKeyring
```

Figure 441. Modified sas.server.props file

Appendix G. Sample Network Dispatcher configuration script

The following sample shell script is the script which we used for our test environments.

```
#!/bin/ksh
################################################################################
# This script must be placed in Network Dispatcher's configurations dir
################################################################################

# Network Dispatcher Non Forwarding Addresses
NFA=rs60008.itso.ral.ibm.com

# CLUSTER Addresses
CLUSTER=www.itso.ral.ibm.com

# List of all the Web Servers
SERVER1=rs6000x.itso.ral.ibm.com
SERVER2=rs6000y.itso.ral.ibm.com

LOGLEVEL=5
# ********************************************
# *    START NETWORK DISPATCHER               *
# ********************************************
# Make sure the root user is the one executing this script.
iam=`whoami`
if [[ "$iam" != "root" ]]
  then
   echo "You must login as root to run this script"
   exit 2
fi

# Starts up NETWORK DISPATCHER server functionality
echo "Starting up Network Dispatcher Server..."
/usr/bin/ndserver start

# Sleep for 5 seconds to allow ndserver to fully start up
# before executor starts up.
sleep 5
# ********************************************
# *     NETWORK DISPATCHER EXECUTOR           *
# ********************************************
```

Figure 442. Sample configuration shell script for the Network Dispatcher (1/3)

```
# The Executor is responsible for routing requests to the Web Servers.
echo "Starting up the Network Dispatcher Executor..."
/usr/bin/ndcontrol executor start

echo "Setting the non-forwarding address on EXECUTOR..."
/usr/bin/ndcontrol executor set nfa $NFA
# ********************************************
# *     NETWORK DISPATCHER DEFINE CLUSTER      *
# ********************************************
# Network Dispatcher will route requests sent to the CLUSTER IP address
# to the corresponding server machines defined to that CLUSTER.
echo "Loading CLUSTER address for Scenarios..."
/usr/bin/ndcontrol cluster add $CLUSTER

# Set attributes for the CLUSTER here using /usr/bin/ndcontrol cluster
# We do not need any of these attributes
# Define the ports that this CLUSTER uses
echo "Creating ports for CLUSTER: $CLUSTER with HTTP:80..."
/usr/bin/ndcontrol port add $CLUSTER:80

# This step adds all the Web Servers to the ports in the CLUSTER
echo "Adding Web Servers to the $CLUSTER CLUSTER..."
/usr/bin/ndcontrol server add $CLUSTER:80:$SERVER1+$SERVER2

# ********************************************
# *     NETWORK DISPATCHER MANAGER             *
# ********************************************
# We will now start the load balancing components of Interactive Network
# Dispatcher.  The main load balancing component is called the manager.
# The manager will update the weight of each of the server machines
# based on four policies:
#   1.  The number of current TCP connections for each Web Server
#   2.  The number of new TCP connections for each Web Server
#   3.  Input from HTTP Advisor on the Network Dispatchers
#   4.  Input from the ISS Manager which receives metrics from the ISS
# Agents on the Web Servers
# Each of these policies can be given a weight or importance.
echo "Starting the Network Dispatcher MANAGER ..."
echo "Logs for ND MANAGER in /usr/lpp/nd/dispatcher/logs/MANAGER.log"
/usr/bin/ndcontrol manager start MANAGER.log
```

Figure 443. Sample configuration shell script for the Network Dispatcher (2/3)

```
# ********************************************
# *     NETWORK DISPATCHER HTTP ADVISORS       *
# ********************************************
# The third load balancing component of Interactive Network Dispatcher
# is the advisors.  The advisors give the manager more insight into the
# servers ability to serve TCP requests (such as HTTP).
echo "Starting the HTTP advisor on port 80 ..."
echo "Logs for HTTP Advisor for port 80 in
/usr/lpp/nd/dispatcher/logs/http_80.log"
/usr/bin/ndcontrol advisor start http 80 http_80.log

# Set attributes for the Network Dispatcher Advisor for HTTP on Port 80
echo "Setting attributes for the HTTP ADVISOR on Port 80..."
/usr/bin/ndcontrol advisor loglevel http 80 $LOGLEVEL
/usr/bin/ndcontrol advisor logsize http 80 1000000

# ********************************************
# *     NETWORK DISPATCHER MANAGER             *
# ********************************************
# Set attributes for the Network Dispatcher MANAGER
# Setting the proportions (which must add up to 100%) sets the
# importance at which the Network Dispatcher Manager decides which
# Web Server to route the incoming request to.
# eg. /usr/bin/ndcontrol manager proportions <active> <new> <advisors>
<ISS>
#    <active>   - Number of active HTTP connections on each Web Server
#    <new>      - Number of new HTTP connections received by Web Server
# so far
#    <advisors> - Metrics received from the ND HTTP Advisor
#    <ISS>      - Metrics received from the ISS Manager collected from
# Agents
echo "Setting the proportions of the ND MANAGER to 20 20 60 0 ..."
/usr/bin/ndcontrol manager proportions 20 20 60 0

# ********************************************
# *     NETWORK DISPATCHER CONFIGURE CLUSTER   *
# ********************************************
# The final step in setting up the Dispatcher machine is to alias the
# Network Interface Card.
/usr/bin/ndcontrol cluster configure $CLUSTER
# END OF NDstart SCRIPT
```

Figure 444. Sample configuration shell script for the Network Dispatcher (3/3)

Appendix H. Parameters for thin Servlet Redirector

In this is appendix, we discuss parameters for configuring thin Servlet Redirector.

H.1 Standalone Servlet Redirector iiopredirector.xml file parameters

H.1.1 Servlet Redirector transport information

The Servlet Redirector transport properties describe how the HTTP plug-in will contact the Servlet Redirector. This information is used when the Servlet Redirector is started.

The following portion of the iiopredirector.xml file describes the transport.

```
<active-transport>ose</active-transport>
   <transport>
      <name>ose</name>
      <code>com.ibm.servlet.engine.oselistener.SMSQTransport</code>
      <arg name="port" value="8110"></arg>
      <arg name="queueName" value="queue1"></arg>
      <arg name="maxConcurrency" value="50"></arg>
      <arg name="type" value="local"></arg>
      <arg name="server_root" value="$server_root$"></arg>
      <arg name="cloneIndex" value="1"></arg>
   </transport>
```

Figure 445. iiopredirector.xml transport entries

The `<active-transport>` entry contains the name of the transport that will be used. In this case, the name of the transport is `ose`. This should match the `<name>` entry under `<transport>`.

The `<transport>` entry contains the information defining a transport.

The `<name>` entry gives us a way to reference this transport. This is what is used when we specify the `<active-transport>`.

The `<code>` entry is the Java class that implements the transport. OSE is implemented by the `com.ibm.servlet.engine.oselistener.SMSQTransport` class.

The remaining six `<arg>` entries define parameters that are passed to the `<code>`.

The `port` argument specifies the port number that OSE will use for communication between the HTTP plug-in and the Servlet Redirector. Port 8110 is the default. Any port between 1025 and 65533 that is not currently active can be used.

The `queueName` argument specifies the name of the queue that will hold the requests between the HTTP plug-in and the Servlet Redirector.

The `maxConcurrency` argument specifies the maximum number of concurrent transport connections between the HTTP plug-in and the Servlet Redirector. Each connection represents a request for a servlet.

The `type` argument specifies the type of the transport used between the HTTP plug-in and the Servlet Redirector. Valid entries are `local` and `remote`. Servlet Redirector will always use `local`.

The `server_root` argument specifies the root directory for the Servlet Redirector. This is set at runtime using the `server_root` environment variable set inside of WebSphere.

The `cloneIndex` argument specifies the number of clones to route the calls among. Only one clone of the standalone Servlet Redirector can be run.

H.1.2 Administrative Server information

The information regarding the Administrative Server is used when the plug-in files are created (see 15.5.1, “Generate the plug-in files” on page 416).

We will need to modify this information.

The following portion of the iiopredirector.xml file describes the Administrative Server.

```
<admin-node-name>localhost</admin-node-name>
<qualify-home-names>true</qualify-home-names>
<name-service-node-name>localhost</name-service-node-name>
<name-service-port>900</name-service-port>
```

Figure 446. iiopredirector.xml administrative server entries

The `<admin-node-name>` specifies the name of the node that conatins the Administrative Server. This value is case sensitive.

The `<qualify-home-names>` specifies whether to fully qualify the host name or not.

The <name-service-node-name> specifies the host on which the naming service resides, which in this example is the same machine as the <admin-node-name>.

The <name-service-port> specifies the port number the naming service listens on. The default is 900.

H.2 StandalonePluginCfg

There are two usage models for the Standalone Plugin Cfg from the command line. You can pass the queue properties from the XML file (iiopredirector.xml):

```
java com.ibm.servlet.oselistener.systemsmgmt.StandalonePluginCfg
-serverRoot <where the product is installed>
-adminNodeName <name of the admin node you are talking to>
-queueProps < <install_root>/properties/iiopredirector.xml>
```

Alternatively, you can pass the queue properties through the command line, without using the XML file:

```
java com.ibm.servlet.oselistener.systemsmgmt.StandalonePluginCfg
-serverRoot <where the product is installed>
-adminNodeName <name of the admin node you are talking to>
-queueName <name the matches the name in iiopredirector.xml>
-queuePort    <port that matches the port in iiopredirector.xml>
-queueType <local | remote, matching iiopredirector.xml>
```

Here is the complete command line syntax of the StandalonePluginCfg class:

```
java com.ibm.servlet.engine.oselistener.systemsmgmt.StandalonePluginCfg
     -serverRoot
    -adminNodeName
[ -nameServiceNodeName ]
  [ -nameServicePort ]
  [ -queueProps ] |
  [ -queueName
  -queuePort
  -queueType ]]
      [[ -traceString ]
   [ -traceFile ]
```

```
    [ -inMemoryTrace ]] |
[ -help | /help | -? | /? ]
```

H.3 IIOPRedirector

There are two usage models for the standalone Servlet Redirector (IIOPRedirector) from the command line. You can pass the arguments from the XML file (iiopredirector.xml), which is loaded from the root of the classloader:

```
java com.ibm.servlet.engine.ejs.IIOPRedirector
```

Alternatively, you can pass all arguments from the command line, without using the XML file:

```
java com.ibm.servlet.engine.ejs.IIOPRedirector <command line arguments>
```

Here is the complete command line syntax of the IIOPRedirector class:

```
java com.ibm.servlet.engine.ejs.IIOPRedirector
[ -queueType <local | remote | remote_java>
       -queuePort <port number>
        -queueName <queue name>
        -reqThreads <num request threads>
        -nativeLog <name of native log file>
        -logMask <log event mask>
        -cloneIndex <index>
        -adminNodeName <node name>
        -qualifyHomeName <true/false> ]
      [ -nameServiceNodeName <host name of the name service> ]
      [ -nameServicePort <port number> ]
      [ -traceString <trace spec>
      [ -traceFile <file name> ]
      [ -inMemoryTrace <number of entries>]] |
      [ -help | /help | -? | /? ]
```

In addition, you can set an optional Java environment variable:

```
com.ibm.servlet.engine.ejs.IIOPRedirector.cache.size=<LRU Cache Size>
```

Appendix I. Accessing remote DB2 UDB databases

DB2 UDB databases can be accessed by applications that run on the same machine as the database, or from a remote machine by using a client/server connection. This appendix shows you how to set up the connection to a remote database.

I.1 Overview of remote access to DB2 UDB server

All remote clients use a communications product to support the protocol the client uses when accessing a remote database server. The protocol stack must be installed and configured before a remote client can communicate with a remote DB2 UDB server.

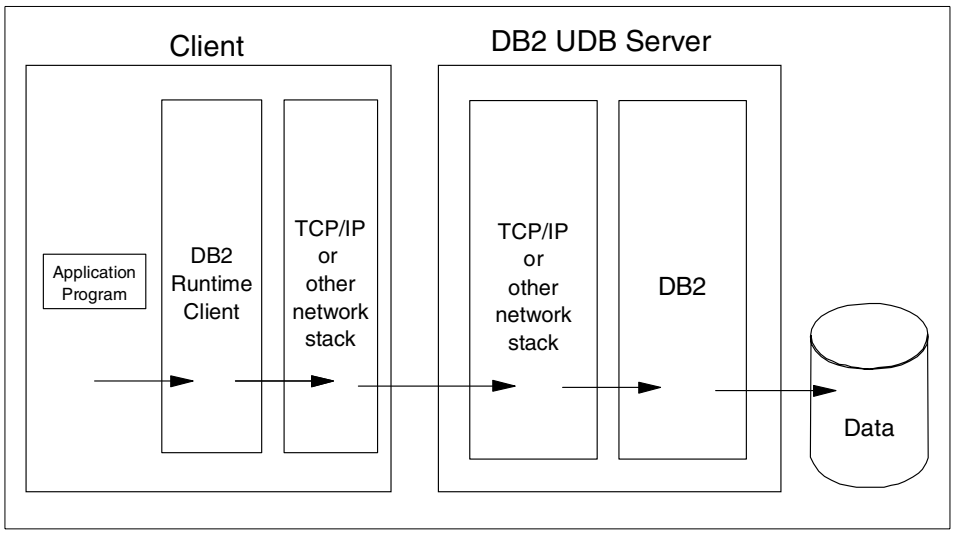

Figure 447. Remote client flow to DB2 UDB server

I.2 Steps for configuring a remote client to access a database server

DB2 UDB provides many different methods for configuring a remote client that wants to access a DB2 UDB database server. Though the methods may differ, the steps are basically the same:

- Make sure that the communication product (or protocol stack) is installed and configured on the client workstation.

- Perform the communication-specific steps. For example, if you are using TCP/IP, you need to update the services file on the server with the port number(s) that the database server is listening on.
- Catalog the remote node. This is the database server to which you want to establish the connection.
- Catalog the remote database. A DB2 UDB database server can support both local and remote clients concurrently.

I.3 Accessing remote DB2 UDB server from WebSphere Admin Server

The following picture shows you how WebSphere configuration will map to the DB2 client configuration for accessing to the remote DB2 UDB server.

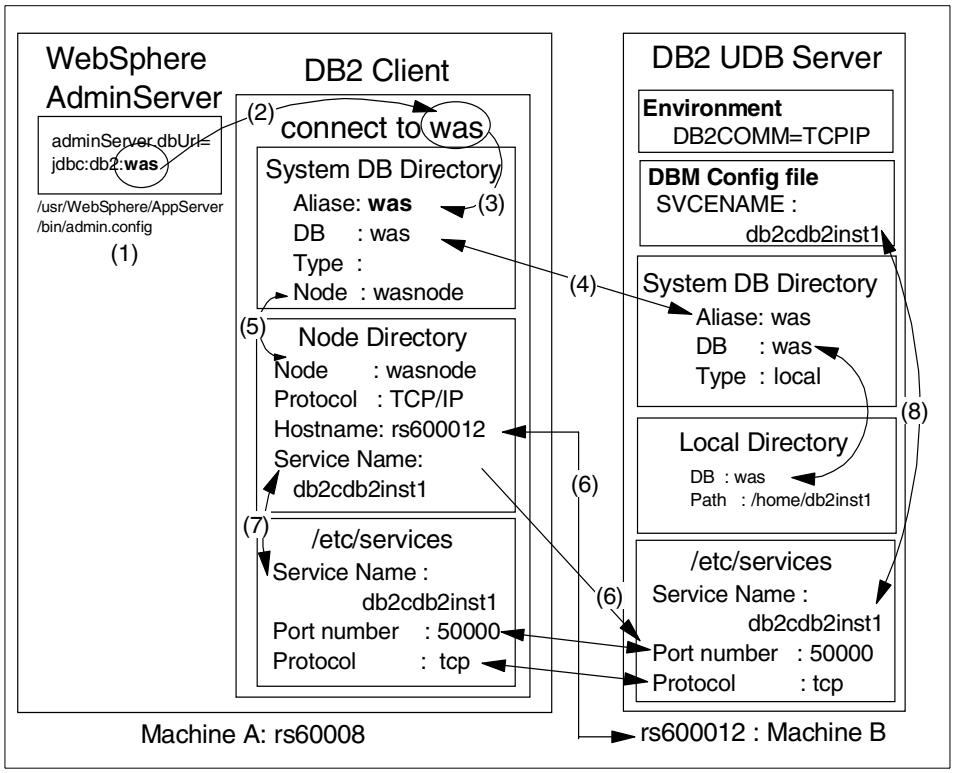

Figure 448. WebSphere and remote DB2 UDB server/client configuration

The following are descriptions of the above picture with each number referred to on the figure.

1. In the <was_dir>/bin/admin.config file, there is an adminServer.dbUrl entry which specifies the database name of the WebSphere Administrative Repository for example:

   ```
   adminServer.dbUrl=jdbc:db2:was
   ```

   The last argument in the line, `was`, is the database name of the Administrative Repository.

   When you start the WebSphere Administrative Server, it will refer to the admin.config file and will find the database name of the Administrative Repository, in our example, `was`.

2. Then, the Administrative Server will connect to the database `was` via the DB2 client. To do so, the DB2 client will run the command, `connect to <database name>`. In our example:

   ```
   connect to was
   ```

3. The database name which we use with the `connect` command (**was)**, must much the database alias (**was**), which is specified in the System DB directory on the client machine (WebSphere machine).

4. The database name to which its alias refers is stored in the System DB directory on the client (WebSphere machine), . In this example, the db2 client catalog database name or aliases named `was` is the same name as the database on the remote server.

   The database name on the client (WebSphere) must match the database alias on the DB server, in our case **was**.

5. The node name information in the System directory, in our case **wasnode** which is associated with the alias **was**, refers to the node entry in the Node directory.

6. The node name entry in the Node directory corresponds to a TCP/IP configuration. The node name has the information of the DB server name and its service name (or port number). In our example, node **wasnode** has the hostname, **rs600012**, and its service port number, **50000**. In Figure 448 on page 554, we enterd the service name instead of port number.

   Now, the DB2 client has enough information to access the database on the remote server.

7. When you create a catalog, if you specified the service name, for instance **db2cdb2inst1** instead of the port number, the service name entry in the Node directory needs to refer to the /etc/service file to get the port number.

The service name information such as service name and its port number has to be stored in the /etc/service file if you specify the service name in the Node directory instead of its port number.

8. The port number which the DB2 client wants to access, in our case **50000**, must much the port that the server instance uses. The value is stored in the /etc/services file at the server. The entry for the port **50000** in our case has the value of service name in our example, **db2cd2inst1**. The value, **db2cd2inst1** in our case in the /etc/services, must be configured in the DBM config file at the server.

We will provide a concrete example for setting up remote access to the WebSphere Administrative Repository in the following sections.

DB2 UDB provides several different methods for configuring a remote client that needs to access a DB2 UDB database server. In this appendix, we explain manual configuration with commands.

I.3.1 Configure the TCP/IP protocol for DB2 UDB

If you have not set up the TCP/IP protocol for DB2 UDB when you installed the DB2 UDB software, you need to configure it before setting up the client. If you have already done this, skip this procedure and go on to the next step.

To enable the TCP/IP protocol, the DB2 UDB server must have the DB2COMM registry variable include TCPIP. For TCP/IP, you also need to update the service name defined in the services file and the DBM configuration file on the database server.

Note

The updating of the services file must be performed on the DB2 UDB server. The services file is updated using an editor and cannot be updated from DB2 UDB.

The parameter in the database manager configuration file is called SVCENAME. The service name is assigned as the main connection port name for the instance and defined in the services file. For example, if the name defined in the services file is `db2cdb2inst1`, the command is the following from the DB2 UDB server:

```
update dbm cfg using svcename db2cdb2inst1
```

After issuing this command, the DB2 UDB instance must be restarted.

I.3.2 Create the database for WebSphere Administrative Repository

Now, you can create a database for the WebSphere Administrative Repository.

From the DB2 UDB server machine, type the following command:

```
create database was

update db cfg for was using applheapsz 256
```

```
$ db2
(c) Copyright IBM Corporation 1993,1999
Command Line Processor for DB2 SDK 6.1.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => create database was
DB20000I  The CREATE DATABASE command completed successfully.
db2 => update db cfg for was using applheapsz 256
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.

db2 =>
```

The command `create database was` will create a database in the /home filesystem. If you want to create a database in a different filesystem, for example, in /wasdb, execute `create database was in /wasdb` instead.

You can specify an alias for the database by specifying a different name for the value of the alias parameter. In our example, we don't specify it. Therefore, the database name and alias are the same.

I.3.3 Create catalogs

You can use the `catalog node` and `catalog db` commands from the client (WebSphere machine) to add a remote database.

Before cataloging a remote database, you should check that the client machine can communicate with the remote DB2 UDB server. Then the client (WebSphere machine) must catalog the remote node, then catalog the remote database.

I.3.3.1 Cataloging a TCP/IP node

When cataloging a TCP/IP node to communicate with a DB2 UDB database server, you must specify a node name after `catalog tcpip node`. This node name will be used in the `catalog db` command and must be unique in the client node directory. In the following section, we will describe this directory in depth.

From the client machine (WebSphere machine), type the following command:

```
catalog tcpip node wasnode remote rs600012 server db2cd2inst1
```

```
$ db2
(c) Copyright IBM Corporation 1993,1999
Command Line Processor for DB2 SDK 6.1.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => catalog tcpip node wasnode remote rs600012 server db2cd2inst1
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2 =>
```

To specify the remote machine, you can use the host name or IP address as the value for the `remote` parameter.

The `server` parameter specifies either the service name or the port number used by the database server. If you specify the service name, it has to be defined in the /etc/service file at the client machine (WebSphere machine). If you specify the port number instead, you don't need to worry about it. These values are located in the /etc/services file (on AIX for example) at the server. If you specify the service name when you create the node catalog, these values should be located in the /etc/services file at the client (WebSphere) also.

```
db2cdb2inst1    50000/tcp   # Connection port for DB2 instance db2inst1
db2idb2inst1    50001/tcp   # Interrupt  port for DB2 instance db2inst1
```

Figure 449. /etc/services on AIX

This identifies which instance of the DB2 UDB server instance to use.

I.3.3.2 Cataloging a remote database

Once the node has been cataloged, the remote database can be cataloged.

From the client machine (WebSphere machine), type the following command:

```
catalog db was at node wasnode
```

```
db2 =>
db2 => catalog db was at node wasnode
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
db2 =>
```

Note

In this appendix, we show you the basic catalog command options. There are many more options you can use. For details on these options please refer to the *DB2 UDB V6.1 Command Reference*.

The database name you use after `catalog db` must match the name of the database alias at the server. You can specify an alias for the remote database by specifying a different name for the value of the `as` parameter. You will then use the alias to connect to the remote database. In our example, we don't specify an alias. Therefore, the database name and alias are the same. The

node name you use for the `at node` parameter must match the node name you used in the `catalog tcpip node` command (in our example, wasnode).

I.3.4 Connect to WebSphere Administrative Repository

Now, you need to test the remote connection which you created. Type the following command from the WebSphere machine:

```
connect to was user db2inst1 using db2inst1
```

```
db2 =>
db2 => connect to was user db2inst1 using db2inst1

   Database Connection Information

 Database server        = DB2/6000 6.1.0
 SQL authorization ID   = DB2INST1
 Local database alias   = WAS

db2 =>
```

I.4 More detailed information for beginners: DB2 directories

Now you can configure remote database. We provide more information which you would like to know.

Accesses to both local and remote databases use entries in the DB2 UDB directories. The directories hide the requirement for the user to know where a database actually resides. Users are able to connect to local databases and remote databases by specifying a database name. The directories that make this possible are:

- System database directory
- Local database directory
- Node directory
- Administration node directory

I.4.1 System database directory

The system database directory resides in the SQLDBDIR subdirectory in the instance directory. This directory is used to catalog both local and remote databases. The directory contains the database name, alias, type and node where the database resides. If the database is local, a pointer to the local

database directory is located in the system database directory. If the database is remote, there is a pointer to the node directory.

I.4.2 Local database directory

The local database directory resides in every drive/path that contains a database. It is used to access local databases in that subdirectory. Each entry in the directory contains the database name, alias, type and location information about the database.

I.4.3 Node directory

Each database client maintains a node directory. The node directory contains entries for all instances that the client will access. The node directory contains communication information about the network connection to the instance. If multiple instances exist on a remote machine, then each instance must be cataloged as a separate node before you are able to access any information contained within the instance.

I.4.4 Administration node directory

The administration node directory contains one definition for each remote system that is known to a DB2 client. Most of the entries for this directory are made during product installation, or by the Control Center or the Client Configuration Assistant.

I.4.5 Examining DB2 UDB directories

We provide a scenario with two systems, a database server and a remote client. From the database server, we issue the `list db directory` command to list the contents of the system database directory. The output is as follows:

```
db2 => list db directory

 System Database Directory

 Number of entries in the directory = 5

Database 1 entry:

 Database alias                 = WAS
 Database name                  = WAS
 Local database directory       = /home/db2inst1
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0

Database 2 entry:

 Database alias                 = SESSION
 Database name                  = SESSION
 Local database directory       = /home/db2inst1
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Indirect
 Catalog node number            = 0
```

You can see that there are 5 databases cataloged on the server. The first database is the `WAS` database.

The local database directory is set to `/home/db2inst1`. If you want to examine this in more detail, you could examine the local database directory by using the command: `list db directory on /home/db2inst1`. This provides more detailed information regarding the location of some of the database files.

Next, you examine the contents of two directories on the client machine (WebSphere machine): the system database directory and the node directory. A client (WebSphere machine) does not have a local database directory since it does not usually contain local databases. The command from the client machine (WebSphere machine) to examine the system database directory is also `list db directory`. Here is the output:

```
db2 => list db directory

 System Database Directory

 Number of entries in the directory = 3

Database 1 entry:

 Database alias                 = WAS
 Database name                  = WAS
 Node name                      = WASNODE
 Database release level         = 9.00
 Comment                        =
 Directory entry type           = Remote
 Catalog node number            = -1

Database 2 entry:

 Database alias                 = SESSION
...
```

The database alias name of the database on the client (WebSphere machine) is called WAS. This name corresponds to the actual database name on the server (WAS). Note that the directory entry type is **indirect** on the server for the WAS database and the type is **remote** on the client.

Note

When a database is created, an entry is automatically placed in the system database directory and the local database directory on the server.

When cataloging the remote database on the client, the database name on the client must match the database alias on the server. The database alias specified on the client workstation is used in the `connect` statement. In our example, we don't specify an alias. Therefore, the name used to connect to the WAS database is WAS.

There is always an associated node name with a client (WebSphere machine) system database directory entry. The node name (in our example wasnode) defines the location of the DB2 UDB server (rs600012, in our case) on the network.

To discover more information about the location of the database server, we examine the WASNODE node entry in the node directory using the `list node directory` command. The output is as follows:

```
db2 => list node directory

 Node Directory

 Number of entries in the directory = 1

Node 1 entry:

 Node name                      = WASNODE
 Comment                        =
 Protocol                       = TCPIP
 Hostname                       = rs600012
 Service name                   = db2cd2inst1

db2 =>
```

The WASNODE node name corresponds to a TCP/IP connection. The TCP/IP host name for the DB2 UDB server is rs600012 and the service name is db2cd2inst1. The TCP/IP information in the client's node directory must match with the information in the server's DBM configuration file.

Appendix J. Special notices

This publication is intended to help I/T specialists to design and configure scalable Web application servers using WebSphere Application Server Advanced Edition. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Application Server. See the PUBLICATIONS section of the IBM Programming Announcement for WebSphere Application Server for more information about what publications are considered to be product documentation.

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. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have

been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX
AS/400
DB2
DRDA
IBM
IBM.COM
Netfinity
OS/400
Redbooks
Redbooks logo
RS/6000
SecureWay
SP
System/390
TXSeries
VisualAge
WebSphere

The following terms are trademarks of other companies:

Lotus and Domino are trademarks of Lotus Development Corporation.

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet

Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

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

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix K. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

K.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 571.

- *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher*, SG24-5858
- *V4 TCP/IP for AS/400: More Cool Things Than Ever*, SG24-5190
- *Building AS/400 Applications for WebSphere Standard 2.0*, SG24-5635
- *Building AS/400 Appls for WebSphere Advanced 3.0*, SG24-5691 (available as a “redpiece” at ibm.com/redbooks)

K.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at ibm.com/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

K.3 Other resources

These publications are also relevant as further information sources:

- *AS/400 Software Installation*, SC41-5120
- *HTTP Server for AS/400 Webmaster’s Guide V4R4*, GC41-5434

K.4 Referenced Web sites

The following Web sites are also relevant as further information sources:

- www.as400.ibm.com/products/websphere/docs/as400v302/docs/index.html
- www.as400.ibm.com/products/websphere/docs/doc.html#AE302
- www.as400.ibm.com/products/websphere/docs/relnotes302_addendum.html

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

S

T

U

V

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6153-00 WebSphere Scalability - WLM and Clustering Using WebSphere Application Server Advanced Edition
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

Redbooks

WebSphere Scalability: WLM and Clustering

Using WebSphere Application Server Advanced Edition

(1.0” spine)
0.875”<->1.5”
460 <-> 788 pages

WebSphere Scalability: WLM and Clustering

Using WebSphere Application Server Advanced Edition

How to scale IBM WebSphere Application Server Advanced Edition

Sample configurations help you to start your WebSphere site

Examines several techniques, including WLM

This redbook discusses various options for scaling applications based on WebSphere Application Server, Advanced Edition. The objective of this book is to explore how the basic configuration can be extended to provide more computing power, by better exploiting the power of each machine, and by using multiple machines. It examines a number of techniques, including:

- Cloning and pooling of multiple Java Virtual Machines
- Distributing the load from one Web server to multiple Application Servers, using Servlet Redirector and OSE Transports
- Using IBM Network Dispatcher to distribute the load between multiple Web servers
- Using the EJB Workload Management facility (WLM) to distribute load at the EJB level
- Distributing session state using the Session Clustering facility

This book shows step-by-step procedures for the UNIX, Windows NT and AS/400 platforms. Both simple and advanced configurations are covered.

This has been done using WebSphere Application Server V3.021 (OS/400, UNIX and Windows NT) and V3.5 (UNIX and Windows) Advanced Edition.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6153-00 ISBN 0738418285