2

Installation and Configuration Overview

As you learned in Chapter 1, the architectural variations you can construct using WebSphere Application Server V6 range from the very simple to the fairly complex. You can build any of the architectures we described in that chapter by following the steps outlined in the appropriate chapter or chapters of this book.

To demonstrate the steps involved in setting up a complete WebSphere Application Server installation, this book walks you through the configuration process for the most sophisticated architecture you learned about in Chapter 1: a highly available, workload-managed (HA/WLM), clustered environment built using the WebSphere V6 Network Deployment package. During the course of this book, you'll assume all the responsibilities and roles required to implement this architecture, including setup of a standalone application server (Chapters 3 and 4); IBM HTTP Server and the plug-in module (Chapters 5 and 6); the Network Deployment package (Chapter 7); node federation (Chapter 8); the HTTP Server distributed plug-in (Chapter 9); clustering (Chapter 10); HTTP session persistence (Chapter 11); the Service Integration Bus, messaging engine, and highly available persistent service (Chapter 12); and Edge Server – Load Balancer (Chapters 13 and 14). You'll learn how to build each piece of the HA/WLM architecture one at a time, step by step, adding each component in a logical way to successfully configure this system.

The steps described in Chapters 3 through 6 are performed using the Base/Express package of WebSphere Application Server. Chapters 7 and beyond require the Network Deployment package. In Chapters 15 through 26, you'll learn about some advanced topics, including security, deployment issues, and management of the WebSphere Application Server environment.

The HA/WLM Architecture

Figure 2-1 shows the sample highly available, workload-managed WebSphere architecture you're going to build. Building this system involves the following high-level tasks:

- Install and configure WebSphere in a cluster environment using the Network Deployment package.
- Configure the WebSphere cluster for memory-to-memory or database session persistence.
- Install and configure HTTP Server and the plug-in module to spray requests across application servers.
- Install and configure Edge Server – Load Balancer servers (primary and secondary) to spray requests across HTTP servers, and configure these servers for high availability.

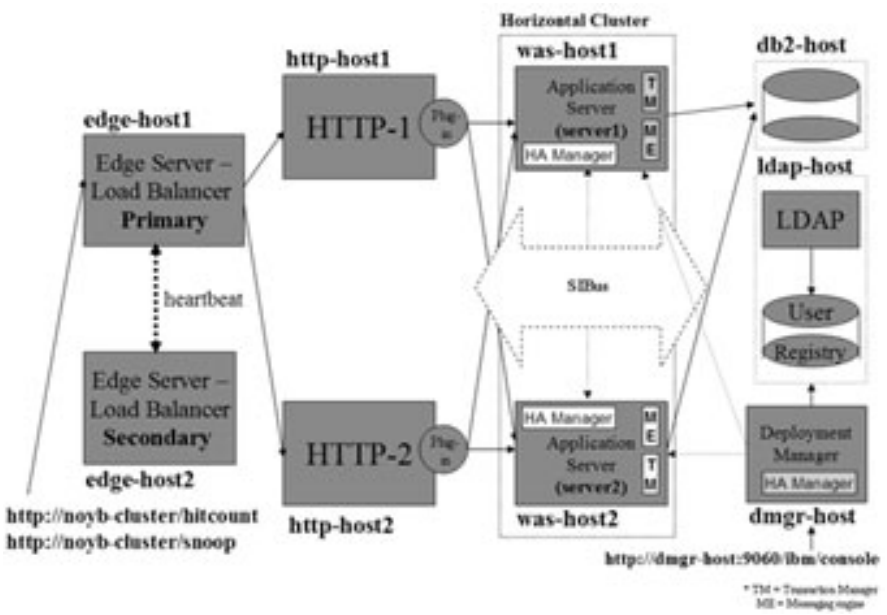

Figure 2-1: Sample HA/WLM WebSphere architecture

As you set up this system, you'll perform additional tasks to verify it, ensuring that the basic WebSphere infrastructure is foolproof and ready for application deployment. During the verification process, you'll answer the following questions about your configuration:

- Is application server failover working?
- Is HTTP session failover (memory-to-memory or database persistence) working?
- Is IBM HTTP Server (with the help of the plug-in module) spraying requests across WebSphere application servers?
- Is HTTP Server failover working?
- Is Edge Server – Load Balancer spraying requests across HTTP servers?
- Is Edge Server – Load Balancer failover working in high-availability mode?

In the rest of this chapter, we give an overview of the specific steps you'll perform to build the sample HA/WLM configuration. Figure 2-2 lists these steps. As we discuss each step in more detail, a corresponding flow chart will show you the high-level decisions and substeps required to accomplish that step.

Step 1: Verify you're ready (Chapter 2)
Step 2: Perform pre-installation tasks (Chapter 2)
Step 3: Create, configure, and verify the deployment manager profile (Chapter 7)
Step 4: Create, configure, and verify the application server profile (Chapter 7)
Step 5: Create, configure, and verify the custom profile (Chapter 7)
Step 6: Federate nodes (Chapter 8)
Step 6a: Federate the node within the application server profile
Step 6b: Federate the node within the custom profile
Step 7: Install, configure, and verify IBM HTTP Server (Chapter 5)
Step 8: Install the distributed remote plug-in (Chapter 9)
Step 8a: Configure HTTP Server node as managed node
Step 8b: Configure HTTP Server node as unmanaged node
Step 9: Create and configure the horizontal cluster (Chapter 10)
Step 10: Enable and configure highly available persistent service (Chapter 12)
Step 11: Configure HTTP session persistence (Chapter 11)
Step 11a: Configure memory-to-memory session persistence
Step 11b: Configure database session persistence
Step 12: Create and configure SIBus and messaging engine (Chapter 12)
Step 13: Install, configure, and verify Edge Server – Load Balancer (Chapters 13 and 14)
Step 13a: Install, configure, and verify Edge Server – Load Balancer servers
Step 13b: Configure Edge Server – Load Balancer servers for high availability

Figure 2-2: Installation and configuration steps

Step 1: Verify You're Ready

Figure 2-3 shows the flow chart you'll follow for Step 1.

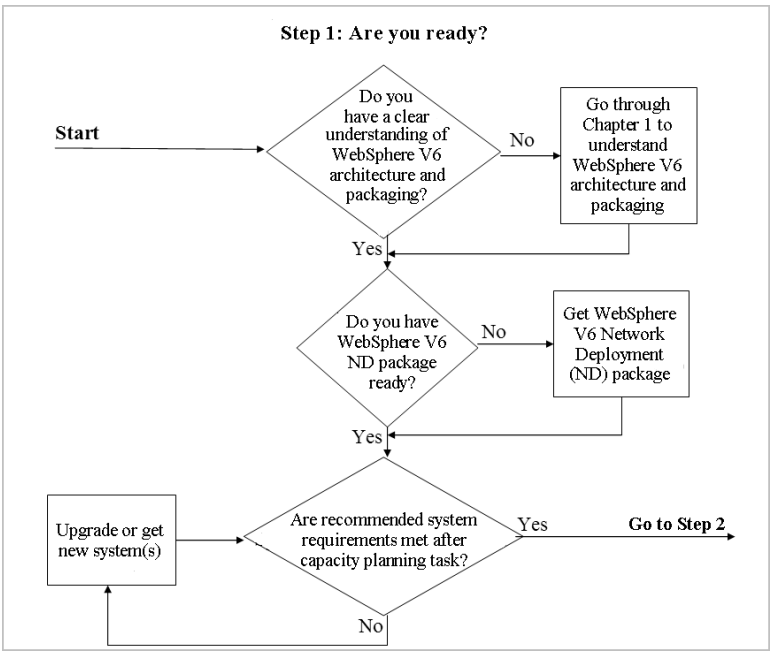

Figure 2-3: Step 1 – Verify you're ready

A successful WebSphere Application Server installation requires a clear understanding of the V6 architecture and packaging. After reading Chapter 1, you should have a good grasp of the capabilities and limitations of each package of WebSphere V6 (Express, Base, and Network Deployment). For example, you can't expect to build a WebSphere cluster environment (as for the HA/WLM example) using the Base or Express package. You can build less complex environments using the Base or Express package, as you'll see in Chapters 3 through 6. To build application server clusters, you need the Network Deployment package. Make sure you obtain the right WebSphere package for the architecture you're going to build.

After obtaining the WebSphere software, make sure the systems on which you plan to install the software will be able to run it (and your application) in the environment you require (e.g., stress test, pre-production, production). If possible, perform capacity planning to ensure you can meet any service level agreements in place. Also, be realistic:

Although a system with the minimum hardware requirements (in terms of CPU, RAM, and so on) may suffice for debugging in development, training, or functional testing environments, it's likely to prove inadequate for performance testing or production environments.

To determine the minimum operating system and hardware requirements for your platform and WebSphere package, consult one of the following Web sites:

http://www-306.ibm.com/software/webservers/appserv/doc/latest/prereq.html
http://www-306.ibm.com/software/webservers/appserv/was/requirements

Step 2: Perform Pre-Installation Tasks

Figure 2-4 shows the flow chart you'll follow for Step 2.

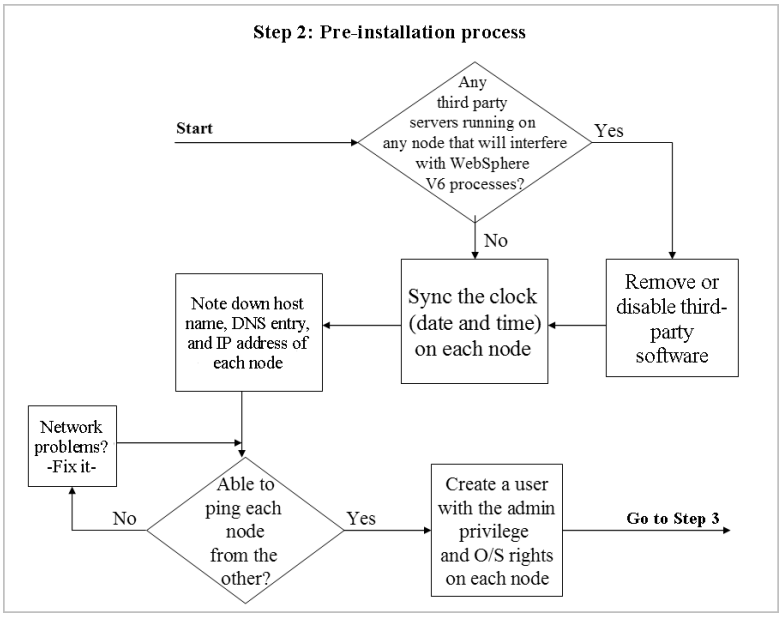

Figure 2-4: Step 2 – Perform pre-installation tasks

Before you actually install the WebSphere software, be sure to check the required configuration and administrative user privileges on each system. Doing so will save time you might otherwise end up spending on problem determination and troubleshooting.

In addition, make sure your TCP/IP network is configured properly on the machine on which you'll be installing the product. The directory structure (core directory naming) of the WebSphere administrative configuration repository uses the host name of the machine on which the WebSphere software is installed, so make sure the host name doesn't change after the installation.

If any software on your system will interfere with the TCP/IP ports used by WebSphere processes and you don't know about it, you'll spend quite a bit of time troubleshooting later. For example, you'll have trouble running HTTP Server on a machine on which a third-party Web server (e.g., Microsoft's Internet Information Server, or IIS) is already installed and running. Web servers use port 80 by default (or port 443 if you've enabled Secure Sockets Layer, or SSL). If you need to have multiple HTTP Server versions on a single physical machine, you must proactively address port-assignment issues to avoid port conflicts during the installation process. Table 2-1 lists the important port numbers used by WebSphere processes. For a complete list of WebSphere ports, consult the WebSphere Application Server V6 Information Center at http://publib.boulder.ibm.com/infocenter/ws60help/index.jsp.

Table 2-1: Default ports used by WebSphere processes

Port description	Default port number
HTTP transport port	80
HTTPS transport port	443
Embedded HTTP transport port	9080
Admin console port	9060
Embedded HTTPS transport port	9443
Admin console secure port	9043
Bootstrap port	2809
SOAP connector port	8880
SAS SSL ServerAuth port	9401
CSIV2 SSL ServerAuth listener port	9403
CSIV2 SSL MutualAuth listener port	9402
ORB listener port	9100
High Availability Manager communications port	9353
Service Integration Bus (SIB) port	7276
SIB secure port	7286
SIB MQ interoperability port	5558
SIB MQ interoperability secure port	5578

To see which ports are being used on your system, run the command **netstat –a** from the operating-system command prompt. Figure 2-5 shows sample output from the netstat command. Before proceeding with the WebSphere installation, remove or disable any third-party software that's likely to cause a conflict.

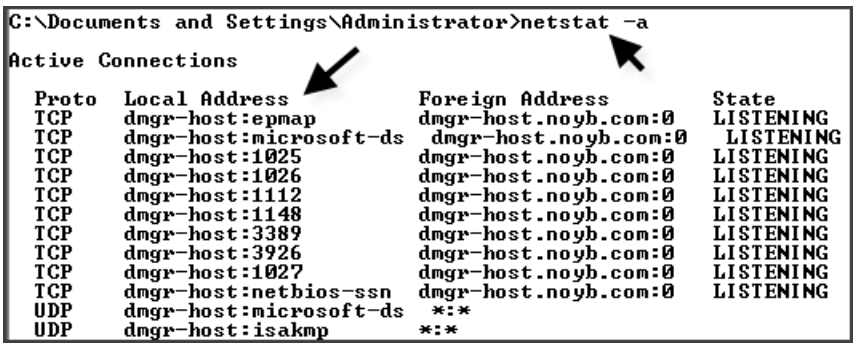

Figure 2-5: Sample netstat command output

Because you'll be using the Network Deployment package to configure multiple nodes, you may encounter synchronization problems if the date and time (clock) on the application server node differ from the date and time on the Deployment Manager node to which you'll be federating the application server. To avoid such problems, synchronize the clock on each node. Also, make sure the timestamp between the Lightweight Directory Access Protocol (LDAP) server and the Deployment Manager are in sync with each other.

For each node, make a note of the host name, fully qualified name (i.e., Domain Name Server, or DNS, entry), and IP address. Be sure you can ping the nodes from each other. In some cases, you may be able to ping a WebSphere node from the HTTP server node but have trouble pinging in reverse. For help fixing this problem, consult your network administrator. It's important to solve any such issues before proceeding with the WebSphere installation.

Note: In our experience, it's best, if you can manage it, to use DNS instead of mapping host names to IP addresses through the hosts file. The latter approach can be error-prone as well as a maintenance nightmare. If you do use the hosts file, be sure to periodically validate the entries in the file and update it if a domain name, host name, or IP address changes. Otherwise, you may spend quite a bit of time troubleshooting in the wrong place. Consult your network administrator about the best way to manage the hosts file.

If a DNS entry isn't available, make sure a host name entry with the IP address exists in the etc/hosts file. If DNS entries aren't set, update the hosts file. On Windows XP systems, you'll find the hosts file in the \WINDOWS\system32\drivers\etc directory. On Unix machines, you'll find it in directory /etc.

To install and configure WebSphere software, you must have administrative privileges and certain operating system rights. For example, to install HTTP Server and WebSphere on Windows platforms, you need rights to

- act as part of the operating system
- log on as a service

On AIX, Linux, and Unix systems, you must log in as root. Before proceeding with the installation, make sure you have the appropriate rights for your operating system environment, or create the appropriate user.

Step 3: Create, Configure, and Verify the Deployment Manager Profile

Figure 2-6 shows the flow chart you'll follow for Step 3.

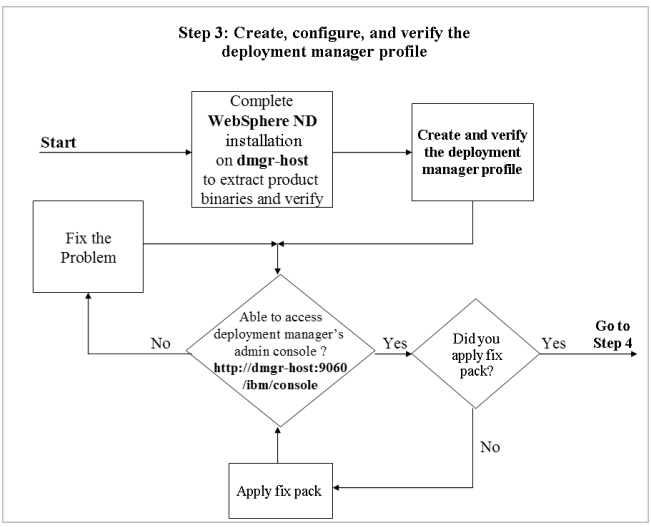

Figure 2-6: Step 3 – Create, configure, and verify the deployment manager profile

You use the Deployment Manager to manage clusters and their cluster members, Web servers and their nodes, and the entire cell configuration. To set up the Deployment

Manager, you'll need to install the WebSphere Network Deployment package on the dmgr-host system, create the deployment manager profile, and verify that you're able to connect to the Deployment Manager's administrative console. Chapter 7 explains these tasks in detail.

Figure 2-7 shows what the Deployment Manager's admin console looks like after the successful creation of the deployment manager profile on dmgr-host.

Figure 2-7: Deployment Manager's admin console

Step 4: Create, Configure, and Verify the Application Server Profile

Figure 2-8 shows the flow chart you'll follow for Step 4.

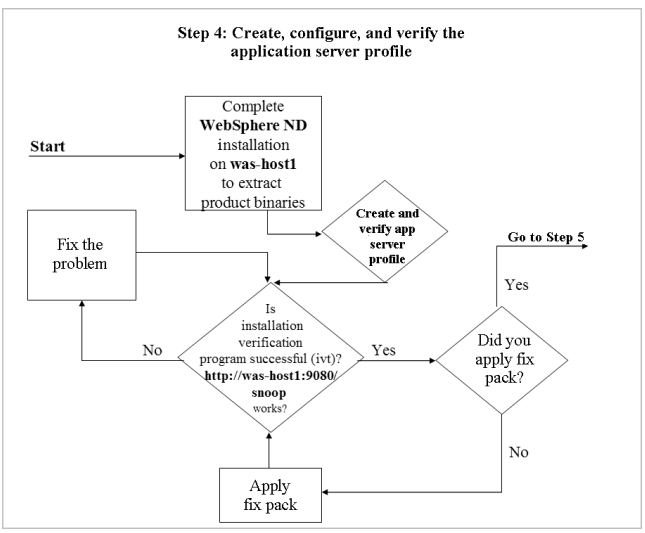

Figure 2-8: Step 4 – Create, configure, and verify the application server profile

Each application you deploy will run on an application server. When you create an application server profile on the was-host1 system, an application server (named server1) is created automatically. When you later create the horizontal cluster, you'll use this application server as a template and also as the first member of the cluster.

Chapter 7 describes how to create, configure, and verify the application server profile on was-host1. Figure 2-9 shows the output of the snoop servlet invoked through the Web container's default HTTP transport port (9080) after the successful creation of the application server profile on was-host1.

Figure 2-9: Snoop servlet output confirming creation of application server profile on was-host1

Step 5: Create, Configure, and Verify the Custom Profile

Figure 2-10 shows the flow chart you'll follow for Step 5.

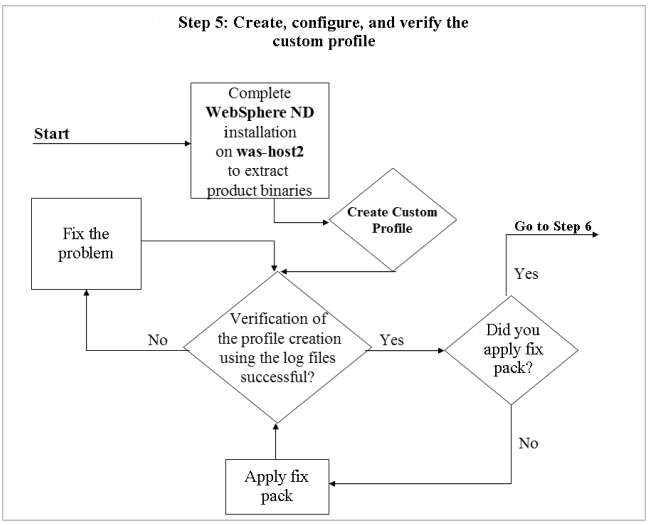

Figure 2-10: Step 5 – Create, configure, and verify the custom profile

In this step, you'll create a custom profile on was-host2 and federate the node within this profile to the Deployment Manager cell. You'll create the second cluster member (server2) when you create the horizontal cluster later.

Chapter 7 explains how to create and verify the custom profile on was-host2. Figure 2-11 shows the contents of the log file (<WASV6-ROOT>\logs\wasprofile\wasprofile_create_Custom01.log) after the successful creation of the custom profile on was-host2.

```
<level>INFO</level>
<class>com.ibm.ws.profile.cli.WSProfileCLICreateProfileInvoker</class>
<method>executeWSProfileAccordingToMode</method>
<thread>11</thread>
<message>INSTCONFSUCCESS: Success: The profile now exists.</message>
```

Figure 2-11: Log file contents after successful creation of custom profile

Step 6: Federate Nodes

When you initially create the Deployment Manager on the dmgr-host system, it's not aware of any application server profile on the was-host1 node or any custom profile on the was-host2 node. To manage the application servers and nodes from the Deployment Manager's admin console, you must first federate those nodes to the Deployment Manager cell on dmgr-host.

Chapter 8 details the steps involved in federating the node within the application server profile on was-host1 to the Deployment Manager cell on dmgr-host. Figure 2-12 shows the flow chart for this task.

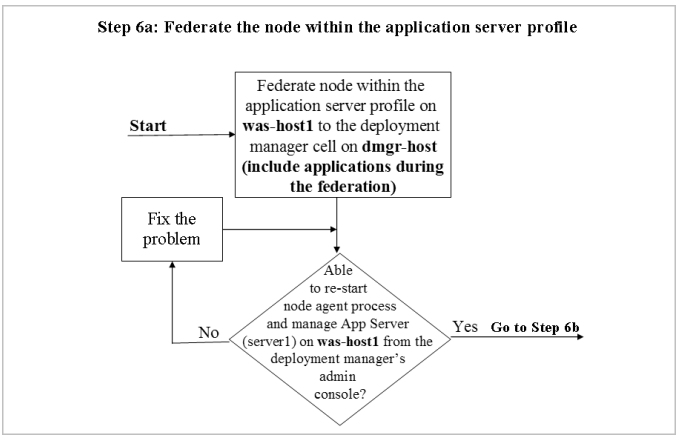

Figure 2-12: Step 6a – Federate the node within the application server profile

Chapter 8 also describes how to federate the node within the custom profile on was-host2 to the Deployment Manager cell on dmgr-host. Figure 2-13 shows the flow chart for this task.

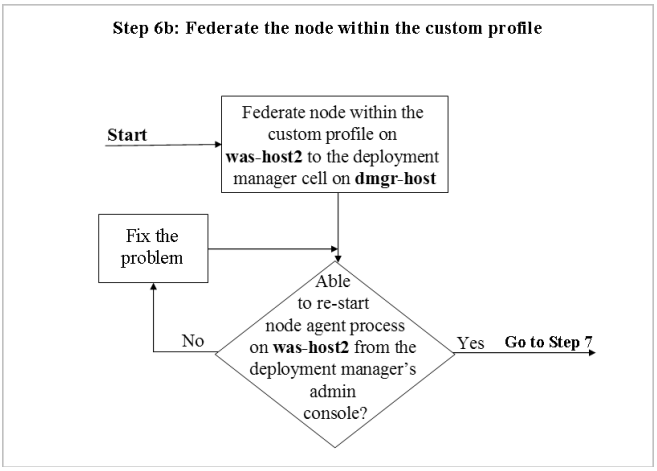

Figure 2-13: Step 6b – Federate the node within the custom profile

The screen in Figure 2-14 shows how the was-host1 and was-host2 nodes appear in the Deployment Manager's admin console after the two nodes have been successfully federated to the Deployment Manager cell on dmgr-host.

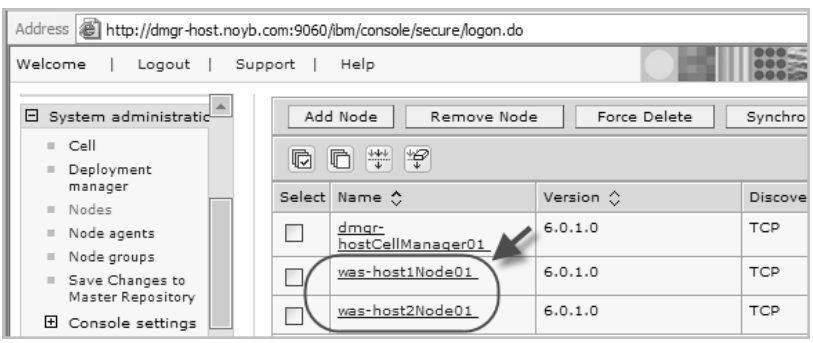

Figure 2-14: Nodes displayed in Deployment Manager admin console

Figure 2-15 shows the Deployment Manager admin console view of the node agent processes on these two nodes. Once you've successfully federated the nodes, you can use this display to manage (e.g., stop, start) the node agents and servers on each node.

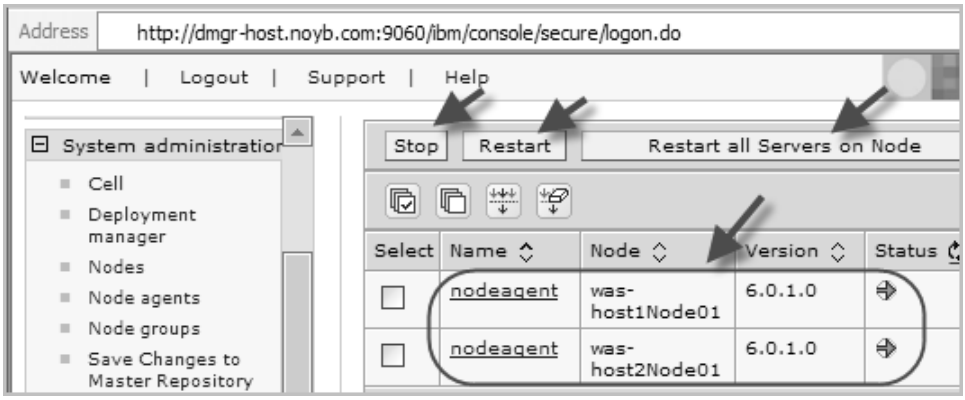

Figure 2-15: Node agent processes

Step 7: Install, Configure, and Verify IBM HTTP Server

Figure 2-16 shows the flow chart you'll follow for Step 7.

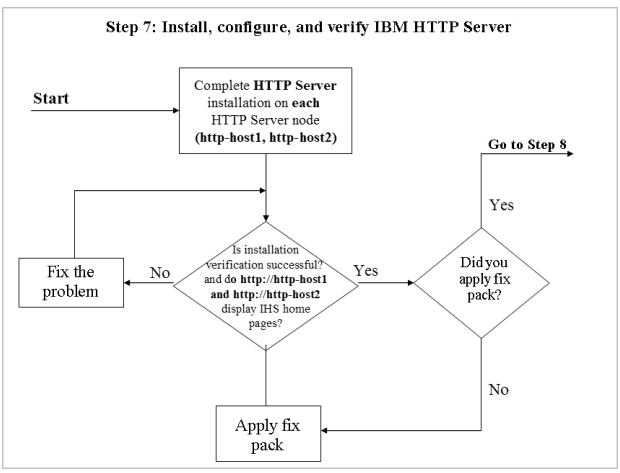

Figure 2-16: Step 7 – Install, configure, and verify IBM HTTP Server

You'll install IBM HTTP Server on two systems (http-host1 and http-host2). Later (in Step 8), you'll install the WebSphere plug-in module on each of these Web servers and configure it with the Deployment Manager to spray requests across application servers. You'll also configure both HTTP servers with Edge Server – Load Balancer to provide high availability for these servers.

Chapter 5 explains how to install, configure, and verify HTTP Server on the http-host1 and http-host2 nodes. Figure 2-17 shows the HTTP Server welcome page that appears once you've successfully installed HTTP Server on http-host1. You'll see a similar result after installing HTTP Server on http-host2 (using http://http-host2.noyb.com).

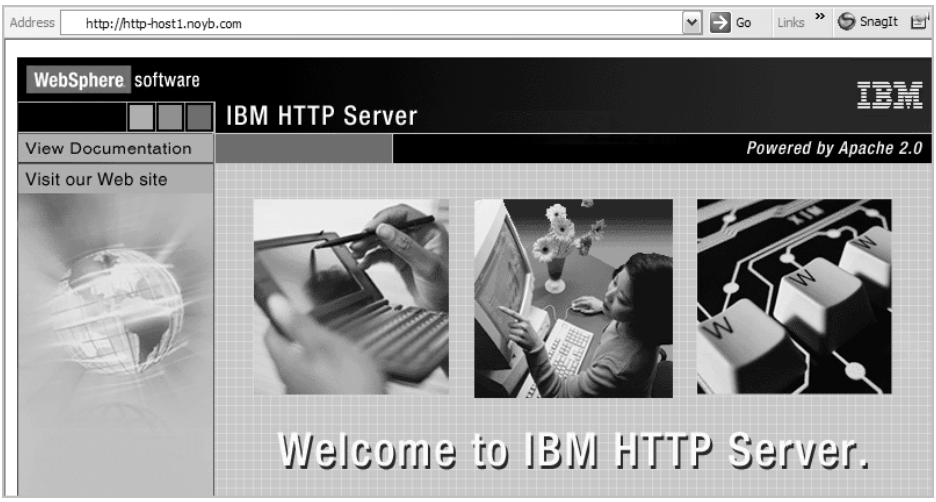

Figure 2-17: HTTP Server welcome page

Step 8: Install the Distributed Remote Plug-in

You can propagate the plug-in configuration file (plugin-cfg.xml) and manage the HTTP servers from the Deployment Manager's admin console. This feature necessitates some upfront planning and requires you to make some configuration changes in the Deployment Manager and on the HTTP server nodes after installing the plug-in. The configuration changes depend on the plug-in architecture you use. Because in this case you're dealing with a remote plug-in, you must choose which of two remote plug-in architectures you want to configure:

- *Distributed remote plug-in – Web server as managed node* — A managed node contains a node agent process that enables you to manage an HTTP server remotely from the Deployment Manager's admin console. You create a node agent process on each HTTP server/plug-in node (http-host1 and http-host2) by creating a custom profile on each node and then federating the node within the custom profile to the Deployment Manager cell on dmgr-host. (This configuration isn't recommended for production environments because the node agent runs in the demilitarized zone, or DMZ, on the Web server node, and running a full-Java process in the DMZ is not a good practice from a security perspective.)

- *Distributed remote plug-in – Web server as unmanaged node* — In the case of IBM HTTP Server, you can use the IBM HTTP administrative server to manage the unmanaged Web server (without a node agent process). Even though it's called an unmanaged node, if you're using IBM HTTP Server, you have the same capabilities as a managed node as far as Web server management from the admin console is concerned.

> **Note:** If you're going through this process simply for educational and training purposes, consider configuring http-host1 as a managed node and http-host2 as an unmanaged node to try out both of these configurations.

Figure 2-18 shows the flow chart you follow to configure the first option, distributed remote plug-in – managed node. Figure 2-19 shows the flow chart for the second option, distributed remote plug-in – unmanaged node. Chapter 9 explains how to install the WebSphere plug-in on each HTTP Server node, how to configure each node as either a managed or an unmanaged node, and how to verify the configuration.

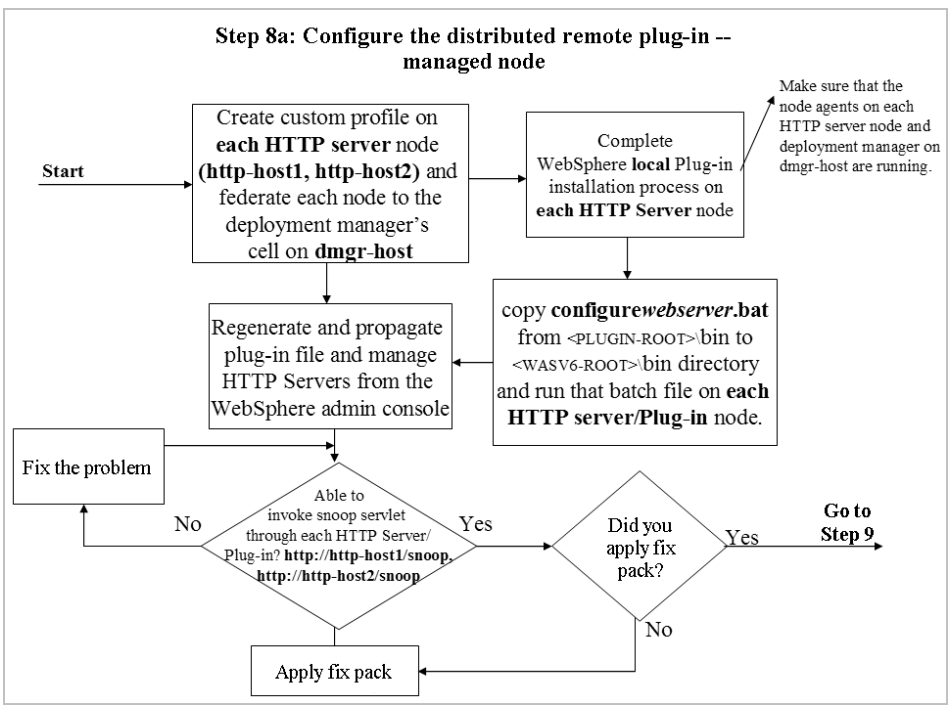

Figure 2-18: Step 8a – Configure distributed remote plug-in – managed node

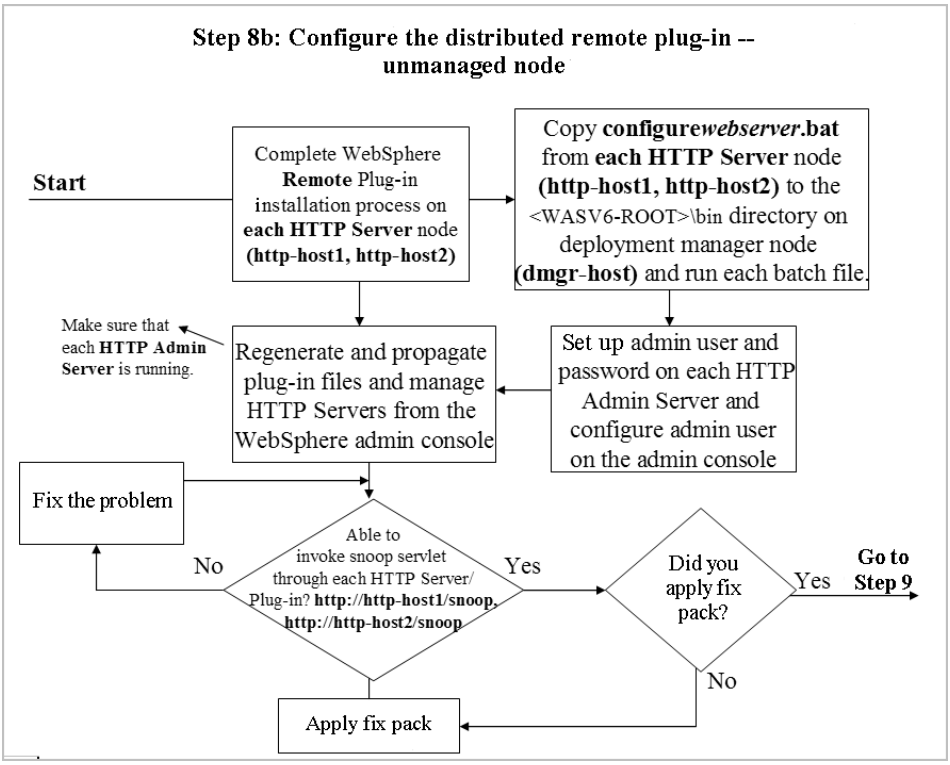

Figure 2-19: Step 8b – Configure distributed remote plug-in – unmanaged node

Figure 2-20 shows the result of invoking the snoop servlet through the HTTP Server/plug-in (http://http-host1.noyb.com/snoop) once the plug-in software has been successfully installed and configured on http-host1. You'll receive a similar result after successfully installing and configuring the plug-in on http-host2 (using http://http-host2.noyb.com/snoop).

Figure 2-20: Snoop servlet output

Once you've successfully installed and configured the plug-in software on the HTTP Server nodes, you can manage the HTTP servers remotely and change the HTTP Server configuration and plug-in properties from the Deployment Manager's admin console, as shown in Figure 2-21.

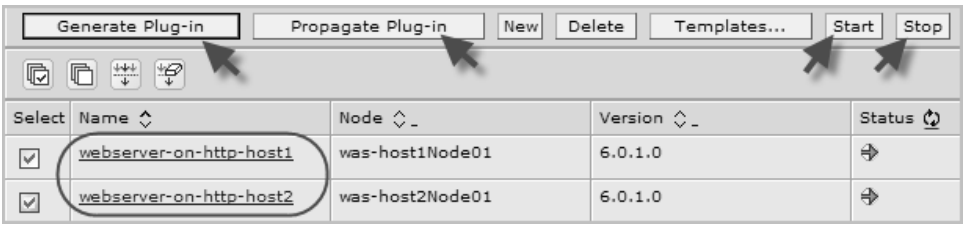

Figure 2-21: Managing HTTP servers and plug-in properties from the Deployment Manager's admin console

Step 9: Create and Configure the Horizontal Cluster

Figure 2-22 shows the flow chart you'll follow for Step 9.

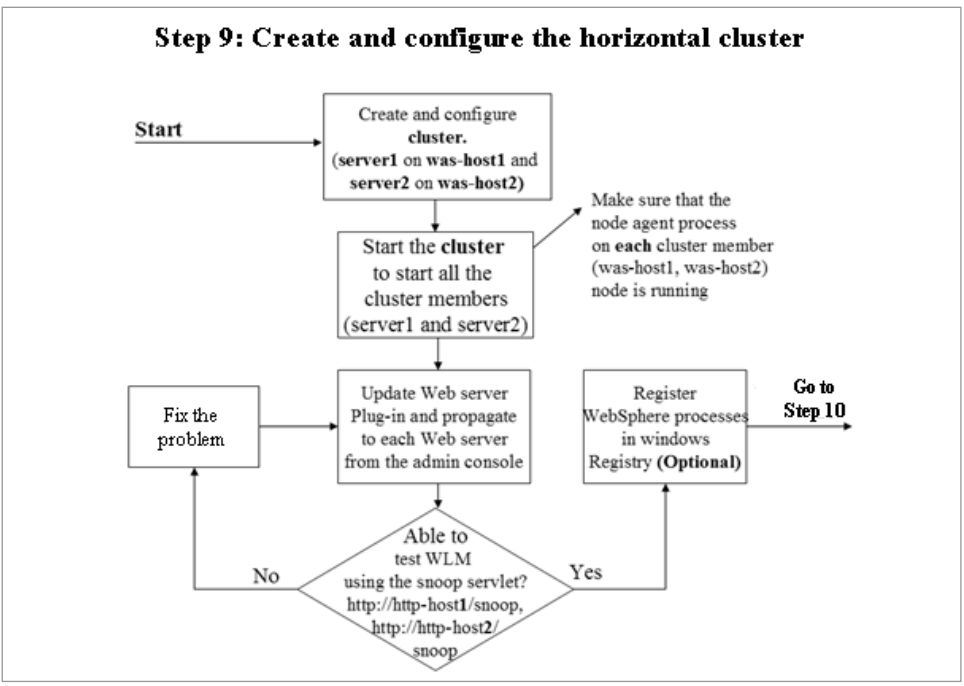

Figure 2-22: Step 9 – Create and configure the horizontal cluster

When you create the horizontal cluster, you'll use the application server named server1 on was-host1 as a template and as the first cluster member, and you'll create server2 on was-host2 as a second cluster member. Chapter 10 explains how to create, configure, and verify the horizontal cluster. If you're building this system in a Windows environment, you can optionally register and customize the service names in the Windows **Services** panel by following the instructions given in Chapter 20.

The partial screens in Figures 2-23 and 2-24 show that the client requests are being workload-managed across cluster members (server1 on was-host1 and server2 on was-host2) from the plug-in on the HTTP server node (http-host1 or http-host2). You should see the same behavior whether you use the URL http://http-host1/snoop or http://http-host2/snoop in the browser. The first figure shows that the first snoop servlet request is sent to and processed by server1 on was-host1 when a user issues either http://http-host1/snoop or http://http-host2/snoop from the browser. Figure 2-24 shows that the second snoop servlet request is sent to and processed by server2 on was-host2 when the user issues either of these requests from the browser.

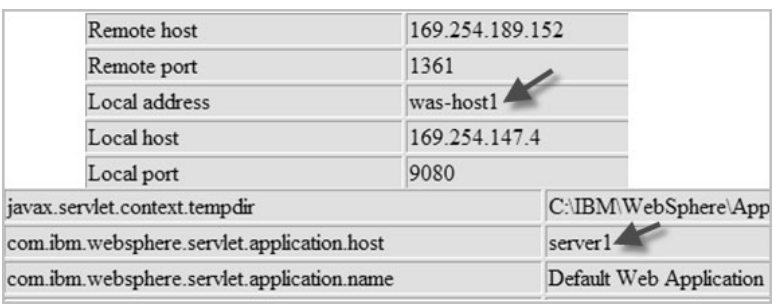

Figure 2-23: First snoop servlet request

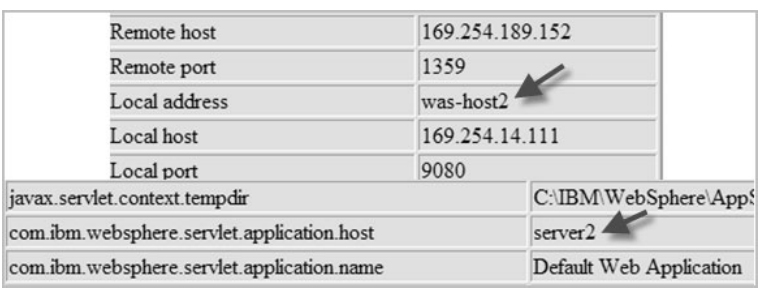

Figure 2-24: Second snoop servlet request

Step 10: Enable and Configure Highly Available Persistent Service

Figure 2-25 shows the flow chart you follow to enable, configure, and verify highly available persistent service. If you want to recover in-flight transactions automatically should the cluster member that's processing the transactions fail, you need to enable and configure this service for the cluster. Chapter 12 explains this process.

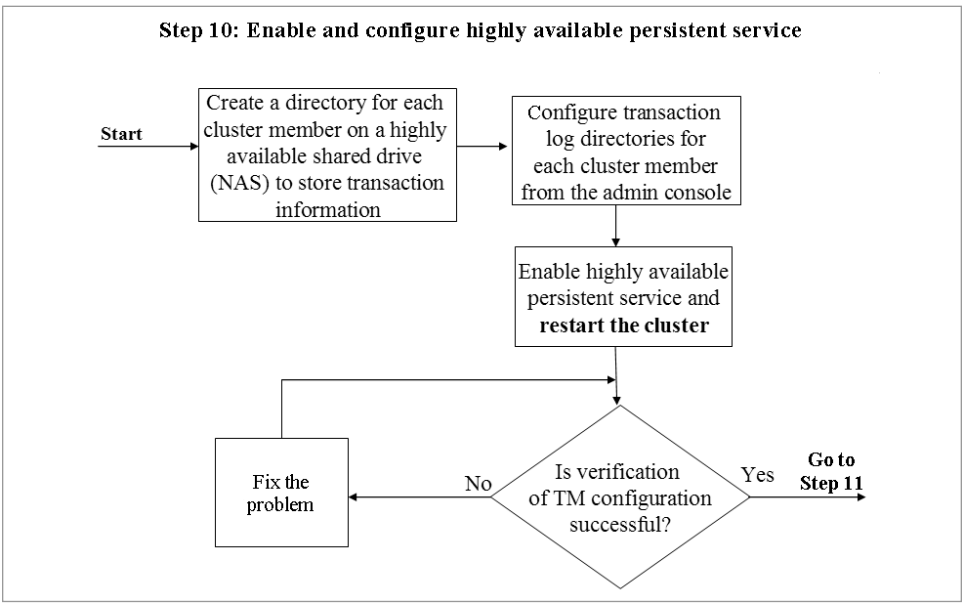

Figure 2-25: Step 10 – Enable and configure highly available persistent service

Step 11: Configure HTTP Session Persistence

You have two choices when it comes to configuring HTTP session persistence for session failover. You can choose to configure HTTP session persistence by storing the session data in another member of that cluster (known as *memory-to-memory* session persistence) or by storing the data in a database as a backup (*database* session persistence). If you like, you can configure the cluster members to use both memory-to-memory (JVM) and database session persistence, but you can enable only one of these options (or neither) at any one time.

Configure Memory-to-Memory Session Persistence

Figure 2-26 shows the flow chart you follow to configure memory-to-memory session persistence. When you configure cluster members for memory-to-memory session

persistence and set the replication mode to "Both client and server," server1 (the first cluster member) on was-host1 saves the session data created on server2 (the second cluster member) on was-host2 for backup. In the same way, server2 saves the session data created on server1 as a backup to provide session failover capability.

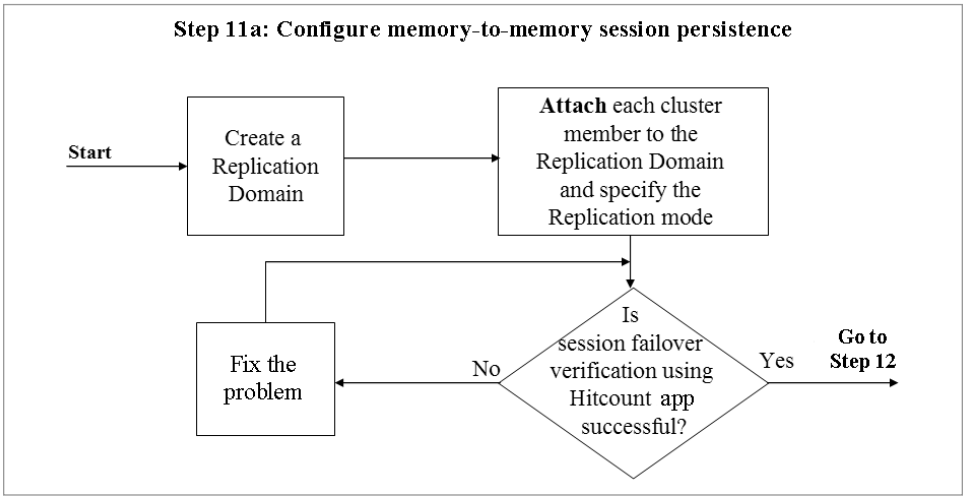

Figure 2-26: Step 11a – Configure memory-to-memory session persistence

Chapter 11 describes how to configure memory-to-memory session persistence and verify session failover. Figure 2-27 shows the session manager configuration on the first cluster member after you've successfully created the replication domain and attached each cluster member to it. You'll see similar session-manager configuration information on the second cluster member (server2) and on any other cluster members you choose to configure.

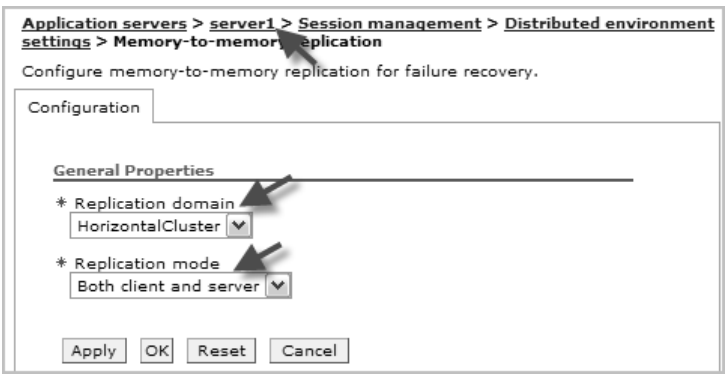

Figure 2-27: Session manager configuration on first cluster member

Configure Database Session Persistence

Figure 2-28 shows the flow chart you follow to configure database session persistence.

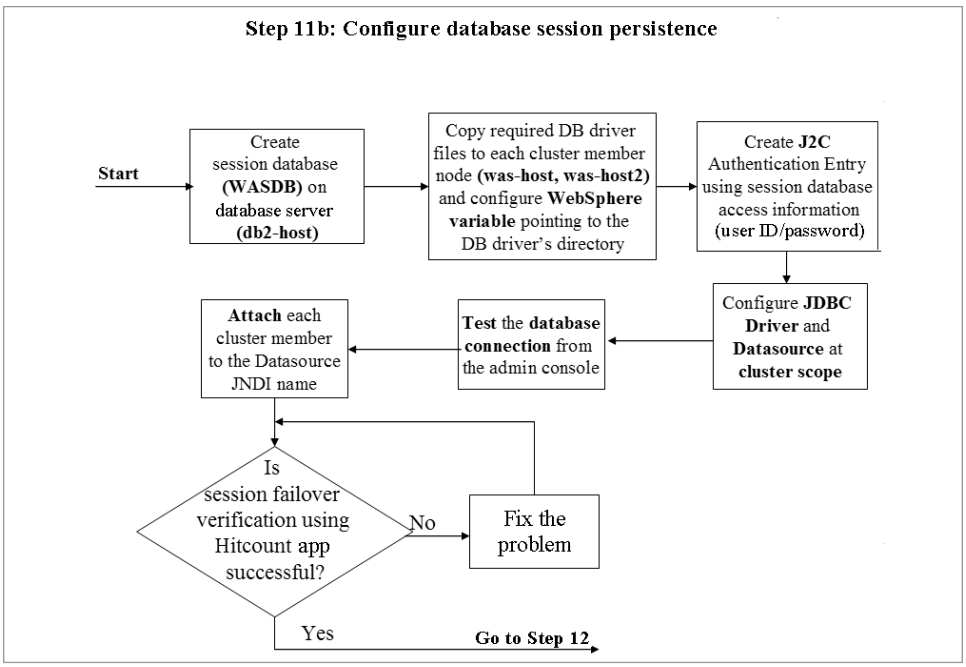

Figure 2-28: Step 11b – Configure database session persistence

When you configure a database for session persistence, all session data created on both cluster members (server1 and server2) is stored in the database as a backup for session failover capability. Chapter 11 explains how to configure database session persistence and verify the session failover. Figure 2-29 shows a successful database connection after the cluster has been configured for database session persistence from the Deployment Manager's admin console.

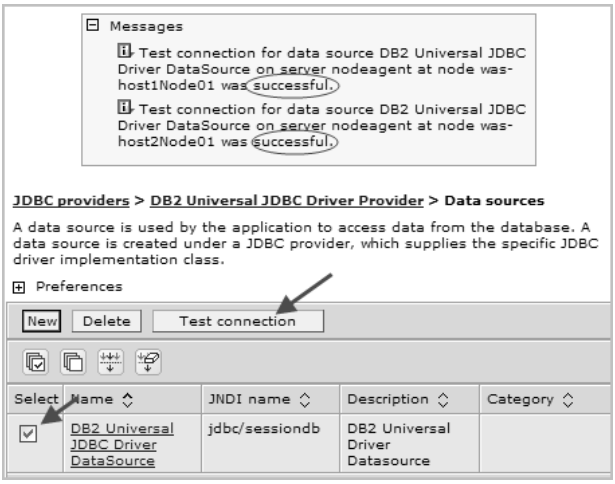

Figure 2-29: Successful database connection

Step 12: Create and Configure the SIBus and Messaging Engine

Figure 2-30 shows the flow chart you follow to create, configure, and verify the Service Integration Bus (SIBus) and messaging engine. If you plan to deploy message-based (JMS/MDB-based) applications, you need to create an SIBus and add the cluster as a member of the bus to create a highly available messaging engine on that cluster. Chapter 12 explains how to create, configure, and verify these components. If you're not deploying message-based applications, you can skip this step.

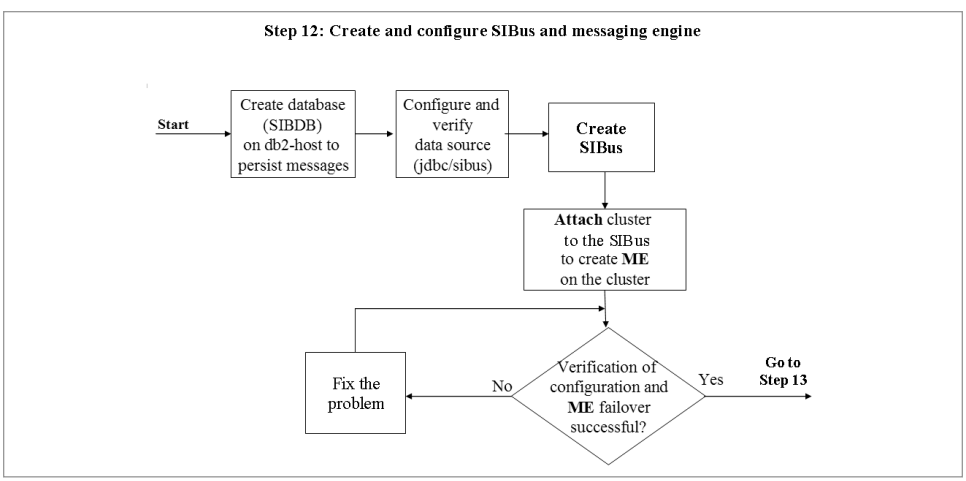

Figure 2-30: Step 12 – Create and configure SIBus and messaging engine

Step 13: Install, Configure, and Verify Edge Server – Load Balancer

The sample architecture you'll build in this book uses the Edge Server – Load Balancer's dispatcher component to spray IP requests across back-end HTTP servers (http-host1 and http-host2). You'll install edge-host1 as the primary (active) load balancer server and install edge-host2 as the secondary (standby) load balancer server.

Install, Configure, and Verify Edge Server – Load Balancer

Figure 2-31 shows the flow chart you'll follow to install, configure, and verify the primary and secondary load balancer servers to spray requests across the back-end HTTP servers. Chapter 13 describes this process.

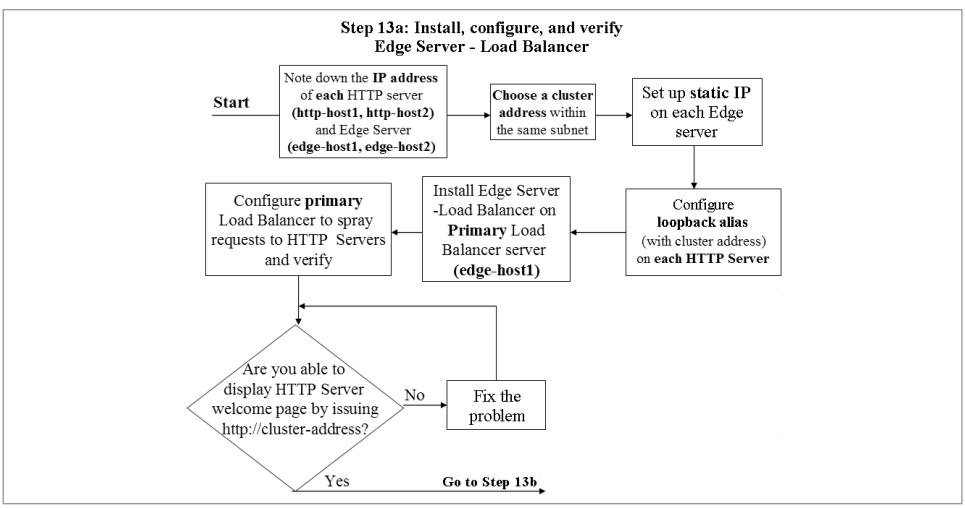

Figure 2-31: Step 13a – Install, configure, and verify Edge Server – Load Balancer

Figure 2-32 shows the output of the snoop servlet invoked through the load balancer using the cluster address or its alias (http://noyb-cluster.noyb.com/snoop). You can refresh the browser a few times and use the servlet output and Load Balancer Monitoring tool to see the distribution of load across HTTP servers and cluster members.

Figure 2-32: Output of snoop servlet invoked through the load balancer

In Figure 2-33, the Load Balancer Monitoring tool is indicating that the load balancer is spraying requests across the HTTP servers after receiving a request issued using the cluster address or its alias (http://noyb-cluster.noyb.com or http://noyb-cluster.noyb.com/snoop) to display the welcome page on each HTTP server.

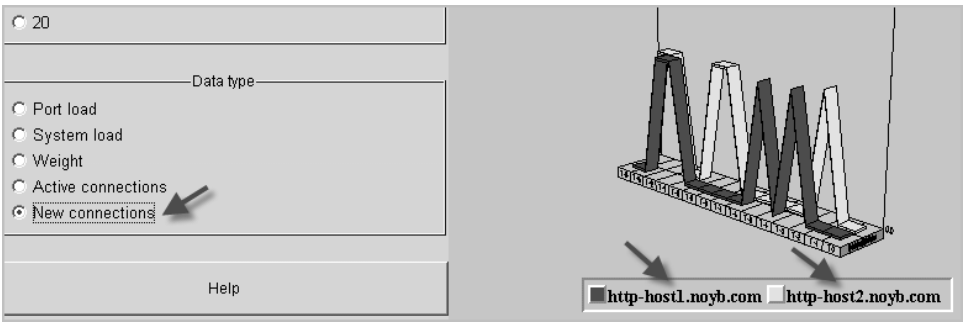

Figure 2-33: Load Balancer Monitoring tool indicating that the Load Balancer is spraying requests across HTTP servers

Configure Load Balancer Servers for High Availability

Figure 2-34 shows the flow chart you'll follow to configure the two Edge Server – Load Balancer servers for high availability. Chapter 14 explains how to perform and verify this configuration.

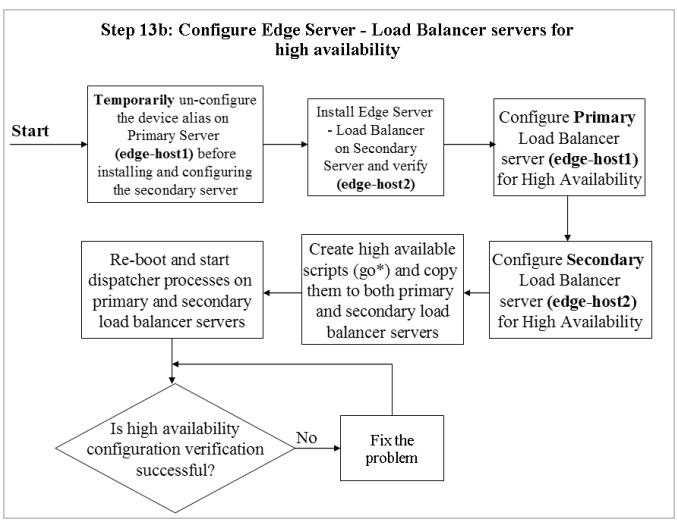

Figure 2-34: Step 13b – Configure Edge Server – Load Balancer servers for high availability

The status display in Figure 2-35 shows that the state of the secondary load balancer on edge-host2 has changed to Active (from Backup) and indicates that this load balancer is serving client requests after primary server failure.

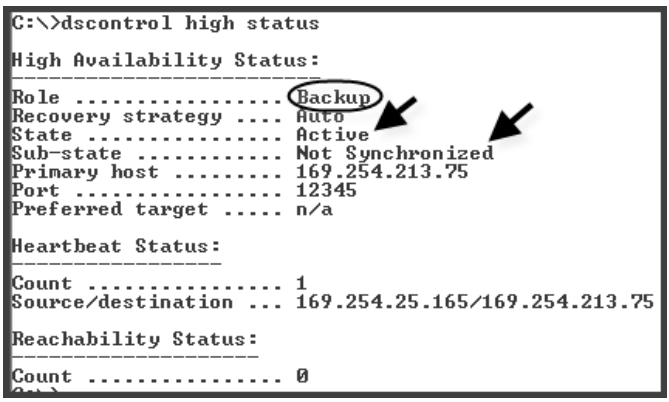

Figure 2-35: High availability status

Additional Tasks and Topics

Once you've installed, configured, and verified your WebSphere Application Server environment, some additional tasks and topics await you. The later chapters of this book address these issues and provide step-by-step instructions to guide your efforts.

Dynamic cache. WebSphere's dynamic caching functionality lets you cache dynamic content generated by J2EE application components (servlets and Java Server Pages) and push that content to the HTTP server/plug-in and/or a caching proxy. Use Chapter 15 to understand and configure the dynamic caching feature.

Security. To protect your WebSphere configuration from unauthenticated and unauthorized access, you should enable WebSphere security even if your applications don't use J2EE security. Chapter 16 provides step-by-step instructions for enabling security and setting up an admin group for WebSphere administrators who need to have access to the cell. There are three kinds of user registries you can use to keep a database of users and the groups to which they belong: custom, operating system, and Lightweight Directory Access Protocol (LDAP). Chapter 16 describes the use of each of these kinds of registries. Use this chapter to enable global security and create and configure admin roles.

Tivoli Directory Server. The preferred user registry is one based on the LDAP protocol. IBM Tivoli Directory Server (ITDS) is an LDAP server that

you can download from *http://www-306.ibm.com/software/tivoli/products/directory-server*. Chapter 17 provides step-by-step instructions on installing, configuring, and verifying ITDS for use as the user registry with WebSphere Application Server.

Secure Sockets Layer. Many sites need to use SSL to encrypt communications with browser clients on the Internet. Commerce sites that process financial transactions and sites that deal in personal information are examples of operations for which encryption is important. For added security, a site may use SSL for communications among the systems that make up a WebSphere environment. Chapter 18 explains how to configure SSL between

- a browser and an HTTP server
- a Web server plug-in and the Web container in WebSphere application servers
- WebSphere application servers and IBM Tivoli Directory Server

WebSphere product updates. A common administrative task is installing product updates for WebSphere. Chapter 19 gives a step-by-step description of this task for the nodes in a WebSphere cell.

Register WebSphere processes. Use Chapter 20 to register WebSphere processes in the Windows registry and run them from the Windows **Services** panel if you're using a Windows operating system.

Web services enablement. Use Chapter 21 to understand and configure the architecture necessary to invoke Web services through the SIBus.

Managing WebSphere Application Server. Consult Chapter 22 to understand WebSphere tracing, the collector tool, First Failure Data Capture, Log Analyzer, thread dumping, heap dump analysis, configuration archive backup and restore, and performance monitoring with Tivoli Performance Viewer.

Application Server Tool (AST) Kit. Use Chapter 23 to understand J2EE packaging.

Enterprise archives (EARs). Use Chapter 24 to understand EAR installation, enhanced EARs, fine-grained application updates, and mapping EAR modules to specific application servers.

WebSphere Rapid Deployment. Use Chapter 25 to understand and configure WebSphere Rapid Deployment.

J2EE system management. Use Chapter 26 to understand system management using the J2EE Management API.