

## Chapter 27

# WebSphere Proxy Server: On Demand Configuration and Cross-Cell Routing

WebSphere Proxy Server is a new type of server supported in WebSphere Application Server Network Deployment (ND) package (in version 6.0.2 and later). This Proxy server receives requests from clients initially on behalf of content servers and work load manages and routes the requests across content servers depending on the policies and filter classification definitions. The content servers in this scenario are On Demand Configuration (ODC) enabled WebSphere Application servers and optionally Generic servers (for example HTTP servers). Using the ODC capability, any deployment manager and application server can publish configuration changes for Proxy server(s) dynamically without doing any additional administrative tasks during runtime when changes occur. Proxy server(s) that will be created and configured on a managed node will subscribe for updates of configuration changes from ODC capable application servers. ODC uses the HA manager service infrastructure to publish and subscribe the updates.

WebSphere Proxy servers can secure the transport (using SSL), content and protect the identity of application servers using the response transformation feature (URL re-writing). The Proxy server can also cache responses to improve throughput and performance. Another good feature to note is SSL offload at the Proxy server. When using this feature you can terminate an SSL (HTTPS) connection at the proxy server after receiving the request from the client and use HTTP as transport protocol between proxy server and the content server(s) (which are application server(s)). You can administer and configure this Proxy server from the deployment manager's admin console (or wsadmin) in an ND environment.

In an earlier chapters we already saw two kinds of proxy servers: an Edge component Caching Proxy server (chapter-15) and the WebSphere plug-in that works with an HTTP server (chapter-6 and 9). After reading the first two paragraphs you might have realized already that this Proxy server is much more capable than the other two reverse proxy servers (the Edge caching server and the WebSphere plug-in) with its advanced configuration capabilities, dynamic routing policies and integrated system management in ND topology. It is interesting to note that the Proxy server can also **route requests across multiple cells** and supports session affinity and failover. You need to aware that you must keep the Proxy server on a node behind the firewall in your ND topology for security reasons because a node agent process runs on this managed node. If you still want to configure this Proxy server in the DMZ for any reason, then we suggest you use either

1. The traditional HTTP server/plug-in instead of WebSphere Proxy server in DMZ zone and let the HTTP server/plug-in spray requests across Proxy servers instead of application servers (refer to section **Configure Plug-in**)

as front-end to the Proxy server later in this chapter for more information) or

2. Isolate the Cell with the Proxy server(s) from the Cell with the application servers and configure the core group bridge service between the cells (refer to section **Cross-Cell Routing Using the Proxy Server** later in this chapter for more information).

**<begin note>**

In Part-VI of this book, you will work with a new type of server called ODR (On Demand Router) which is an extension to this Proxy server with request flow management to enforce SLA guarantees over and above what a Proxy server supports. ODR is supported in the WebSphere Application Server V6 - Extended Deployment Package.

**<end note>**

The diagram shown below depicts the architecture of a network deployment environment with federated nodes including a proxy server that you are going to build in this chapter. This is based on the system we built earlier (chapter-10) with minor modifications in which we used a Proxy server in place of an HTTP server/plugin module to route requests across the application servers (content servers) after receiving requests from clients. You will deploy a unique application on each node (Defaultapp on was-host1 and PlantsByWebSphere on was-host2) to test this environment, and you will see On Demand Configuration (ODC) behavior in action. At a later part of this chapter you will configure and verify the Proxy server to route requests to non-ODC compliant servers, and how to configure ESI caching, and also see how Proxy server could work-load manage requests across cluster members after creating the cluster.

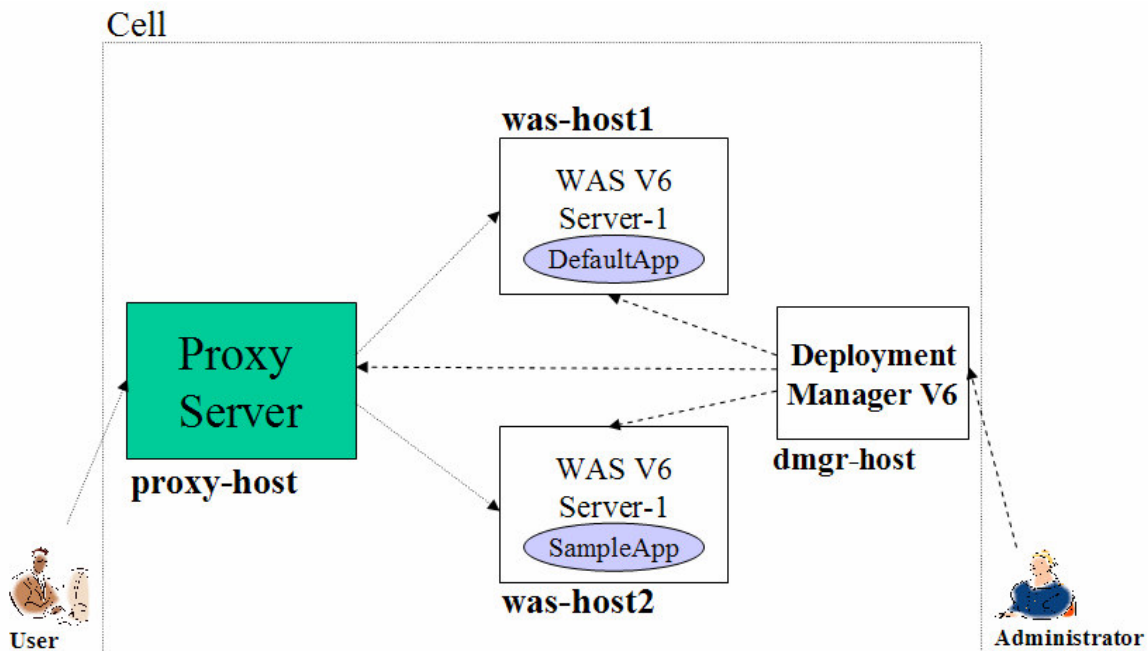


Figure 27-1: Proxy server routing requests across application servers on federated nodes

UNDER REVIEW

Author: Rama Turaga

Page 2

USE AT YOUR OWN RISK

Courtesy to: Owen Cline  
and Peter Van Sichel

## Configure Network Deployment Environment using Proxy server

Before creating the Network Deployment configuration using Proxy server specified earlier, you must complete some prerequisite tasks. After successful completion of these tasks, your system architecture should look similar to the one shown earlier (The Proxy server will be added after completing these prerequisite tasks).

### Prerequisite Tasks

To set up the environment we discussed earlier, complete the following tasks.

**Task 1: Create a deployment manager profile.** Use Chapter 7 to install, create, and verify a deployment manager profile on dmgr-host. We'll assume no application server profile exists on this machine at this time. This means the default manager profile you created now will be the default profile on that node. **Make sure that you apply the fix pack level to V6.0.2 or higher.**

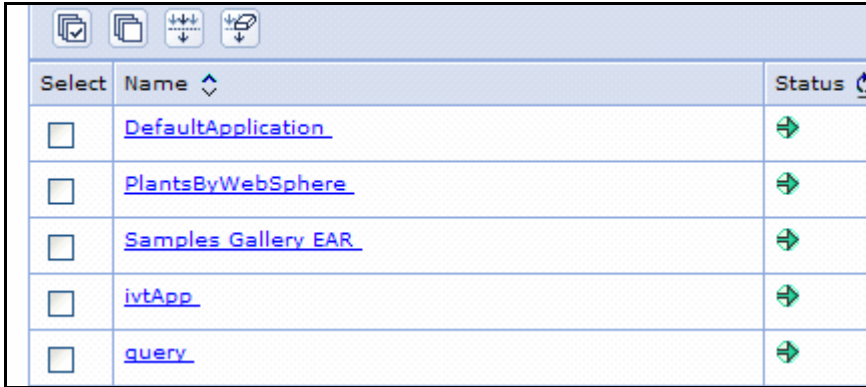
**Task 2: Create an application server profile.** Use Chapter 7 to install, create and verify an application server profile on was-host1. Do not install sample applications on this node during the installation. Verify that you see three applications as shown in the screenshot below, after successful installation. **Make sure that you apply the fix pack level to V6.0.2 or higher.**



Select	Name	Status
<input type="checkbox"/>	<a href="#">DefaultApplication</a>	Ready
<input type="checkbox"/>	<a href="#">ivtApp</a>	Ready
<input type="checkbox"/>	<a href="#">query</a>	Ready

Figure 27-2: Application server profile with Default Application

**Task 3: Create second application server profile.** Use Chapter 7 to install, create and verify an application server profile on was-host2. Make sure that you select the check box to install sample applications on this node during the installation (this deploys PlantsByWebSphere application on this node). Verify that you see the following applications as shown in the screenshot below after successful installation. **Make sure that you apply the fix pack level to V6.0.2 or higher.**

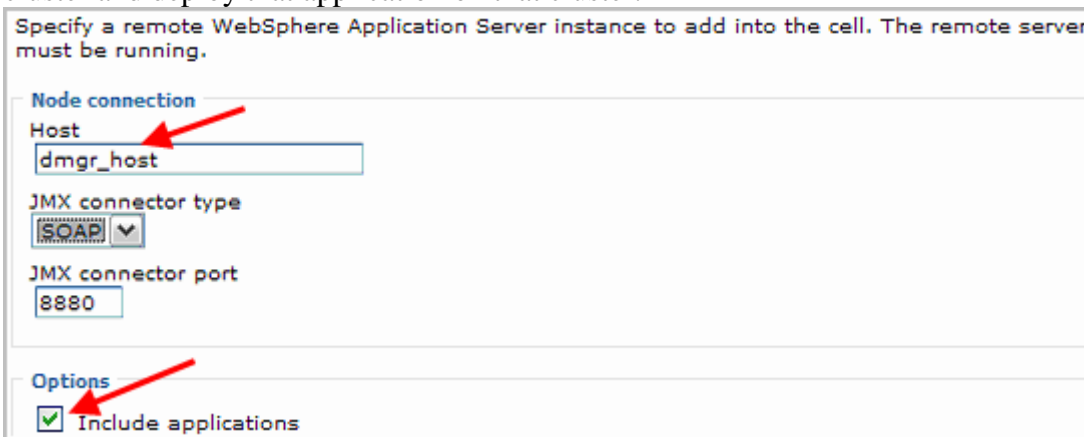


Select	Name	Status
<input type="checkbox"/>	<a href="#">DefaultApplication</a>	➔
<input type="checkbox"/>	<a href="#">PlantsByWebSphere</a>	➔
<input type="checkbox"/>	<a href="#">Samples Gallery EAR</a>	➔
<input type="checkbox"/>	<a href="#">ivtApp</a>	➔
<input type="checkbox"/>	<a href="#">query</a>	➔

Figure 27-3: Application server profile with PlantsByWebSphere Application

**Task 4: Create a custom profile.** Use Chapter 7 to install, create, and verify a custom profile on proxy-host. **Make sure that you apply the fix pack level to V6.0.2 or higher.** You will create and configure a Proxy server on this node later.

**Task 5: Federate nodes within application server profiles.** Follow Chapter 8 to federate nodes (was-host1 and was-host2) within the application server profiles to the Deployment Manager cell on dmgr-host and verify. While federating these nodes be sure to select the option to “include applications” as shown in the screenshot below. This means after federation you should see DefaultApplication, ivtApp and query applications mapped to server1 on was-host1 and PlantsByWebSphere and Samples Gallery applications mapped to server1 on was-host2 (Navigate to **ApplicationsEnterprise Applicationsapplication!Target mappings**). Remember that in a network deployment environment an application (DefaultApplication for example) with the same context root can not exist on two different servers unless they are cluster members. If you want to have the same application on multiple application servers then you need to create a cluster and deploy that application on that cluster.



Specify a remote WebSphere Application Server instance to add into the cell. The remote server must be running.

**Node connection**

Host:

JMX connector type:

JMX connector port:

**Options**

Include applications

Figure 27-4: Selecting to include applications during federation

1. Start the application servers on the federated nodes from the deployment manager’s admin console (Navigate to **ServersApplication servers**). Verify that you are able to invoke snoop servlet from a browser using the URL **http://was-host1:9080/snoop** and Plants By WebSphere application using **http://was-host2:9080/PlantsByWebSphere** after successful federation.

**Task 6: Federate node within the custom profile.** Follow Chapter 8 to federate the node with the custom profile on proxy-host to the Deployment Manager cell on dmgr-host and verify. (Use the addNode command from the proxy-host command prompt as shown in the screenshot.)



```
C:\WINDOWS\System32\cmd.exe
C:\IBM\WebSphere\AppServer\profiles\Proxy01\bin>addNode dmgr-host
```

Figure 27-5: Using addNode to federate the node that will hold the Proxy server

## Augment Profiles

Next, take the following steps to augment the profiles and create and configure the Proxy server. You need to augment the profile before creating a Proxy server. After augmenting the deployment manager profile you will see entries related to Proxy server on the admin console and server template sub directory (proxy\_augment) under profileTemplates directory. After augmenting the custom profile directory you should be able to create a Proxy server using the Proxy server template on that node. So only after augmenting the required profiles Proxy server capabilities will be enabled for the cell.

**Step 1.** Stop all the WebSphere processes (application servers, node agents and deployment manager) on all the nodes in that cell before augmenting the profiles.

**Step 2.** Open a command prompt on dmgr-host. Navigate to <WASV6-ROOT>\bin and issue the command “augmentProxyServer.bat Dmgr01” as shown in the figure below to augment the deployment manager’s profile.



```
C:\IBM\WebSphere\AppServer\bin>augmentProxyServer.bat Dmgr01
INSTCONFSUCCESS: Profile augmentation succeeded.
```

Figure 27-6: Augment Deployment manger profile

1. Navigate to <WASV6-ROOT>\logs\wasprofile directory, Open wasprofile\_augment\_Dmgr01.log file, scroll down to the end of the file and make sure that the log file indicates success as “INSTCONFSUCCESS”. You should consult this log file if you face any problems during the augmentation.

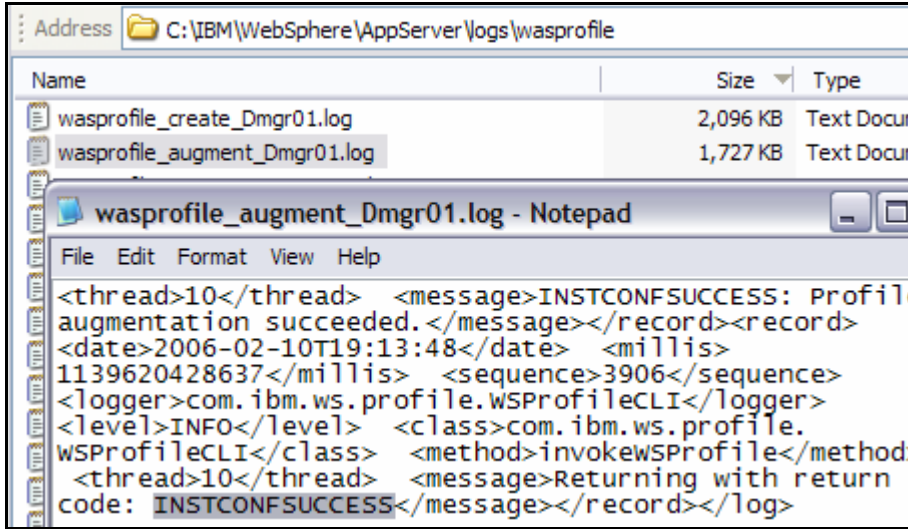


Figure 27-7: Checking the Dmgr profile augmentation log for success.

2. Navigate to `<WASV6-ROOT>\profileTemplates` directory and you should see a new directory, `proxy_augment`, added after augmenting the profile representing the Proxy server template.

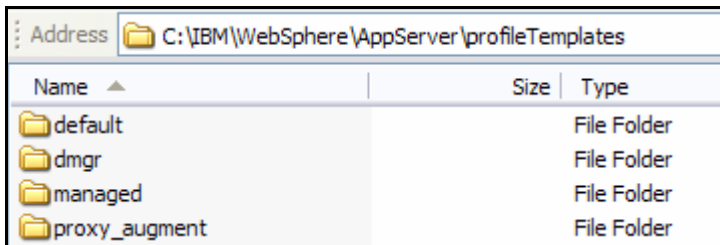


Figure 27-8: The proxy\_augment directory in profileTemplates

**Step 3.** Open a command prompt on proxy-host (where you created a custom profile). Navigate to `<WASV6-ROOT>\bin` and issue the command “`augmentProxyServer.bat Proxy01`” as shown in the figure below to augment the custom profile. You will create a Proxy server in this profile after augmenting it.

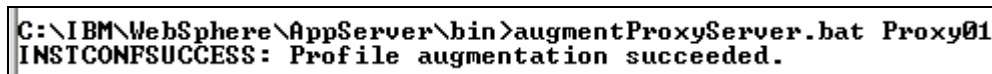


Figure 27-9: Augmenting custom profile to create a proxy server

1. Navigate to `<WASV6-ROOT>\logs\wasprofile` directory, Open `wasprofile_augment_Proxy01.log` file, scroll down to the end of the file and make sure that the log file indicates success as “`INSTCONFSUCCESS`”. You should review this log file if you face any problems during the augmentation.

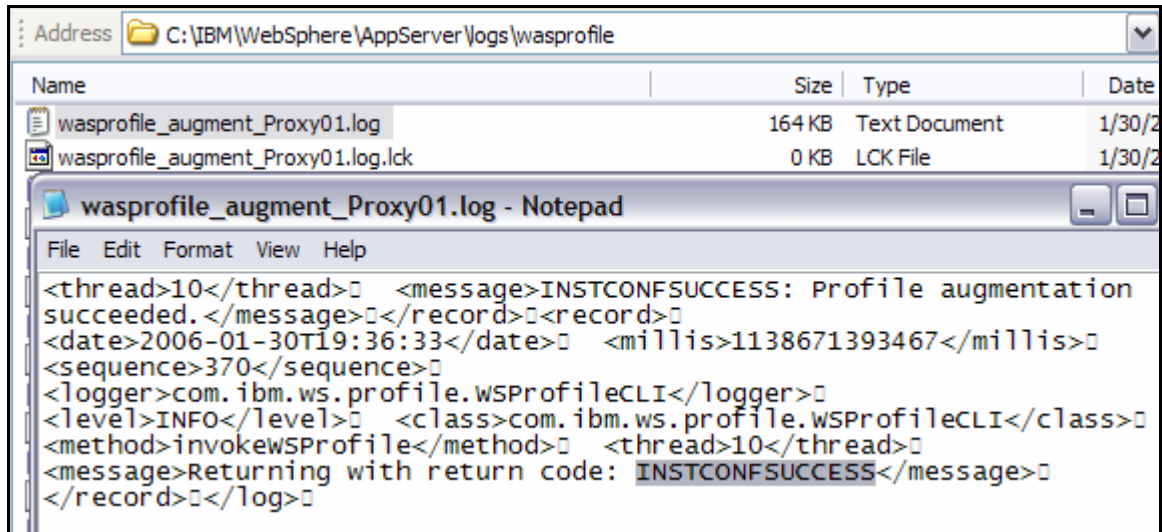


Figure 27-10: Checking the custom profile augmentation log for success.

**<begin note>**

You augment only the deployment manager and custom profile (in which you intend to create Proxy server(s)). There is no need to augment application server profiles on nodes was-host1 and was-host2.

**<end-note>**

**Step 4.** Start the deployment manager process and node agent processes on all the nodes (proxy-host, was-host1 and was-host2) after successfully augmenting the deployment manager and the custom profile.

**Step 5.** Connect to the Deployment Manager's admin console: **http://dmgr-host:9060/ibm/console** and you should find that the Proxy server option has been added to the admin console. This is the result of augmenting the deployment manager profile.

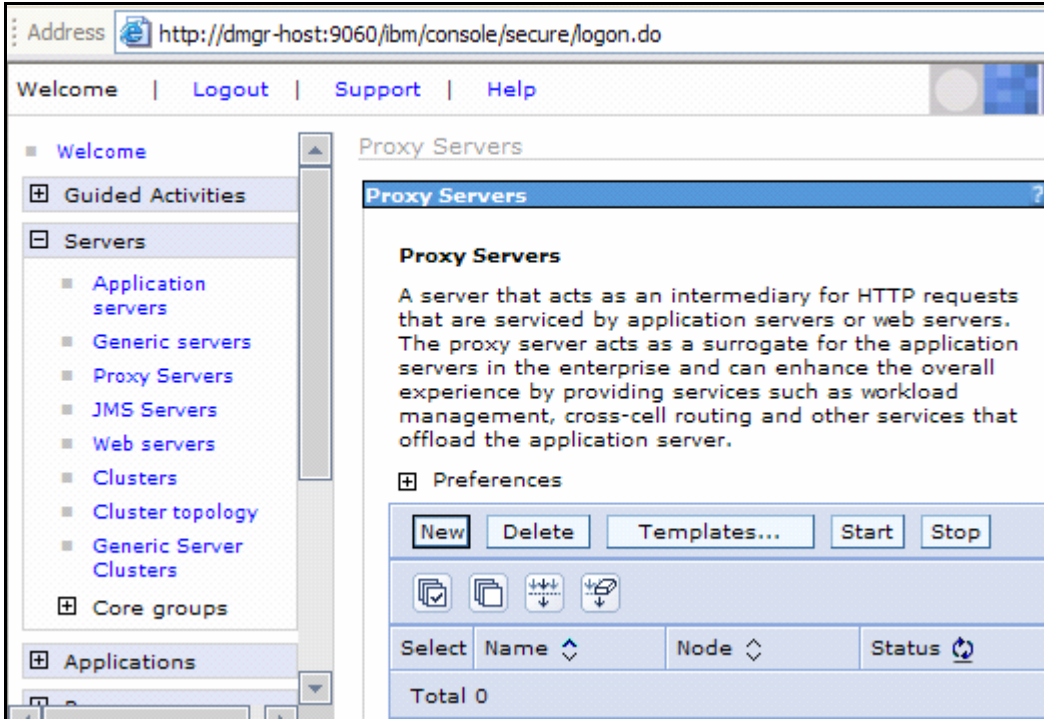


Figure 27-11: New Proxy server option and selection to create a new Proxy server

### Create and configure the Proxy server

**Step 6.** Now, you will create a Proxy server on the augmented custom profile (Proxy01). To do so, expand **Servers**, select **Proxy Servers**, and then click the **New** button.

**Step 7.** On the next panel (Figure 27-x), select the node within the custom profile (Proxy01) from the drop down list and specify a name for the Proxy server. We'll use the name pserver1 for this example. When you are finished with this panel, click **Next**.

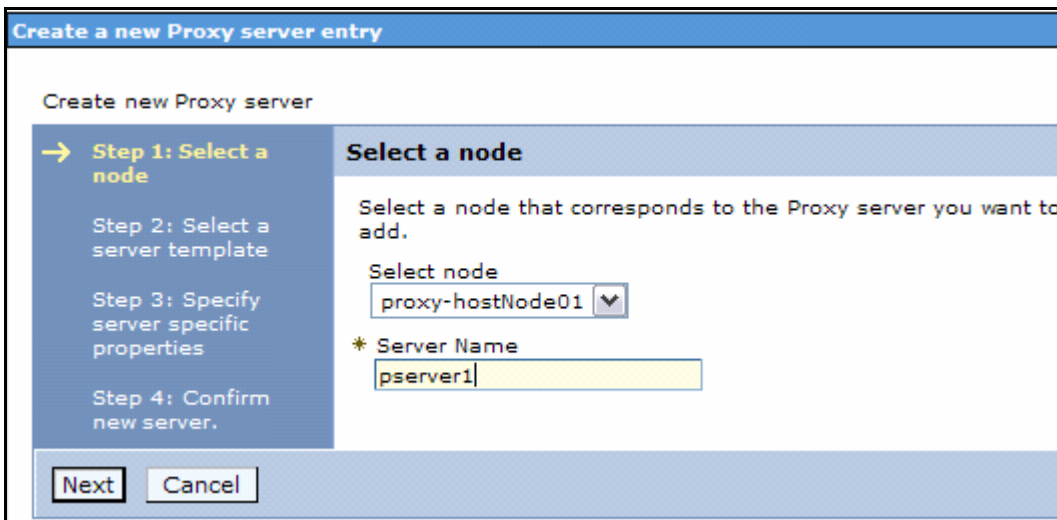


Figure 27-12: Specifying Proxy server information

**Step 8.** On the next panel, select the default proxy server template radio button as shown in the screenshot and click on **Next**.

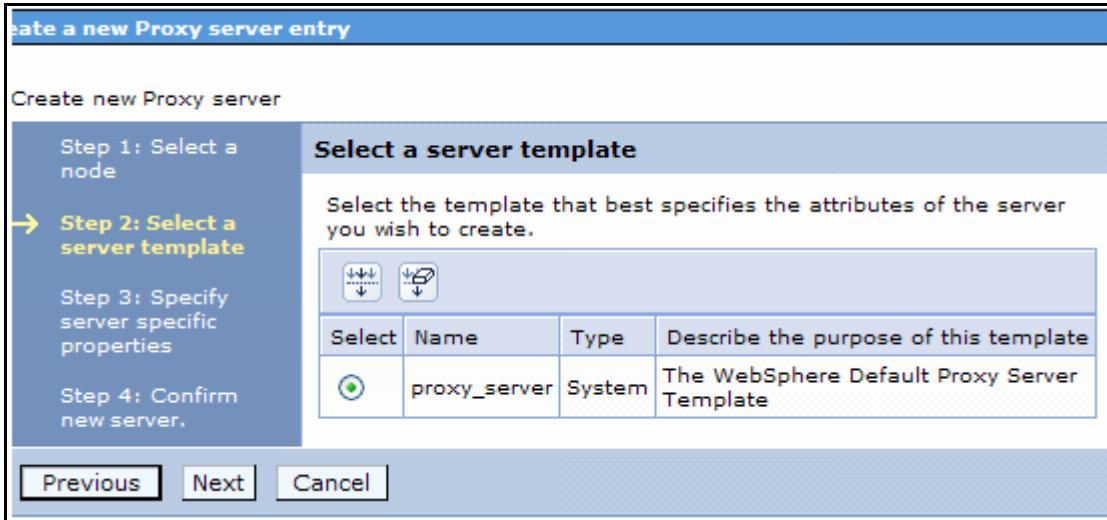


Figure 27-13: Selecting the Proxy server template

**Step 9.** On the next panel (Figure 27-x), Select the check box for “Generate Unique Http Ports” if this node has existing servers configured, Otherwise do not select the check box if the Proxy server is the only server on this node. Then click **Next**.

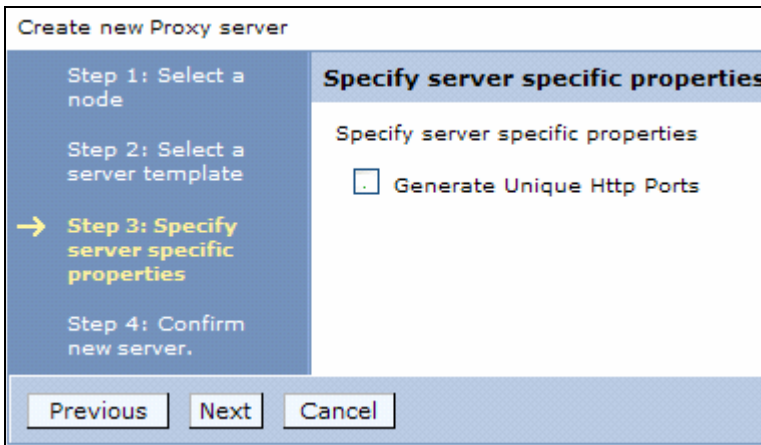


Figure 27-14: Specify to create Unique Http Ports

**Step 10.** Click **Finish** on the summary panel, save the configuration when prompted, and click **OK**.

### Verify Proxy server creation

To verify the proxy server creation, complete the following steps. After completing these steps you will realize how easy it is to maintain the WebSphere environment compared to having an HTTP server/plugin. In this architecture you do not have to generate or propagate plug-in files as it supports ODC just to name a few advantages.

**Step 11.** From the Deployment Manager's admin console, expand **Servers** and click **Proxy servers** to see the Proxy server (pserver1) you created. as shown in the screenshot below.

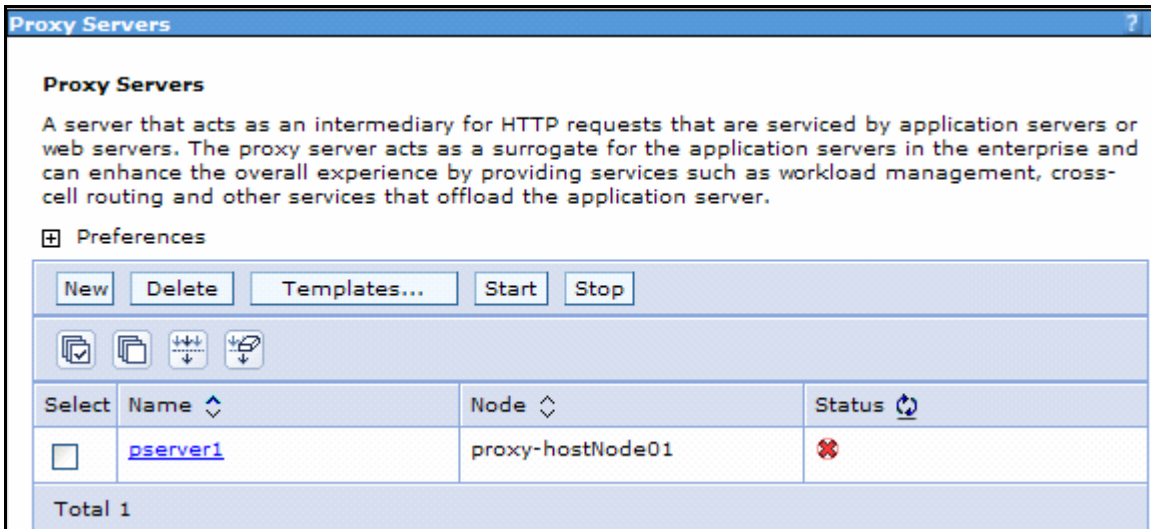


Figure 27-15: Successful creation of Proxy server

**Step 12.** Click on the **Proxy server** (pserver1) you created, and expand **Ports** to see the ports that have been assigned to this server as shown in the screenshot. Look at the port number that was assigned for **PROXY\_HTTP\_ADDRESS** (which is 80 in this case). You will access the contents on application servers through this port on the Proxy server.

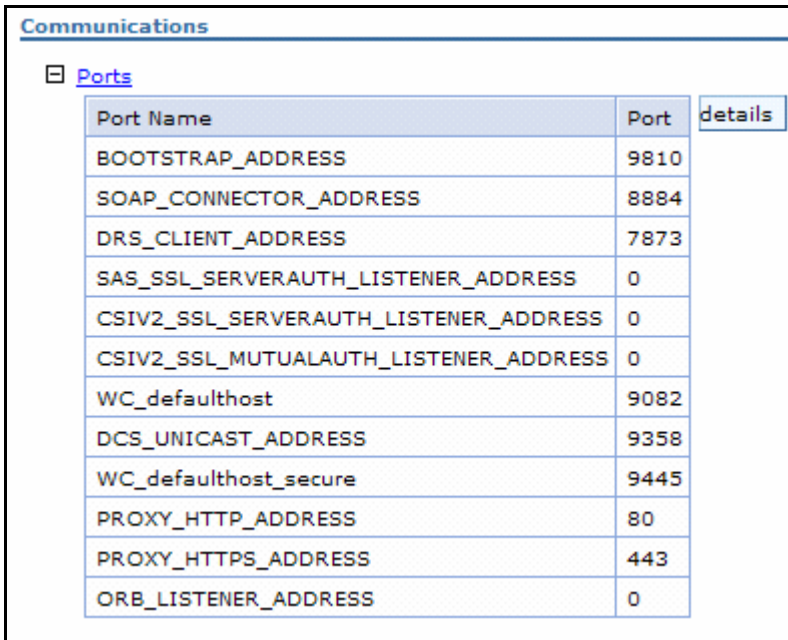


Figure 27-16: List of port numbers assigned

**Step 13.** Start the Proxy server. Select the check box for your Proxy server (pserver1), and click the **Start** button to start the Proxy server as shown below.

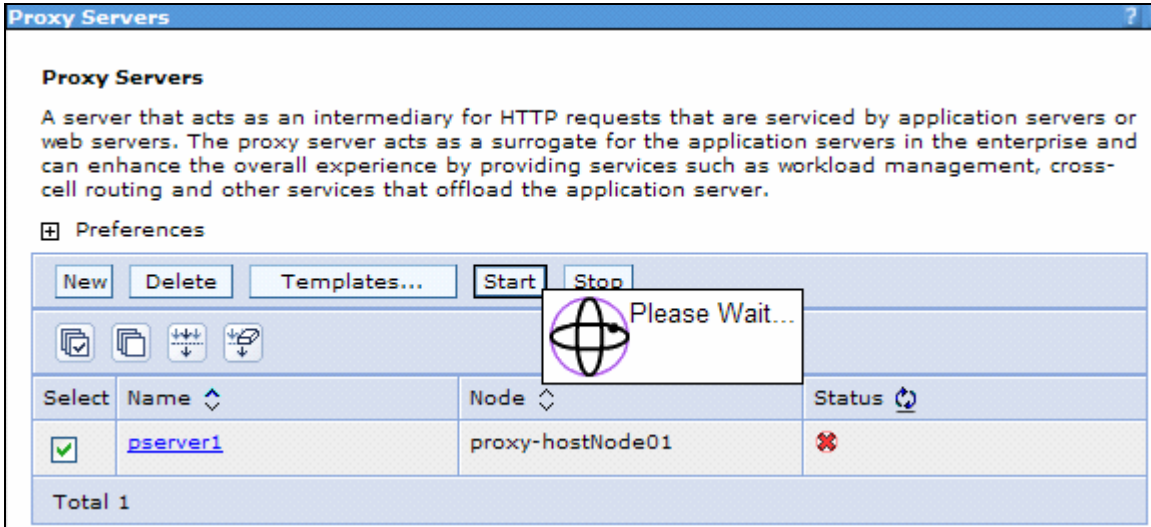


Figure 27-17: Starting the Proxy server

**Step 14.** Next, verify that the proxy server has been started.

1. Navigate to <PROFILE-ROOT>\logs\pserver1 directory on proxy-host, Open SystemOut.log file, scroll down to the end of the file and make sure that the log file indicates that the server was started successfully. You can also observe that the directory structure and the logging architecture looks like an application server. If you have any problems during the Proxy server startup or runtime, then you need to look at the log files under this directory.

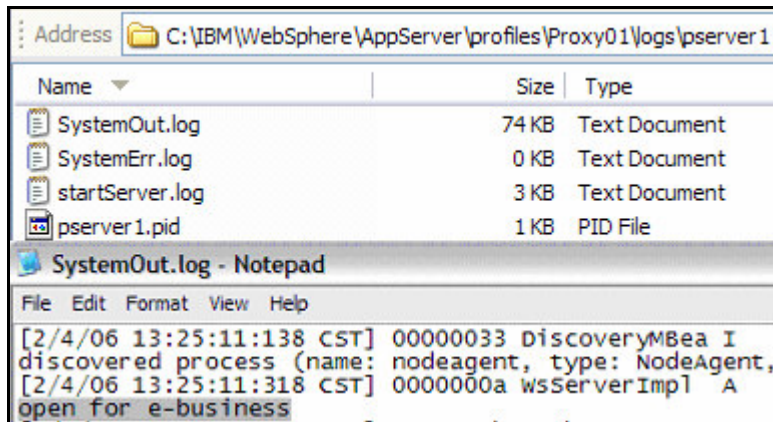


Figure 27-18: Verifying that the proxy server started successfully

2. Deployment manager's admin console should show pserver1 listed with a status of "running" (green arrow), as shown below.

Select	Name	Node	Status
<input type="checkbox"/>	<a href="#">pserver1</a>	proxy-hostNode01	

Figure 27-19: Proxy server started

**Step 15.** Open a browser, and issue the URL <http://proxy-host/snoop> to invoke the snoop servlet through the Proxy Server.



Figure 27-20: snoop servlet request served through the Proxy server

**Step 16.** Issue the URL <http://proxy-host/PlantsByWebSphere> to invoke the Plants By WebSphere through the Proxy Server.

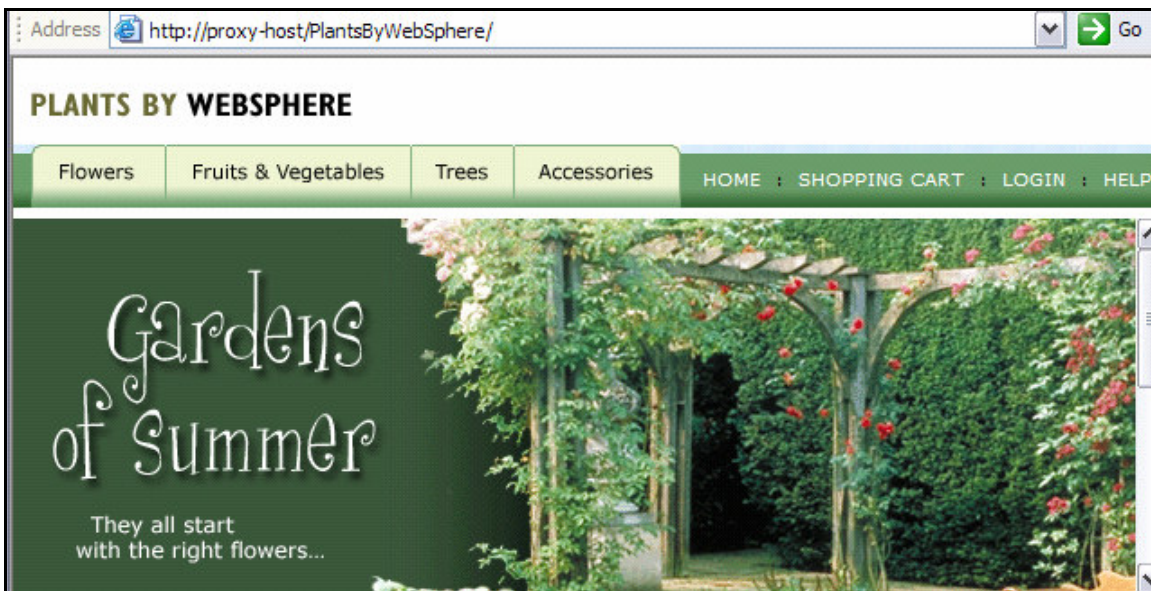


Figure 27-21: Plants By WebSphere application served through the Proxy server

**<begin note>**

We had to map default application's web module to the proxy server to serve the default application through the proxy server in our sample configuration. The reason could be that the default application's context root is "/". Navigate to **Application|Enterprise applications|Defaultapplication** and click on map modules to the application servers to map modules to the proxy server in addition to the application server(s). You need to do

this only for the default application, all other applications will be served through the proxy server without mapping to the proxy server.

**<end note>**

## Configure Proxy Server to Route Requests to Non ODC Compliant Servers

The Proxy Server also supports routing requests to the web servers or application servers that are not On Demand Configuration (ODC) compliant, as well as to servers that are not managed by the deployment manager. If you have performed Content Based Routing (CBR) tasks using the Load Balancer described in chapter-13, then you quickly realize that this function looks similar. In this section you will configure the Proxy server to route requests to an HTTP server (non-ODC compliant) according to the rules specified in the configuration as shown in the diagram below. Basically, after receiving the request from a client, the Proxy server looks at the content of the URL and forwards the request to the HR application on the Web server (running on web-host) if the URI matches “/hr/\*”.

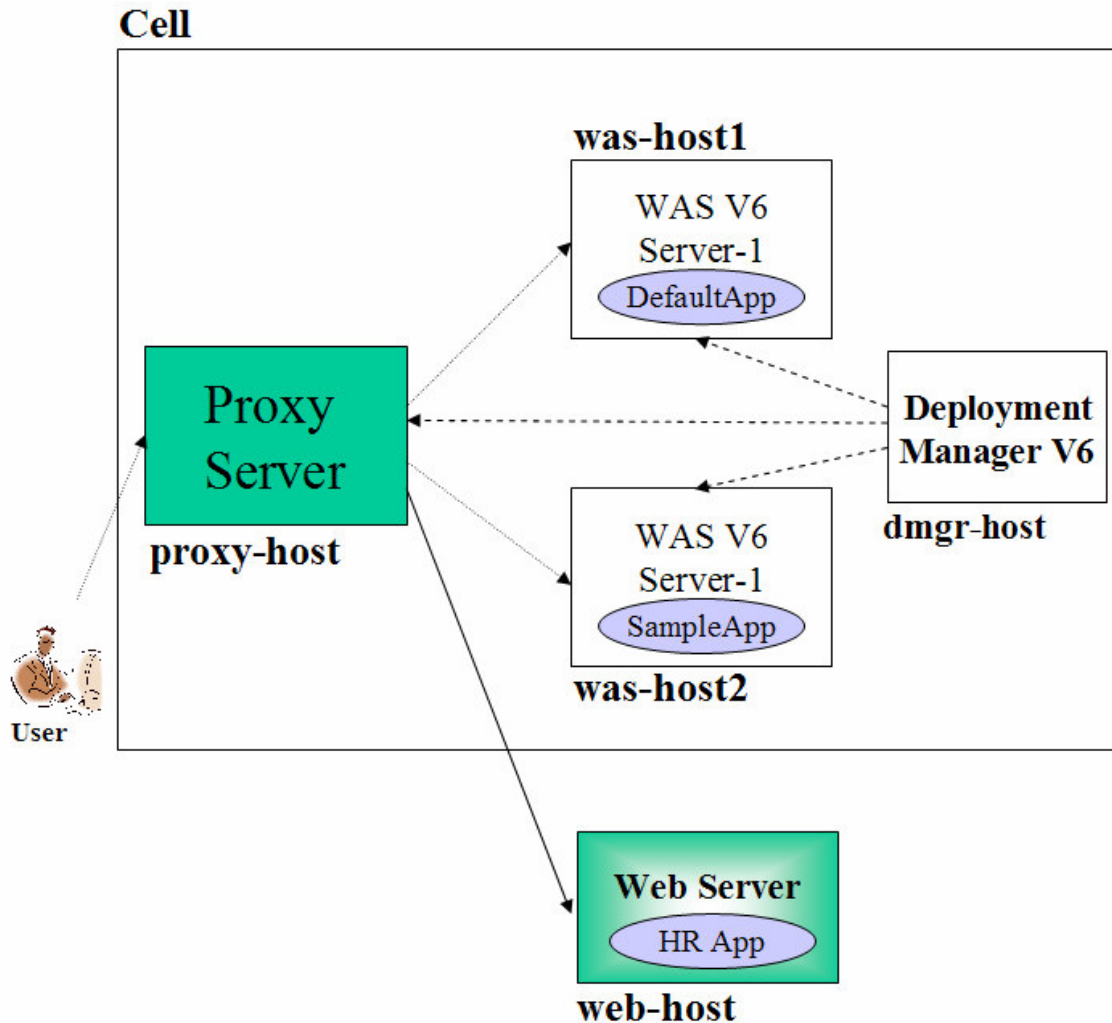


Figure 27-22: Proxy server routing requests to non ODC compliant Web server

### Create and configure Generic Server Cluster

**Step 1.** Navigate to **Servers|Generic Server clusters** and click on **New** to create cluster of generic servers. On the resulting panel enter the name for this cluster (WebCluster in our example) and the protocol (HTTP or HTTPS) according to your environment. We used HTTP in our sample configuration. Click on the **Apply** button after entering this information. You can see that the options under **Additional Properties** will be enabled after clicking on **Apply**.

**Generic Server Clusters > New**

A cluster of generic servers (servers that are not managed by WebSphere such as web servers) that host a common set of resources and that can be routed to as a unit.

Configuration

**General Properties**

\* Name  
WebCluster

\* Protocol  
HTTP

The additional properties will not be available until the general properties for this item are saved.

**Additional Properties**

- Custom Properties
- Ports

Apply OK Reset Cancel

Figure 27-23:

**Step 2.** Click on **Ports** to add the hostname and port number of servers that you want to include under this cluster (WebCluster). In our sample configuration we have only one Web server on **web-host** listening for requests at port **80**. On the resulting panel click on the **New** button to enter Host and Port information as shown below. Click on **OK** and save the configuration.

**Generic Server Clusters > WebCluster > Ports > New**

Port Number of the server

Configuration

**General Properties**

\* Host  
web-host

\* Port  
80

\* Weight  
2

The additional properties will not be available until the general properties for this item are saved.

**Additional Properties**

- Custom Properties

Apply OK Reset Cancel

Figure 27-24: Adding a new member to the generic web cluster

### Create an URI Group

**Step 3.** Navigate to **Environment|URI Groups** and click on **New** to create an URI pattern pointing to the web resources running Web server (located on web-host). Enter **HR App** for the name and **/hr/\*** for the URI pattern as shown below. Click on **OK** and save the configuration.

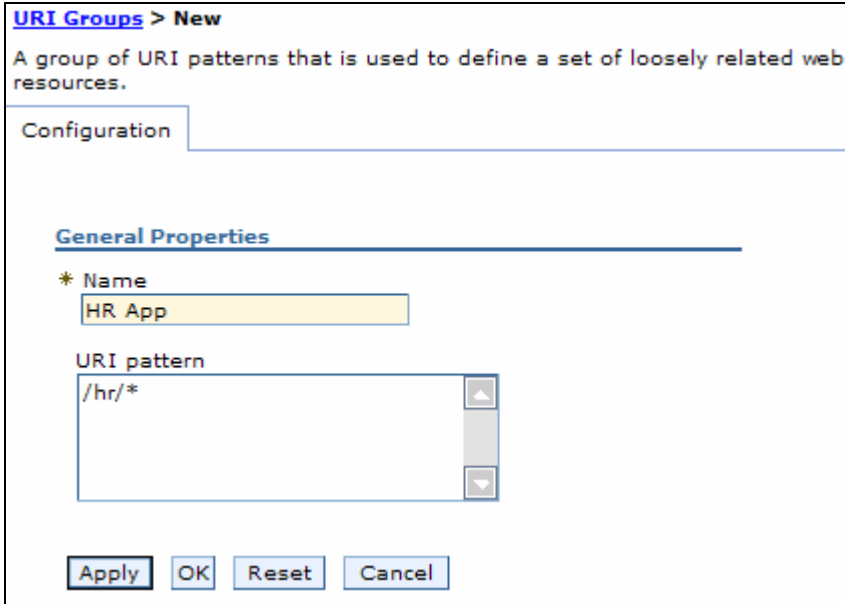


Figure 27-25: Specifying the URI pattern for the generic web cluster member

**Step 4.** You need to make sure that the Web server on web-host receives and processes the requests with URIs that match /hr/\*. To support this test, create a test HTML page (home.html) under the Web server's document root (this directory depends on your locale and other parameters in the configuration file) as shown below. The default document root directory for IHSV6 is <IHSV6-ROOT>\htdocs\en\_US if your locale is en\_US. Review chapter-13 under the Content Based Routing section for more information on this. Start the Web server and verify that you can access this page from the browser directly before accessing it through the Proxy server. Open a browser and issue the URL **http://web-host/hr/home.html** assuming that the Web sever is listening at default port of 80.

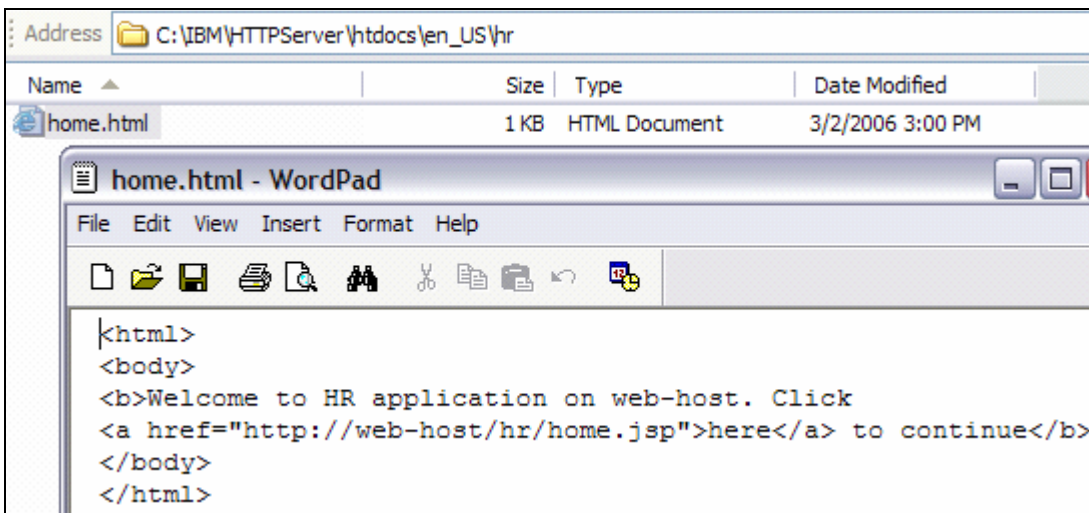


Figure 27-26: Simple test document

## Configure Proxy server to route requests to Web Server

**Step 5.** Navigate to **Servers|Proxy servers** and click on the **Proxy server (pserver1)**, expand options under **HTTP Proxy Server Settings** and click on **Routing rules** as shown below.

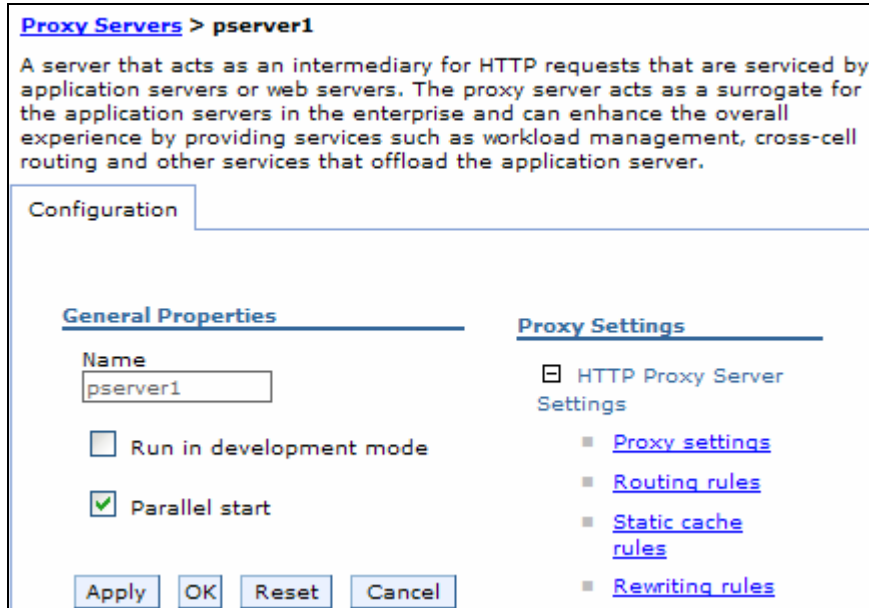
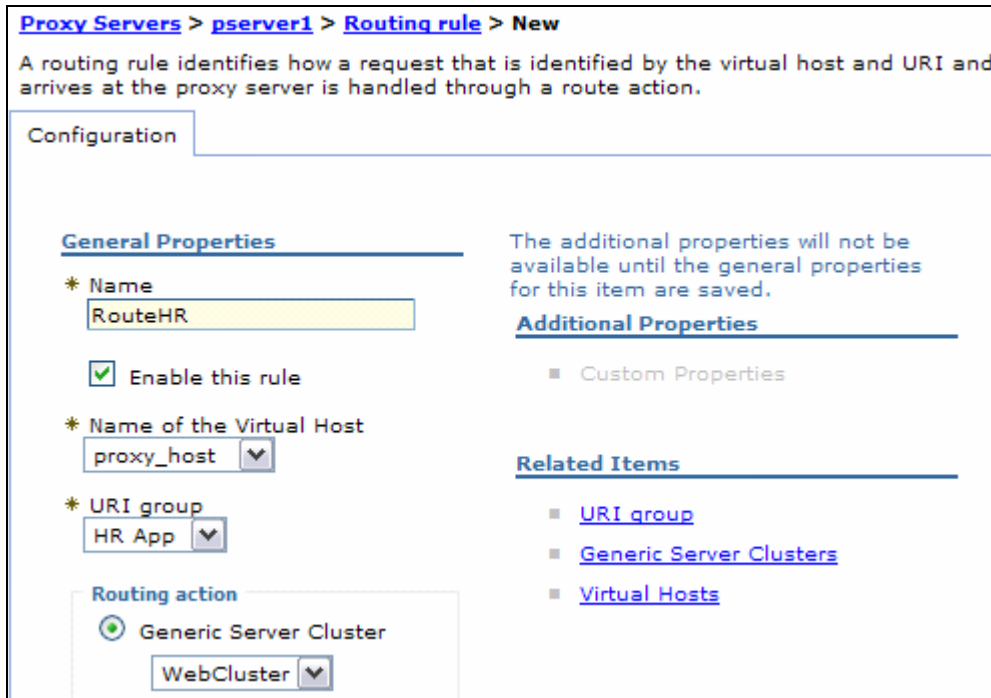


Figure 27-27: From the main Proxy server screen, you can navigate to the Routing rules.

**Step 6.** Click on **New** to create a new Routing rule. On the resulting panel enter **RouteHR** for the name, select **proxy\_host** as virtual host, **HR App** as URI group and **WebCluster** as Generic Server Cluster. Keep all other values to defaults. Click on **OK** and save the configuration.



**Proxy Servers > pserver1 > Routing rule > New**

A routing rule identifies how a request that is identified by the virtual host and URI and arrives at the proxy server is handled through a route action.

**Configuration**

**General Properties**

\* Name  
RouteHR

Enable this rule

\* Name of the Virtual Host  
proxy\_host

\* URI group  
HR App

**Routing action**

Generic Server Cluster  
WebCluster

The additional properties will not be available until the general properties for this item are saved.

**Additional Properties**

■ Custom Properties

**Related Items**

- [URI group](#)
- [Generic Server Clusters](#)
- [Virtual Hosts](#)

Figure 27-28: Configuring a new routing rule

## Verify Proxy server routing to Non-ODC compliant Web server

**Step 7.** Make sure the Web server (on web-host) and Proxy server are started. Open a browser and invoke the HR application on the web-host through the Proxy server using the URL **http://proxy-host/hr/home.html**. You should receive a response from the Web server as shown below.

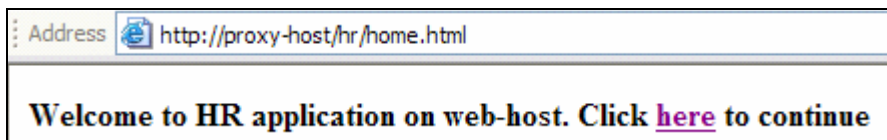
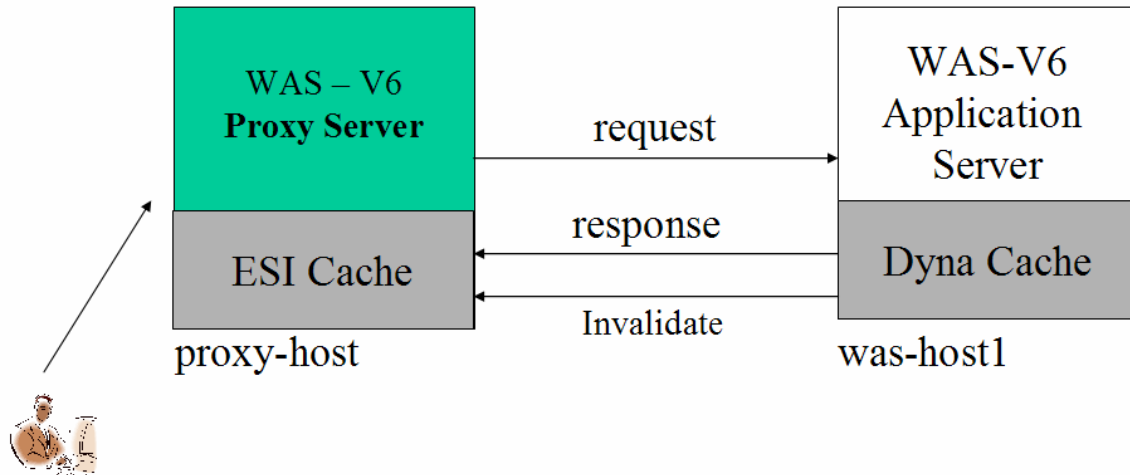


Figure 27-29: Initial response from the HR web application

## Configure Proxy server for caching

You can configure the Proxy server to cache the static and dynamic content that has been served by application servers at the backend. The configuration is similar to the one you did in chapter-15, but here you configure ESI caching on the Proxy server instead of configuring caching on the HTTP plug-in. The diagram shown below explains the simplified caching architecture using the Proxy server (instead of the HTTP server/plug-in).



<http://proxy-host/snoop>

Figure 27-30: Proxy Server Caching Architecture

### Configure Application Server Caching

**Step 1.** Configure the application server (server1) on was-host1 to cache snoop servlet. Use **Task 1** (Configure the Application Server and Enterprise Application to Use Dynamic Cache) in **chapter-15** to configure dynamic caching and verify. With this configuration, static content will also be cached by the Proxy server that is served by both of the application servers on was-host1 and was-host2.

**Step 2.** Use **Configure the application server** section under **Task 2** (Configure the Plug-in File and Application Server to Push Cached Content to the ESI Processor) in **chapter-15** to further configure application servers to push cached content externally. Bear in mind that you do not perform steps that are related to the plug-in because you are using the Proxy Server instead of the plug-in in this configuration. You will configure the Proxy server to cache this content in the next step. Later, you will verify this configuration..

### Configure Proxy Server Caching

**Step 3.** Navigate to **Servers|Proxy servers** and click on the **Proxy server** (pserver1), expand options under **HTTP Proxy Server Settings** and click on **Proxy settings** as shown.

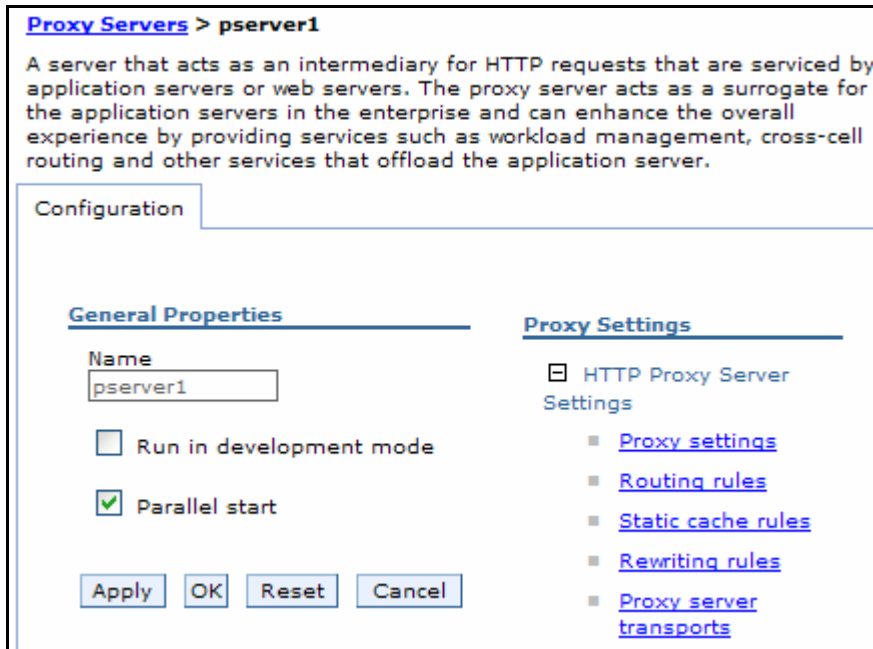


Figure 27-31: From the main Proxy server screen, you can navigate to Proxy settings.

**Step 4.** Scroll down in the resulting panel and you should see a section for **Caching**. By default caching is enabled on the proxy server. If your intention is to cache only the static content served by application servers at the backend then you do not have to do anything else. But if you also want to cache the dynamic content served by application servers, select the check box for **Cache Dynamic Content** as shown below.

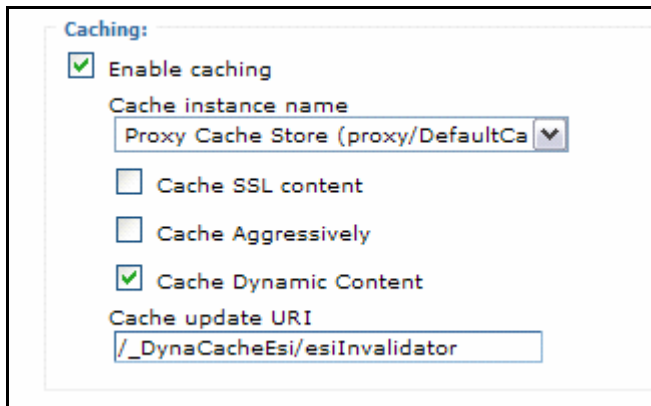


Figure 27-32: Enabling dynamic content caching

<begin note>

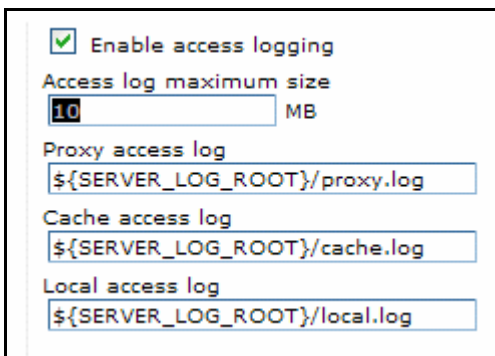
Remember that you need to disable dynamic caching (**unselect** Cache Dynamic Content) before you verify the Work Load Management (WLM) of Proxy server later in this chapter.

<end note>

**Step 5.** Scroll down in the same panel and you should see a section for **Logging**. You are going to verify the Proxy server ESI caching using these log files. (The Cache

Monitor application used in chapter-15 does not support viewing the cached content at the Proxy server at the time of writing this chapter.) This type of logging is disabled by default. Enable logging temporarily to test the caching by selecting checkbox for **Enable access logging**. Set the log file size to a comfortable size depending on your environment (Log file will be overwritten with the new contents after reaching the maximum size). There are three log files as shown below, each logging a different kind of information as explained below:

1. **proxy.log**: a log for responses (URIs) that are received from the application servers
2. **cache.log**: a log for content (USRIs) that is served from the Proxy server cache.
3. **local.log**: a log for locally served proxy content (URIs) that is **not** served from the Proxy server cache, for example redirects and internal errors.



The screenshot shows a configuration window for proxy logging. At the top, there is a checked checkbox labeled "Enable access logging". Below it, the "Access log maximum size" is set to "10" MB. There are three text input fields for log file paths: "Proxy access log" is set to "\${SERVER\_LOG\_ROOT}/proxy.log", "Cache access log" is set to "\${SERVER\_LOG\_ROOT}/cache.log", and "Local access log" is set to "\${SERVER\_LOG\_ROOT}/local.log".

Figure 27-33: Proxy log file configuration

## Verify Proxy Server Caching

**Step 6.** Verify Caching configuration at Proxy server by invoking the snoop servlet through Proxy server (**http://proxy-host/snoop**). The first request will always be a cache miss and will be served from the application server (serve1 on was-host1) and stored in the Proxy server's cache. You can see this information in proxy.log after the first request. Resubmit (or click on Refresh button) the snoop request and the subsequent request will be served from the Proxy server's cache. Next invoke the PlantsByWebSphere application through the Proxy server (http://proxy-host/PlantsByWebSphere) and re-submit it few times to see that the static content will be stored in Proxy server's cache. You can see cached content information is logged in cache.log file as shown below.

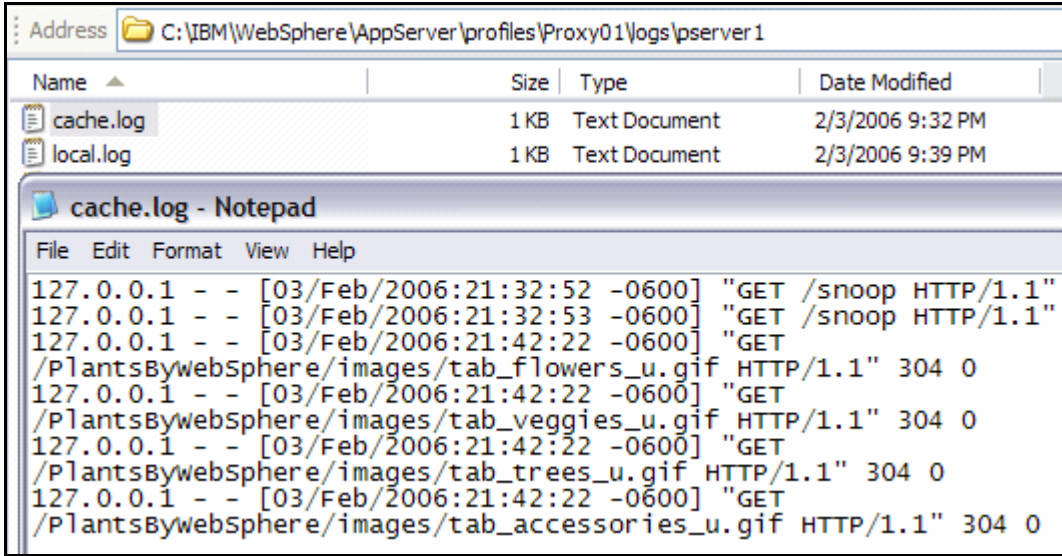


Figure 27-34: Example of Proxy server cache.log content

## Proxy server Workload Management

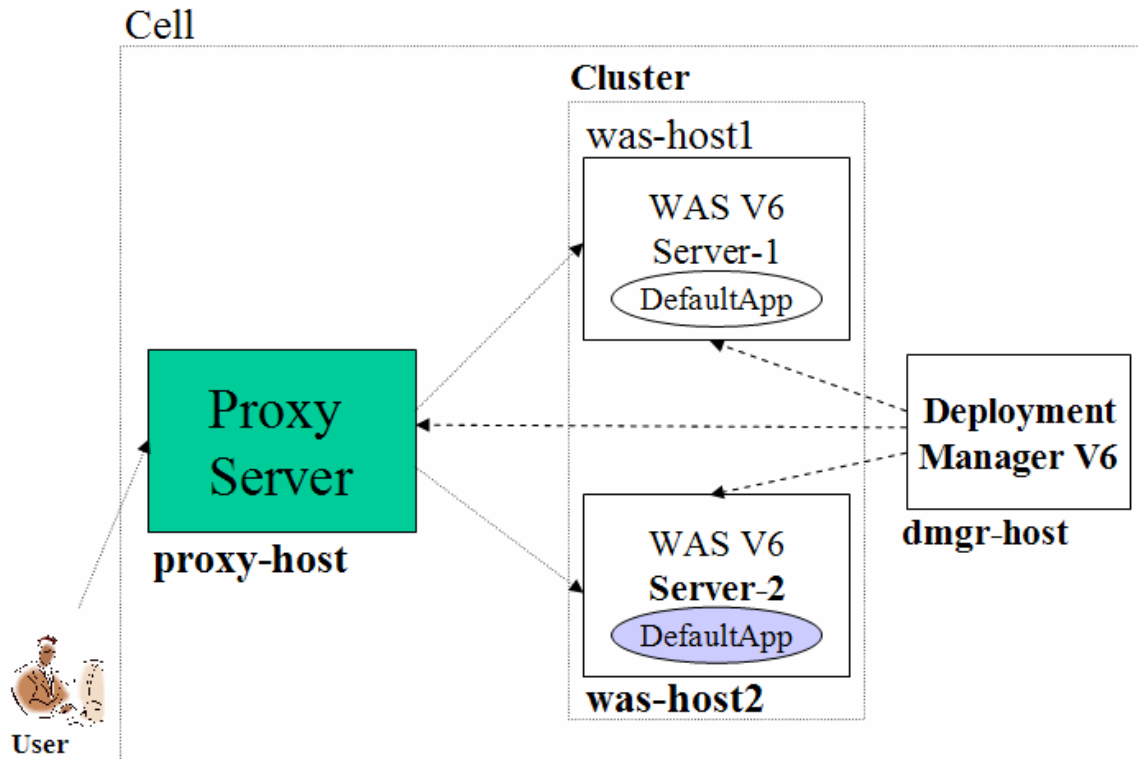


Figure 27-35: Proxy server workload managing requests across cluster members

done by the HTTP server plug-in module. Use chapter-10 to create a vertical or horizontal cluster using server1 on was-host1 as a server template and as a first cluster member. Create the second cluster member on was-host1 (Vertical Cluster) or was-host2 (Horizontal Cluster). The Proxy server automatically recognizes the cluster and workload manages the requests across the cluster members without any further configuration. (No plug-in configuration file generation or propagation is needed in this case.) Make sure that you disable dynamic caching on the Proxy before verifying the Proxy server WLM (Refer to the Configure Proxy server for Caching section earlier in this chapter.) Use the URL <http://proxy-host/snoop> to test the Proxy server WLM. Refer to chapter-10 for verification steps.

The Proxy server also supports session affinity and session failover. To verify these features configure session persistence using the instructions given in chapter-11.

## Proxy server High Availability

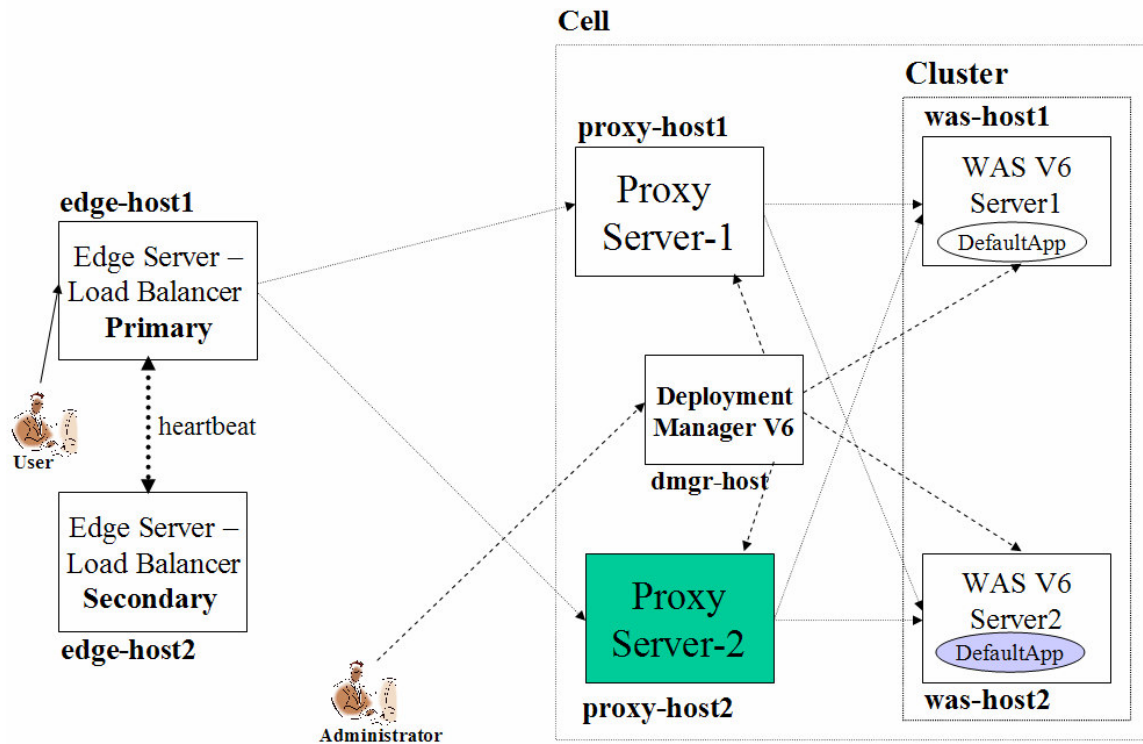


Figure 27-36: Proxy server high availability architecture

Even though WebSphere V6 does not support clustering (for high availability) of Proxy servers at the time of writing this chapter (similar to the application server clustering), you can create a second Proxy server using the first one as a template and let a Load Balancer (or HTTP server/plug-in) spray requests across these similarly configured Proxy servers for high availability at the Proxy server layer. The architecture diagram above depicts how a WebSphere Network Deployment cell supports high availability at the Proxy server layer using two Proxy servers. The best practice to configure highly

available Proxy servers is to create, configure and verify the first Proxy server as needed and configure the subsequent Proxy server(s) using the first one as a template. This way all of the Proxy servers will have the same configuration and functionality (provided they are configured on similar systems).

### *Create and configure a second Proxy server for High Availability*

**Step 1.** Create a custom profile on proxy-host2; federate the node within this custom profile to the deployment manager cell on dmgr-host during its creation. Augment the custom profile using the instructions earlier and create a Proxy server in this custom profile. In our sample configuration we decided to create the second Proxy server (pserver2) in the existing custom profile in which we created the first Proxy server (pserver1) earlier. But in production environments it is recommended you create the second Proxy server on a separate node.

**Step 2.** From the Deployment Manager's admin console, Navigate to **Servers|Proxy servers**, click on **Templates..**, on the resulting panel click on the **New** button to create a template as shown below.

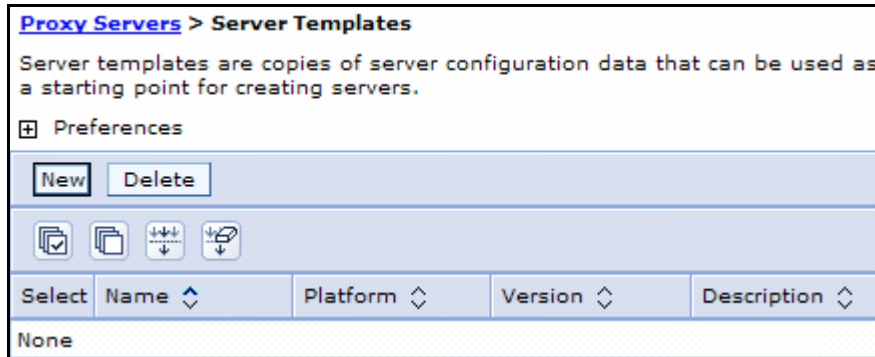


Figure 27-37: Creating an new Proxy server template

**Step 3.** On the resulting panel choose the Proxy server (pserver1 in our example) radio button to create a template out of the existing Proxy server you created earlier.

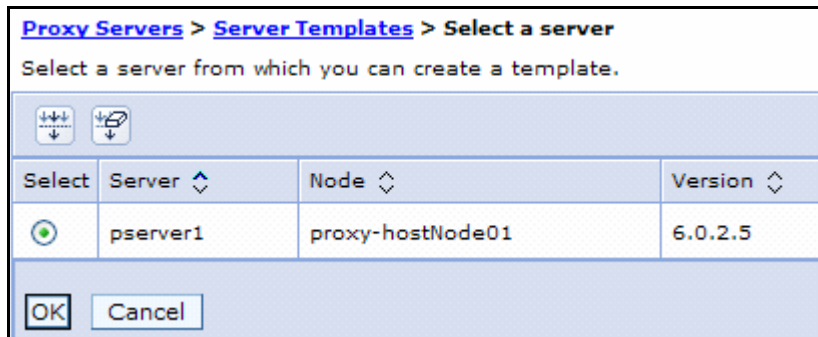


Figure 27-38: Using pserver1 as the Proxy server template

**Step 4.** Specify the name for the new template (proxytemplate in this example) and optionally a description for the template. Click on **OK** and save the configuration.

**Proxy Servers > Server Templates > Select a server > Create new server template**

Specify template name and description for the template to be created. New template will be created based on the configuration of the selected server.

Server Name  
proxy-hostNode01/p

\*Name  
proxytemplate

Description  
for proxy cloning

OK Cancel

Figure 27-39: Naming and describing the Proxy server template

**Step 5.** Navigate to **Servers|Proxy servers** and click on the **New** button to create the second Proxy server using the template we created in the earlier steps.

**Step 6.** In the resulting panel select the node from the drop down list in which you want to create this Proxy server. In this sample configuration, we selected proxy-hostNode01 (where pserver1 is also created) but in a production environment you would create in proxy-hostNode02 for example. Specify the name for the Proxy server (we specified it as pserver2). Click on **Next**.

**Create new Proxy server**

→ **Step 1: Select a node**

Step 2: Select a server template

Step 3: Specify server specific properties

Step 4: Confirm new server.

**Select a node**

Select a node that corresponds to the Proxy server you want to add.

Select node  
proxy-hostNode01

\* Server Name  
pserver2

Next Cancel

Figure 27-40: Naming a second Proxy server

**Step 7.** In the resulting panel choose the template you created earlier (proxytemplate) to create your new Proxy server (pserver2). Click on **Next**.

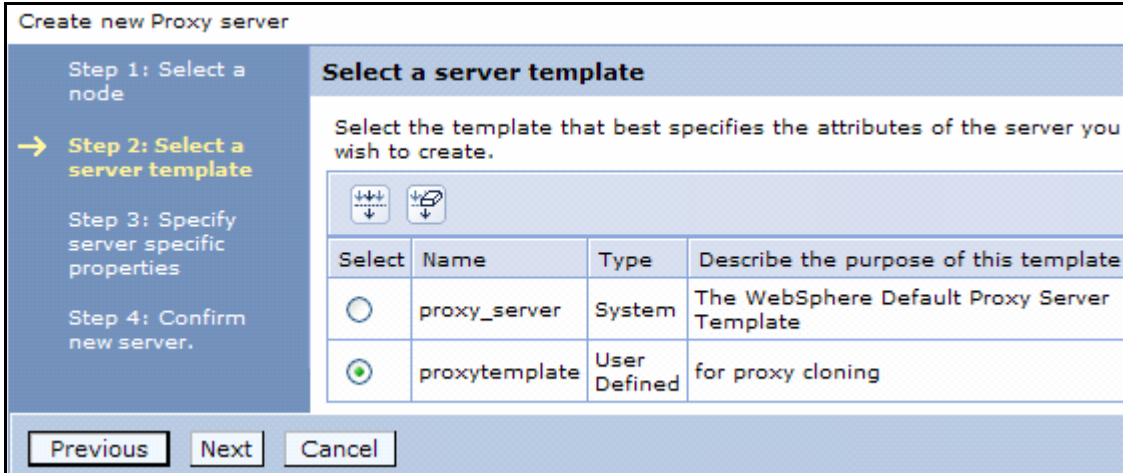


Figure 27-41: Choosing the proxy template to create a new Proxy server

**Step 8.** On the next panel, select the check box for “Generate Unique Http Ports” if this node has existing servers configured on it. (Since we already have pserver1 on this node, we selected this check box.) You do not need to select this check box if the Proxy server is the only server on this node. (Here we are equating a “node” with a physical machine. Obviously, port conflicts can occur if you have more than one Proxy server configured on the same machine.) Then click **Next**.

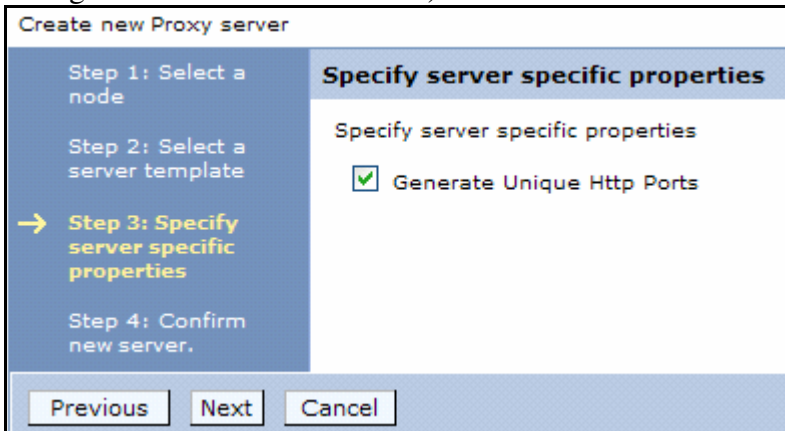


Figure 27-42:

**Step 9.** On the resulting summary panel, click on **Finish** and save the configuration. You should now see the second Proxy server in the Proxy server panel.

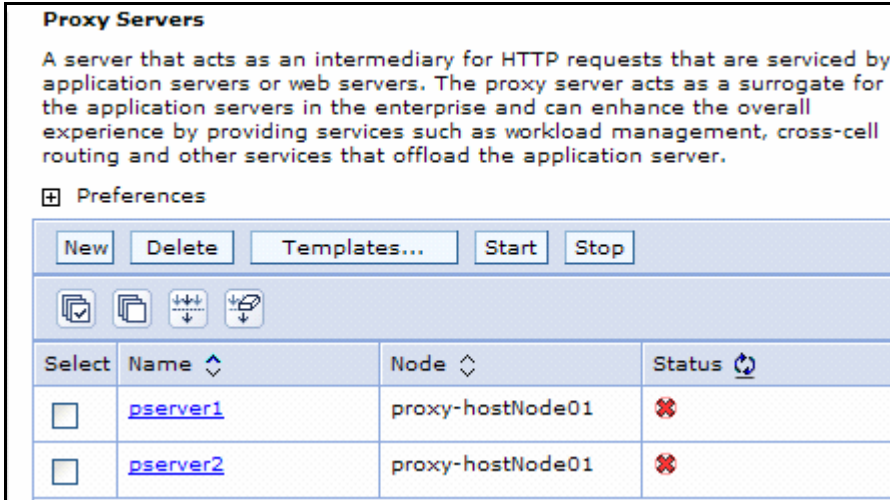


Figure 27-43: Two Proxy servers in the cell.

**Step 10.** Verify the creation and configuration of second proxy server and that it is able to spray requests across the application servers. Before you verify, click on the second **Proxy server** (pserver2) you created, and expand **Ports** to see the ports that have been assigned to this server. Look at the port number that was assigned for PROXY\_HTTP\_ADDRESS (which is 81 in this case).

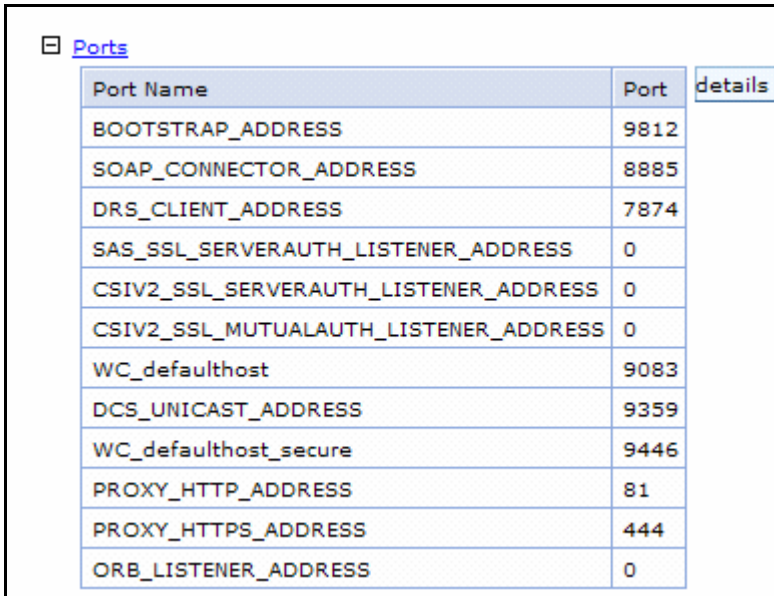


Figure 27-44:

**Step 11.** Verify that port 81 and 444 (for SSL) are added to the host aliases for the default\_host virtual host. (Navigate to **Environment|Virtual Hosts|default\_host|Host Aliases.**) You need to add these ports to the host aliases in order to accept requests that have port 81 in the URL. If you configured your second Proxy server (pserver2) on a separate node (proxy-host2 for example) and used the default port numbers (80 and 443), then you would not need to add modify the host aliases for the default\_host because ports

80 and 443 are already in the aliases list. Use the instructions given in chapter-10 for more information on adding Host Aliases.

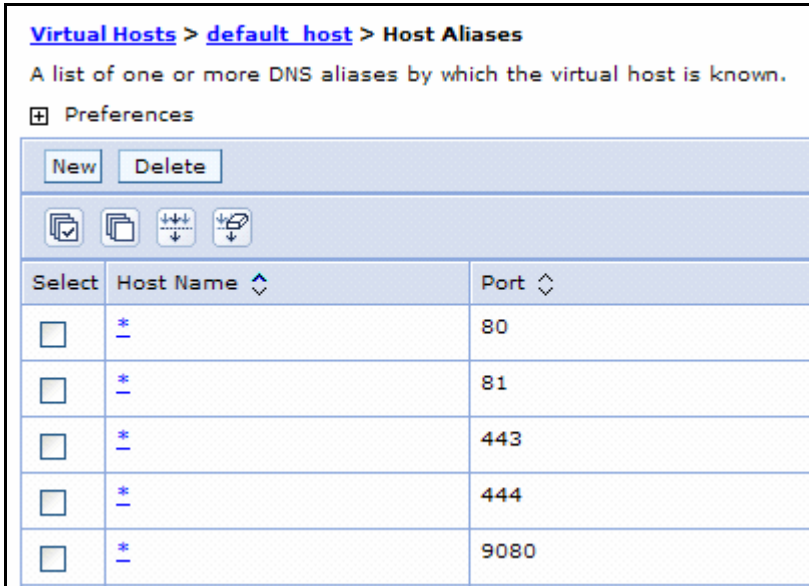


Figure 27-45: Host aliases for the default\_host virtual host

**Step 12.** If the Proxy server receives a request with the port 81 (or 444) in its URL then you need to verify that these ports are added in the host aliases under **proxy\_host**. (Navigate to **Environment|Virtual Hosts|proxy\_host|Host Aliases**.) If you configured your second Proxy server (pserver2) on a separate node (proxy-host2 for example) and used the default port numbers (80 and 443), then you do not have to add those ports to the aliases for proxy\_host because they are configured already.

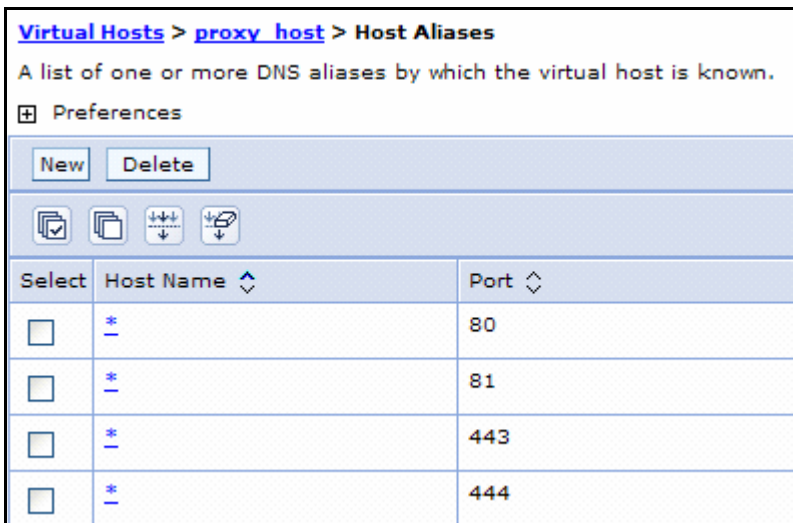


Figure 27-46: Host aliases for the proxy\_host virtual host

### Verify second Proxy server creation

**Step 13.** Start the second Proxy server (pserver1) and verify that you are able to invoke snoop (**http://proxy-host:81/snoop**) and PlantsByWebSphere (**http://proxy-host:81/PlantsByWebSphere**) through the second Proxy server. If you configure the second Proxy server on a separate node using default ports then use http://proxy-host2/snoop and http://proxy-host2/PlantsByWebSphere.

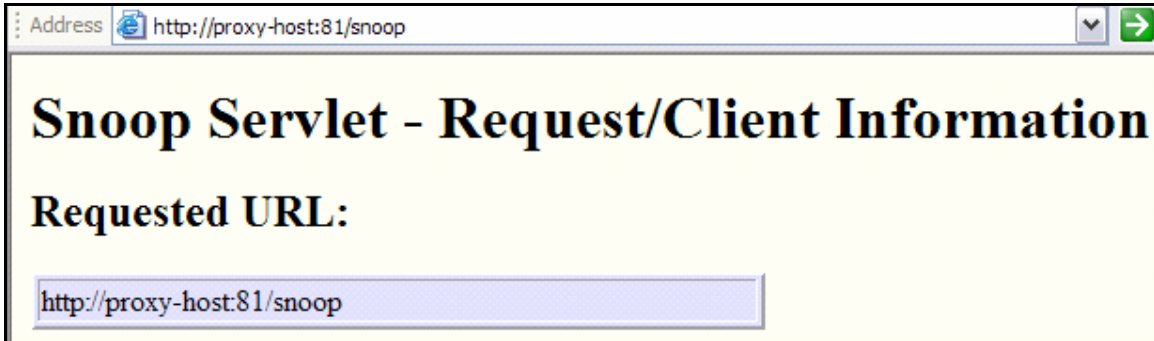


Figure 27-47: Using snoop to verify the second proxy server

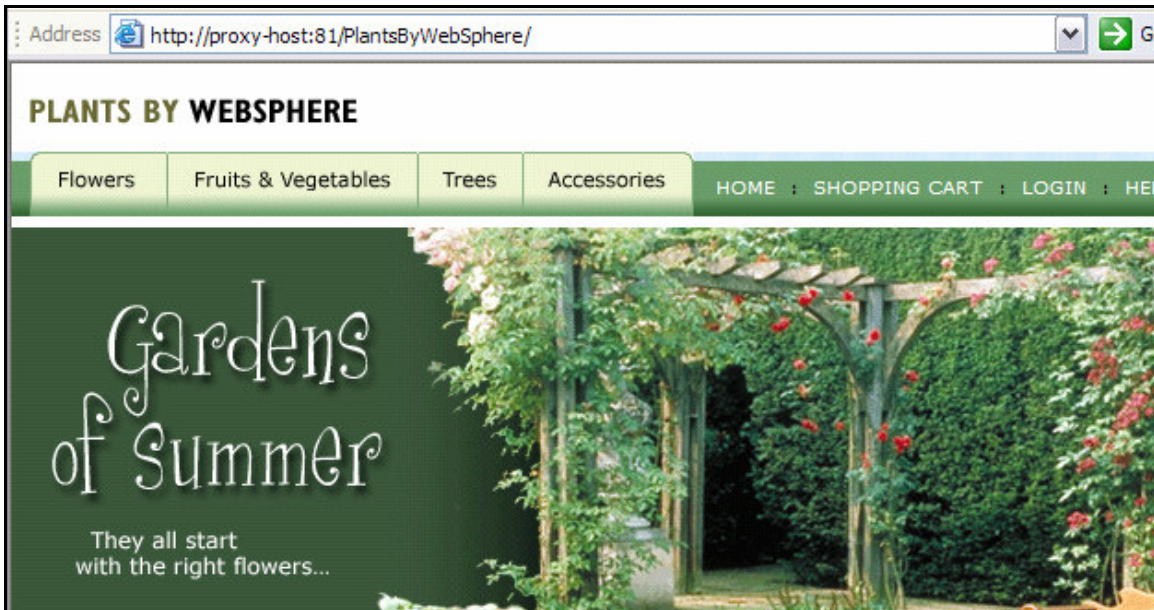


Figure 27-48: Using Plants by WebSphere to verify the second proxy server

**Step 14.** Now use chapter-13 and 14 to install and configure Load Balancer to spray requests across Proxy servers.

## Configure Plug-in as front-end to the Proxy server

The Proxy server node needs a node agent process to manage the node just like any other node in a WebSphere cell. A node agent is a relatively powerful process and should not run in a DMZ where it may be subject to attack. The node agent and Proxy server are Java processes and obviously need a Java Runtime Environment which also tends to be disallowed by network administrators in the DMZ for security reasons. Thus it is not recommended that a Proxy server node be placed in the DMZ. In production environments you may want to place proven reverse proxy servers (HTTP server/plugin) in the DMZ instead of a Proxy server. In this configuration a plug-in will spray requests across Proxy servers (instead of application servers) and Proxy servers spray requests across the cluster members after receiving a request from the HTTP server plug-in as shown in the diagram below. In such a case you do not generate or propagate the plug-in configuration file (plugin-cfg.xml) as described in earlier chapters. Instead, you generate this plug-in file from the Proxy server and propagate it to the HTTP server/plugin node as explained in this section. You also need to disable automatic plug-in file generation and propagation to the Web server's configuration.

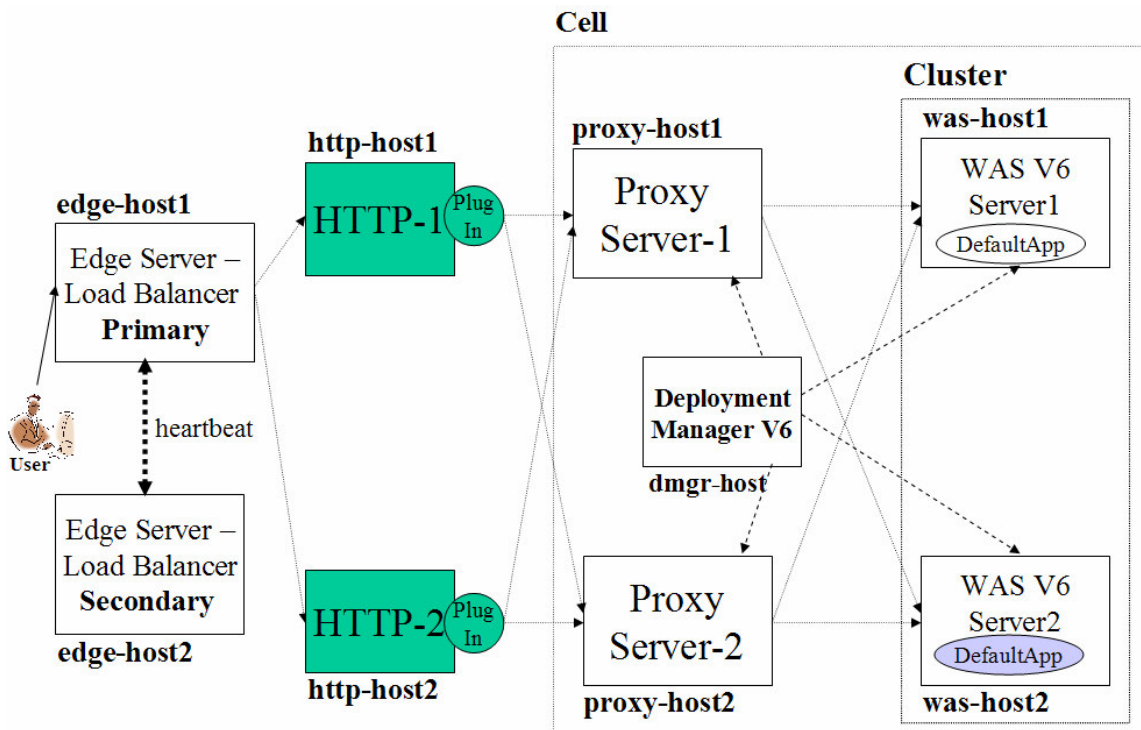


Figure 27-49: Highly available Proxy servers having HTTP servers/Plug-in in DMZ

### Disable Automatic Generation and Propagation of Plug-in file

**Step 1.** If you have configured the WebSphere supported Web server as a managed node (IHS V6 even if it is on an unmanaged node) and enabled the plug-in file auto generation and propagation feature, then disable it now by navigating to **Servers|Web servers| Web server definition |Plug-in properties**. You need to disable

this feature on each Web server you configured by deselecting the two check boxes as shown in the screenshot below. Click on OK and save the configuration. If you have not configured Web servers as managed nodes, then skip this step.

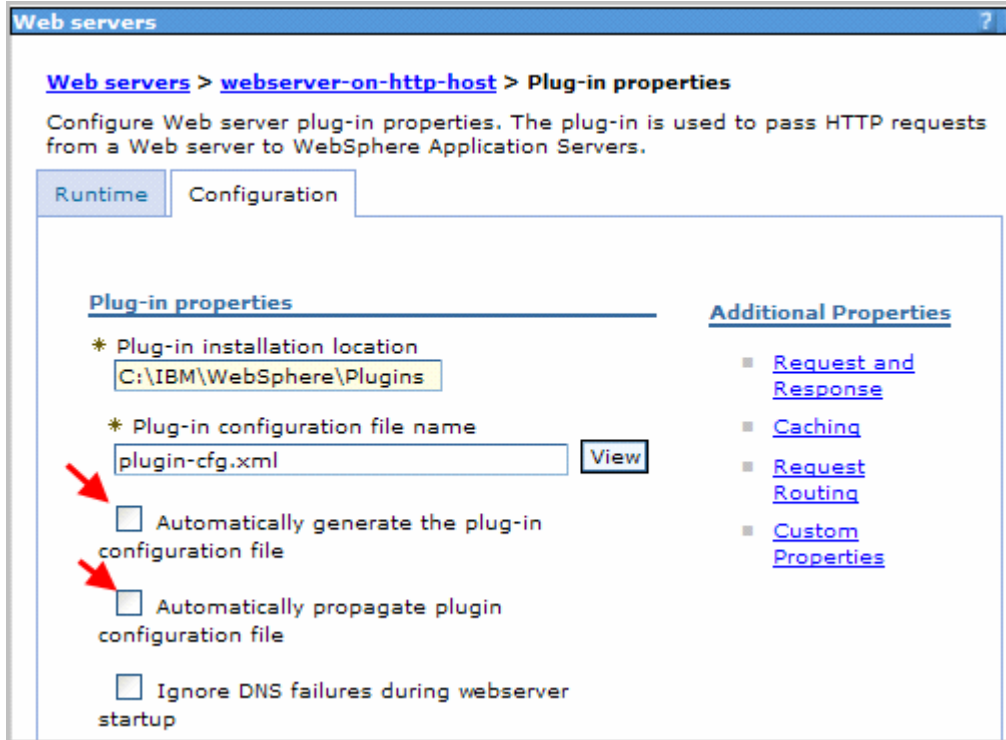


Figure 27-50: Disabling automation plug-in configuration file generation and propagation.

### Configure Proxy Server to Generate and Propagate Plug-in file

**Step 2.** Navigate to **Servers|Proxy servers** and click on the **Proxy server** (pserver1), expand options under **HTTP Proxy Server Settings** and click on **Proxy settings** once again. Scroll down until you see the sub section **Proxy Plug-in Configuration Policy** in this panel. Select **Cell** from the drop down list to **Generate Plug-in Configuration** for the entire cell. In this On Demand Configuration (ODC), if you want to propagate an automatically generated plug-in file to the HTTP server/plugin node, then you need to specify the operating system dependent script file with its full path (cpplugin.bat). You will see contents of this script in the next step. Click on OK and save the configuration. Make sure that you perform this step on each Proxy server.

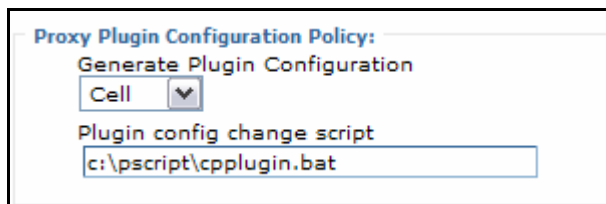


Figure 27-51: Specifying the script for automatic plug-in configuration file propagation

**Step 3.** Navigate to <PROFILE-ROOT>\etc directory of each custom profile where Proxy server has been created (<WASV6-ROOT>\profiles\Proxy01 in our example). You should see the generation of the plugin-cfg.xml file under this directory. In the next step we will create a script file (cpplugin.bat) to copy this plug-in file to the HTTP server/plugin-in node.

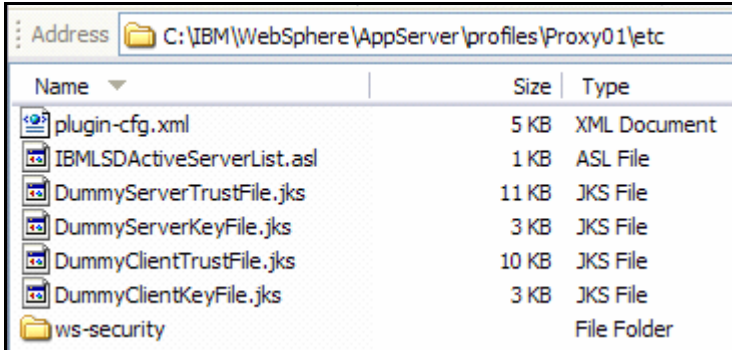


Figure 27-52:

**Step 4.** Before creating the script file, verify the hosts (http-host1 and http-host2 in this example) and the directory under which you need to copy the plug-in configuration file (plugin-cfg.xml). To see this, open the httpd.conf file of IHS V6, scroll down to the end, and see where the plug-in module expects to find the plug-in file. Because we are configuring this on windows environment, we mapped http-host1 as drive Y and http-host2 as drive Z.

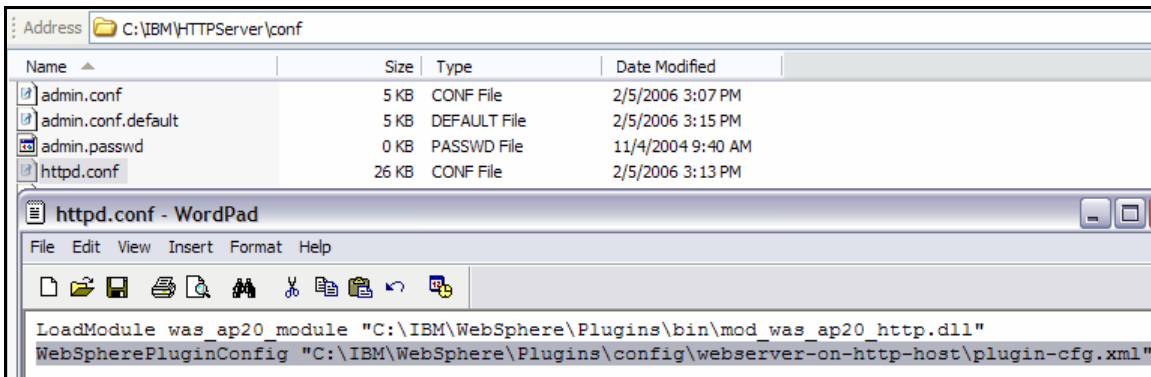


Figure 27-53:

**Step 5.** Create the script file (.bat or .sh) depending on the operating system to copy the plug-in file on each Proxy server. Because we mapped both the HTTP server nodes (http-host1 and http-host2) to drives Y and Z, the script file looks very simple. On UNIX machines, make sure that you give required permissions to run this script and permissions to be able to copy the plug-in file to the destination.

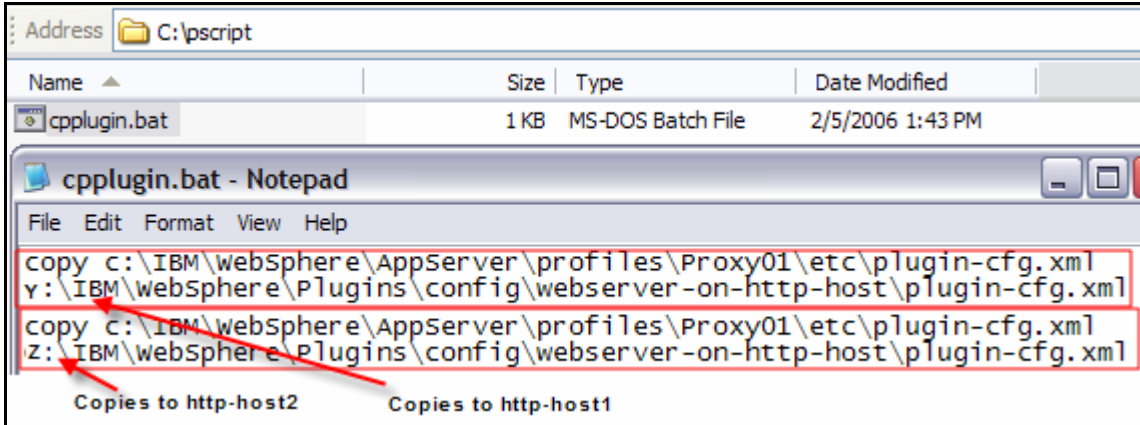


Figure 27-54:

**Step 6.** After creating the script file, examine the destination directory to ensure that the script is able to copy the generated plug-in file automatically after the configuration change by looking at the time stamp of `plugin-cfg.xml` file. Open the propagated plug-in file and look at the contents under `ServerCluster`, and make sure that the plug-in is spraying requests across the Proxy servers (`pserver1` and `pserver2`) instead of application servers in this case. If you have problems copying generated files to the destination, then look for errors in `<PROFILE-ROOT>\logs\pserver\SystemOut.log`. You need to give a full path in the directory structure including the plug-in file as shown in the example script above.

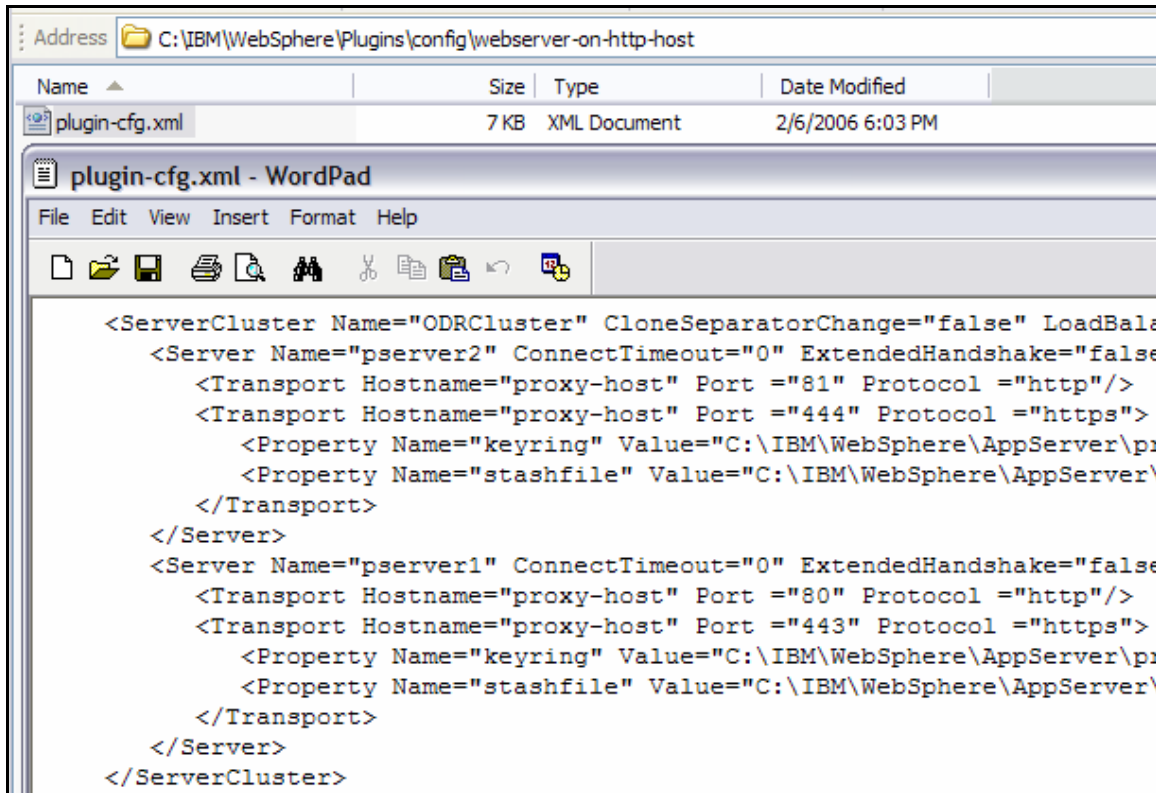


Figure 27-55:

## Verify the configuration

**Step 7.** Make sure the HTTP servers are started after new plug-in file propagation from the Proxy server and verify that you are able to invoke snoop (<http://http-host/snoop>) and PlantsByWebSphere (<http://http-host/PlantsByWebSphere>) through each HTTP server.

**Step 8.** Now use chapter-13 and 14 to install and configure a Load Balancer to spray requests across the HTTP servers.

## SSL configuration in Proxy server

Proxy server uses the default SSL certificates that are shipped with the WebSphere package without any further configuration. But for production environments we suggest you create SSL repertoire with trusted SSL certificate using the instructions given in chapter 18. You also need to configure client (browser or plug-in) and the content servers (application servers) for SSL configuration to function normally.

**Step 1.** To configure SSL between the client (browser or plug-in) and the Proxy server, navigate to Servers|Proxy servers|*proxy server*|HTTP proxy server settings|Proxy server transports and do the following:

1. Select the HTTPS transport chain (**HTTPS\_PROXY\_CHAIN**).
2. Click **SSL Inbound Channel (SSL x)**.

3. From the SSL repertoire drop down list select the SSL repertoire you want to use.
4. Click **OK**, save the configuration and re-start the Proxy server.

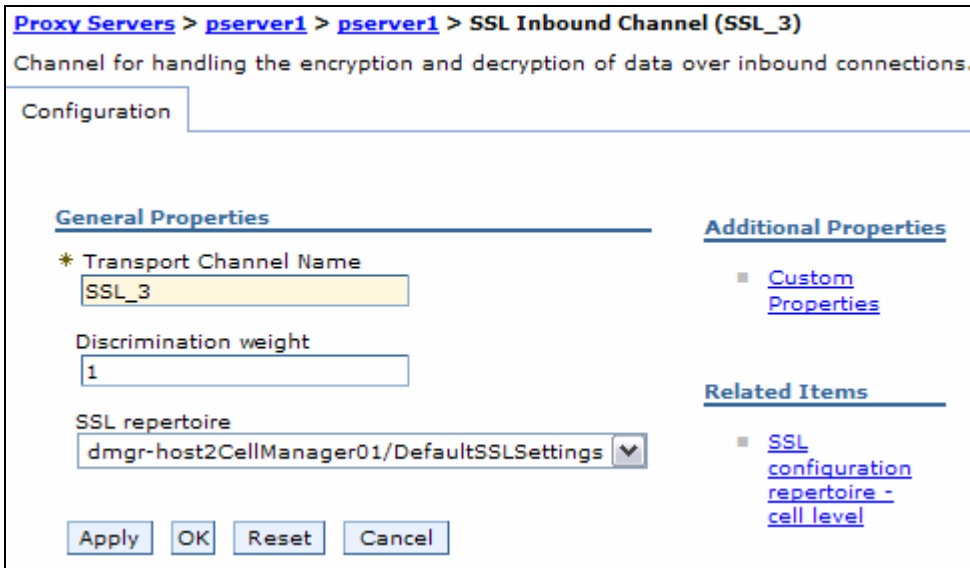


Figure 27-56:

**Step 2.** If you also want to configure SSL between the Proxy and the content servers (application servers), navigate to **Servers|Proxy servers|proxy server|HTTP proxy server settings|Proxy settings** and do the following:

1. Select the out bound SSL alias from the drop down list.
2. Click **OK**, save the configuration and re-start the Proxy server.

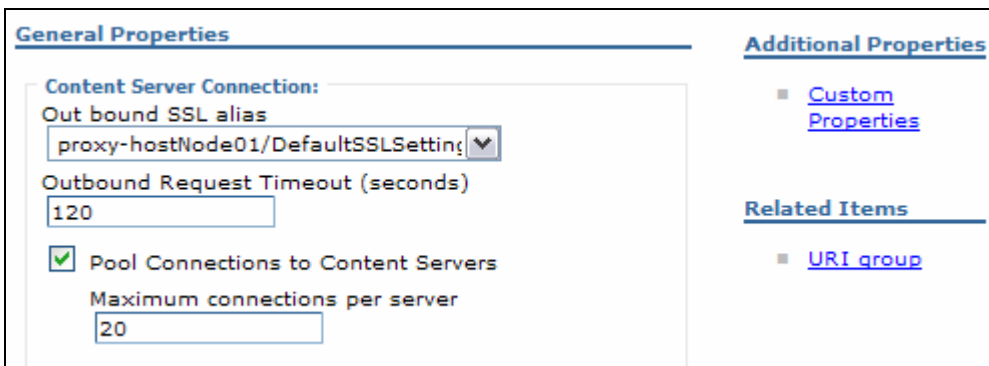


Figure 27-57:

## SSL Termination at the Proxy Server

In enterprise environment with proxy server routing to multiple applications in the content servers, you want to use HTTP protocol between the proxy server and the content for certain applications (or web modules) and still use HTTPS between the client and the proxy server (SSL offload) then do the following:

1. Navigate to **Applications|Enterprise Applications|your application|web modules|your web module|Web Module Proxy Configuration**.

2. Select **HTTP** from the Web Module Transport Protocol drop down list.
3. Click OK and save the configuration.

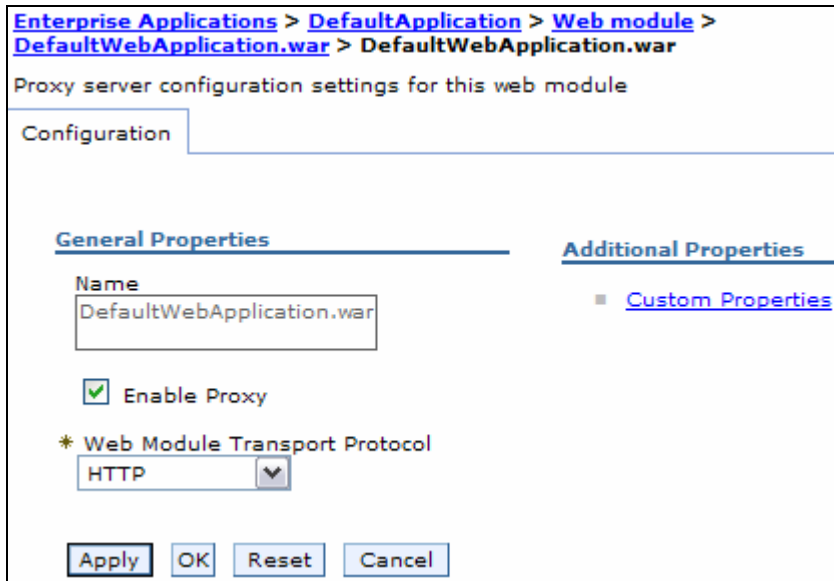


Figure 27-58:

### Disable Proxy server routing to selected web module(s)

After you augment the proxy server functionality in a cell, proxy server routing will be enabled by default for all the web modules. If you want to disable proxy server routing for one or more applications then deselect Enable Proxy in the earlier step.

## Logging During Creation and Management of Proxy Server

The diagram below depicts the logging that takes place during creation, augmenting the profile and management of the Proxy server. Notice that that this logging architecture looks similar to the WebSphere application server and deployment manager.

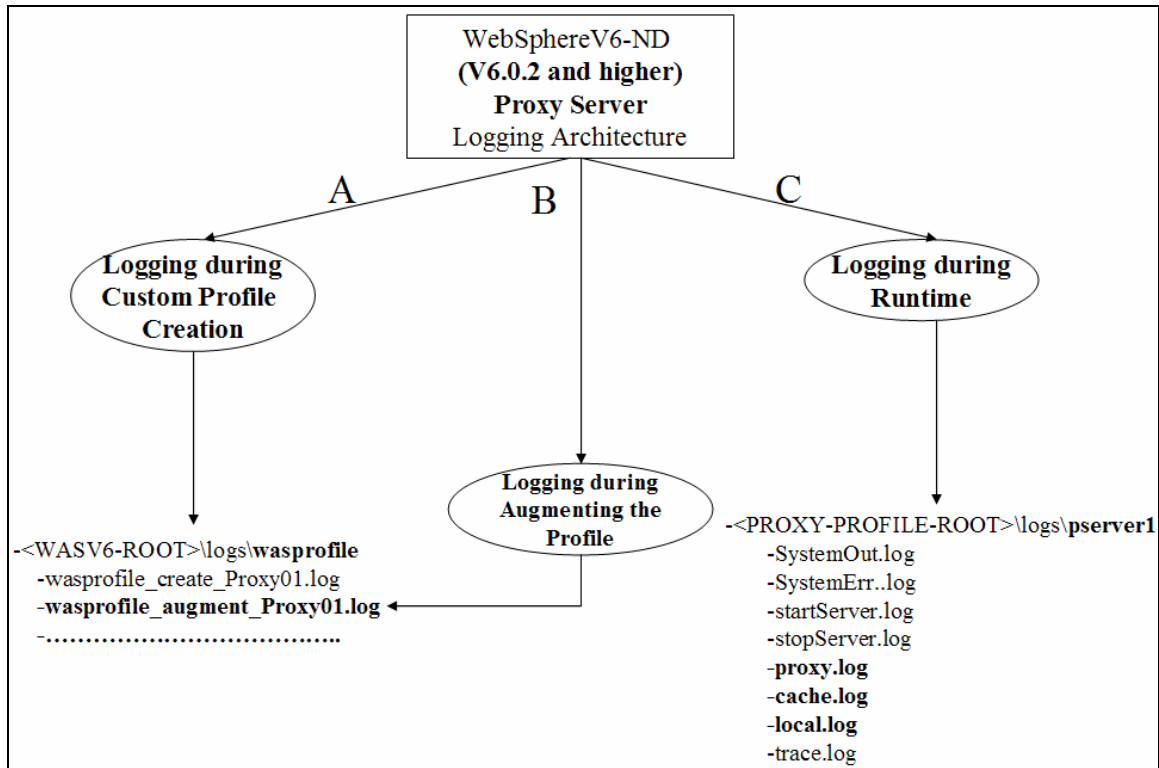


Figure 27-59: Proxy Server Logging

## Cross-Cell Routing Using the Proxy Server

If you have to configure Proxy server(s) in the DMZ zone in a production environment due to various architectural reasons and having an HTTP server/Plug-in as the front end to Proxy server(s) is not a viable solution for your customized environment then we suggest you to configure Proxy server(s) in a separate cell from the cell where your application servers are located as shown in the diagram below. You need to configure core group bridge service on each cell to establish cross cell communication for Proxy server to route requests to different cell(s). This architecture gives the advantage of having all the features available in Proxy server including cross cell routing which are not available in traditional HTTP server/plug-in (as front end to Proxy servers) configuration.

To test the cross cell routing we created two cells (Cell-1 and Cell-2) as shown in the diagram below. Cell-1 is configured with a horizontal cluster with default application running on it. For a matter of fact you can have a single application server on a federated node in this cell to test the cross cell routing (having a cluster is not a requirement). Deployment manager profile on Cell-1 has been augmented to support Proxy server functionality. Cell-2 is configured with a Proxy server on a managed node. Deployment manager and custom profile (on which we created the Proxy server) have been augmented to support Proxy server functionality. We need to configure Core group bridge service on both cells so that the Proxy server on Cell-2 can route requests to applications on Cell-2 (cross-cell routing). Refer to sections at the beginning of this chapter to enable proxy server functionality on profiles.

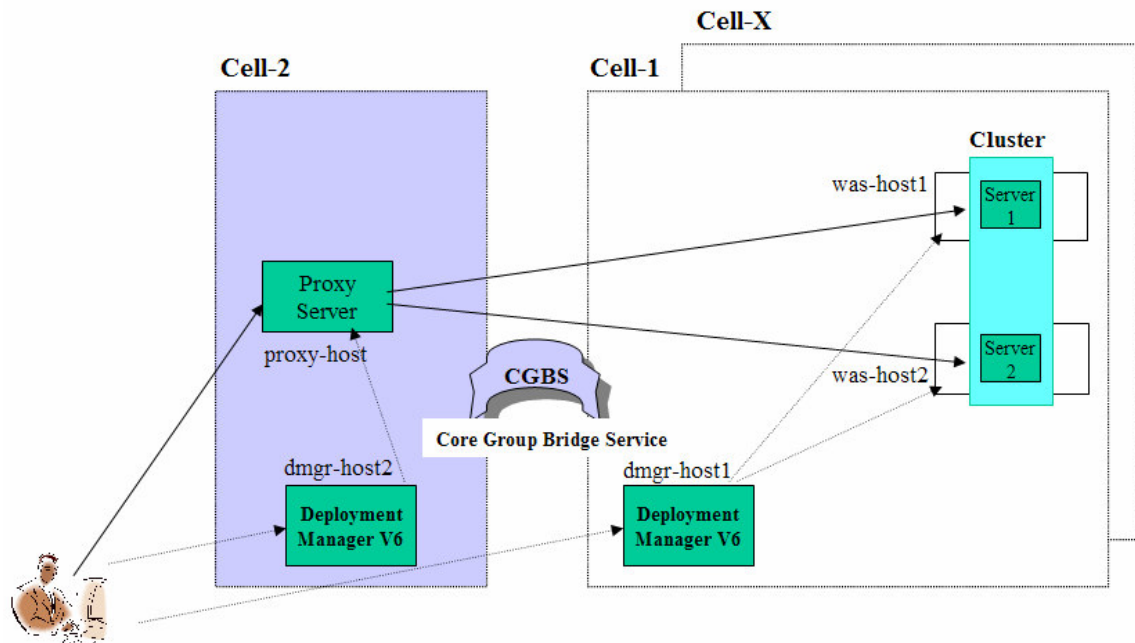


Figure 27-60: Proxy server Cross-cell routing having Proxy Server in DMZ

<begin-note>

As Proxy server is supported on version 6.0.2 and higher releases, we concentrate only the steps required to configure core group bridge service for version 6.0.2 and above. Configuration of core group bridge service had been greatly simplified on V6.0.2 and later by defining custom property **CGB\_ENABLE\_602\_FEATURES** as you will see later in this section. If you are configuring core group bridge service for reasons other than cross-cell routing using the Proxy servers on cells that are below V6.0.2 then you have to consult information center for more information for your environment.

<end note>

Before we actually configure CGBS (Core Group Bridge Service) for Proxy server cross-cell routing, let us try to understand the components involved in cross-cell routing at a high level. Use the diagram below as a reference while we talk about these components. The first task in CGBS configuration is selecting WebSphere processes (node agent or applications server processes for example) to act as bridge interfaces in each cell. Bridge interfaces on each cell will communicate with the other cell through its Core Group Access point. The Core Group Access Point (CGAP) do not represent any processes in a cell, it is a just a logical entity representing the bridge interfaces in that cell. To make this bridge interface service highly available, you want to dedicate two or more processes as bridge interfaces on each cell depending on size of your cell. It is recommended that one use a node agent as bridge interface, as it will have less work compared to an application server process in a cell. As you can see in the diagram below, we chose a node agent process and proxy server as bridge interfaces in cell-2. If you have a second node agent running in this cell (cell-2) then it is recommended to use that node agent instead of the

proxy server. Same way node agent processes on was-host1 and was-host2 were defined as bridge interfaces in cell-1.

Your next task is to create peer access point on one of the cells (we chose to create Peer access point on Cell-2 where Proxy server is available in our sample configuration) by providing Core group access point and bridge server information of the other cell (cell-1 in this case). Peer access point (PAP) do not represent any processes in a cell, it is a just a logical entity representing bridge interfaces on the other cell.

The third task is to define the CGB\_ENABLE\_602\_FETURES custom property on each cell. After enabling this custom property (and restarting the processes), the bridge interfaces and Core group access point on Cell-2 will be recognized automatically by Cell-1 without having to configure Peer access point on Cell-1 (remember that we configured Peer access point on only one cell (Cell-2 in our example)). This reduces a lot of administrative steps that have to be performed on each cell. This feature is not available in prior versions of V6.0.2.

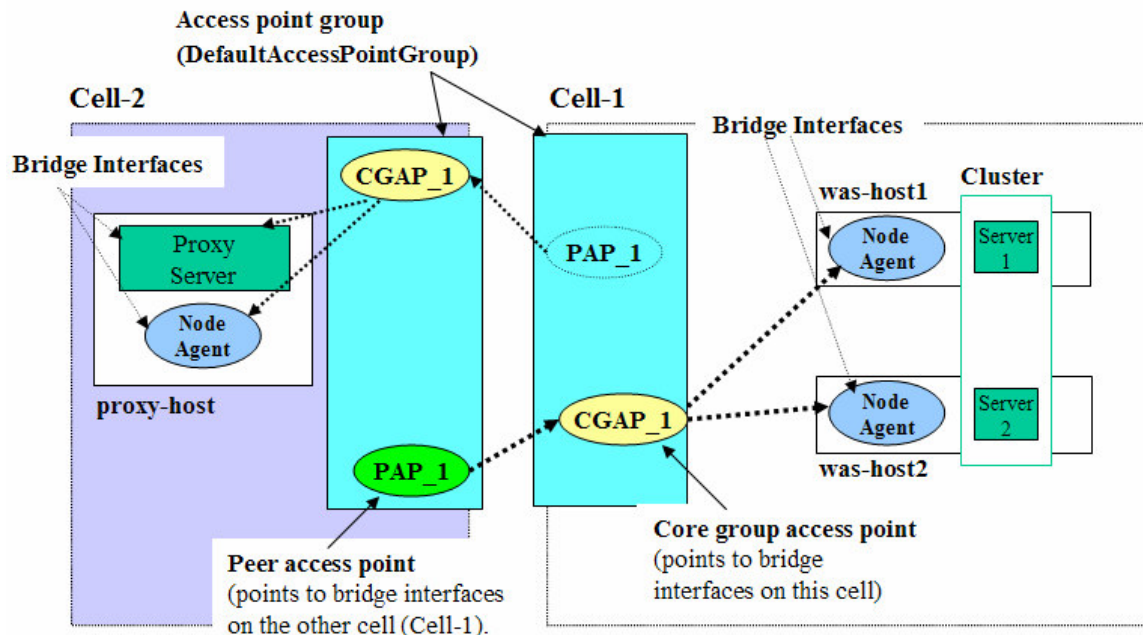


Figure 27-61: Simplified architecture of Core group bridge service

### Access Point Group

The Access point group defines set of core groups that communicate with each other. Refer to chapter-12 for more information on core groups. In cross cell communication, an access point group contain one Core group access point (CGAP\_1) with one or more Peer access points (PAP\_1). In our sample configuration we will have only one Peer access point (for cell-2) as it needs to talk to only one other cell (Cell-1). If Cell-2 needs to communicate with another Cell (Cell-x apart from Cell-2 for example) then you need to configure two Peer access points one for each backend cell.

When you install The WebSphere V6 ND package and create deployment manager profile, WebSphere automatically creates a default access point group called DefaultAccessPointGroup. To verify this default configuration navigate to **Servers|Core groups|Core group bridge settings** on deployment manager's admin console of each cell (Cell-1 and Cell-2) as shown in the screenshot below. You need to verify that the name of the access point group (DefaultAccessPointGroup in this example) must be same across all the cells that are trying communicate with each other.

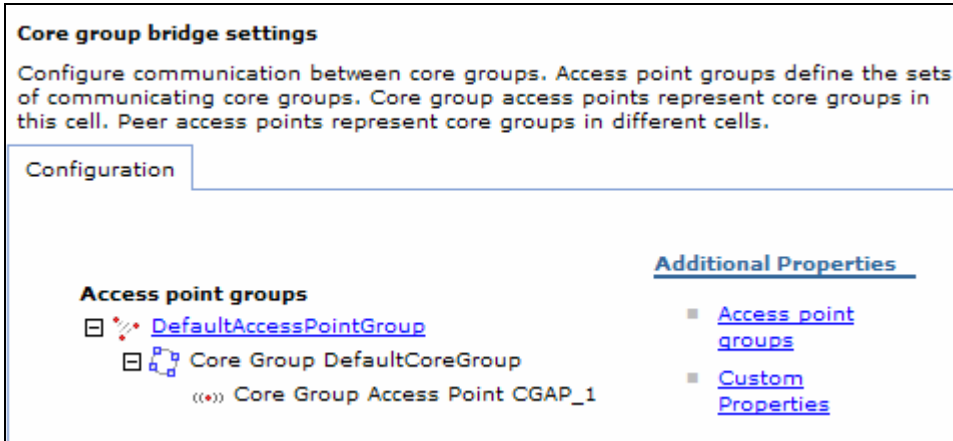


Figure 27-62

### *Configure Core Group Access Point*

In this task we will define bridge interfaces under Core group access point (CGAP\_1) on each cell. Let us start with Cell-2 where Proxy server is configured.

**Step 1.** Navigate to **Servers|Core groups|Core group bridge settings|DefaultAccessPointGroup|Core group access points**. Select **CGAP\_1|DefaultCoreGroup** and click on **Show Detail** button as shown in the screenshot below.

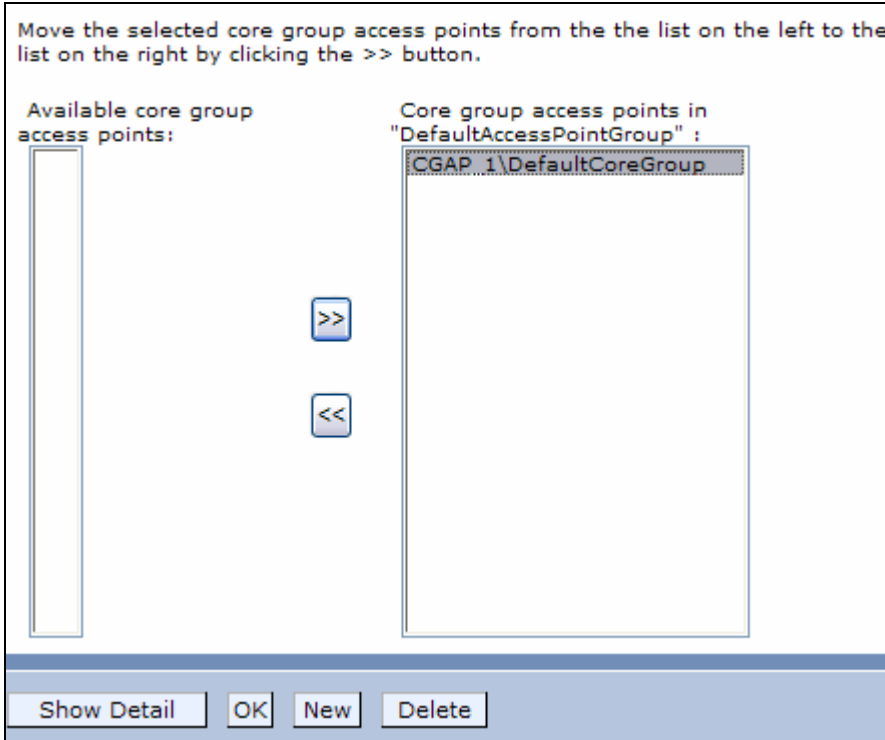


Figure 27-63

**Step 2.** On the resulting screen, click on **Bridge interfaces** link. Click on **New** to define the bridge interface.

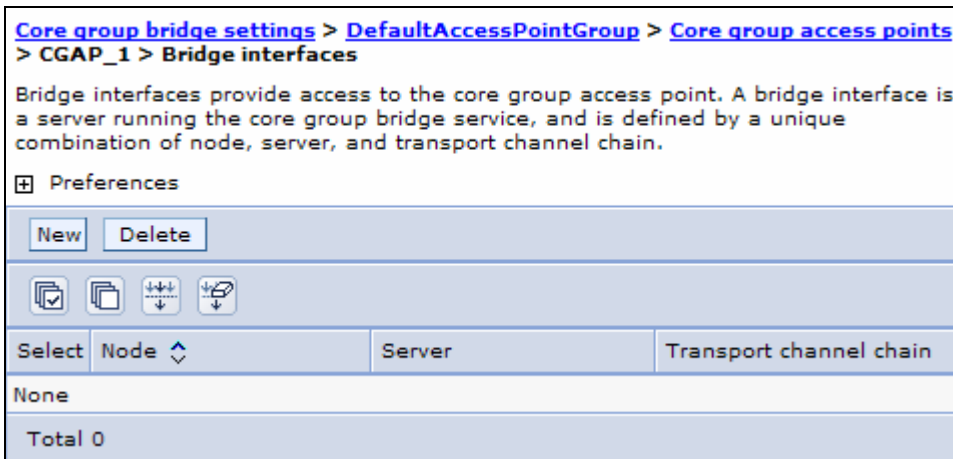


Figure 27-64

**Step 3.** Select the node agent on Proxy server node as one of the bridge interfaces as shown in the screenshot below and click on OK.

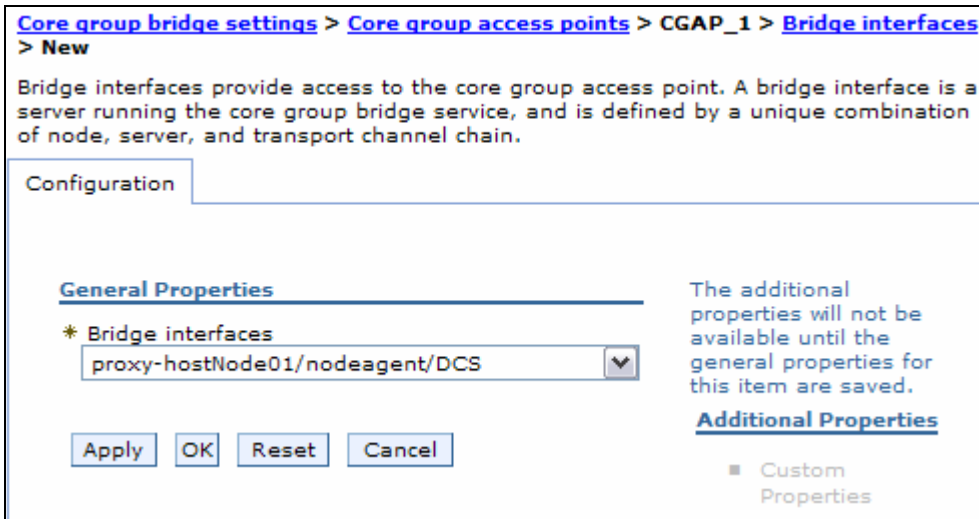


Figure 27-65

**Step 4.** Click on **New** again to define the second bridge interface. On the resulting screen select Proxy server as the second bridge interface as shown in the screenshot below. Click on OK and save the configuration. In a production environment it is recommended to have a second Proxy server (preferably on a different node) for high availability. In such a case select the node agent process on the second proxy server node instead of proxy server as a bridge interface.

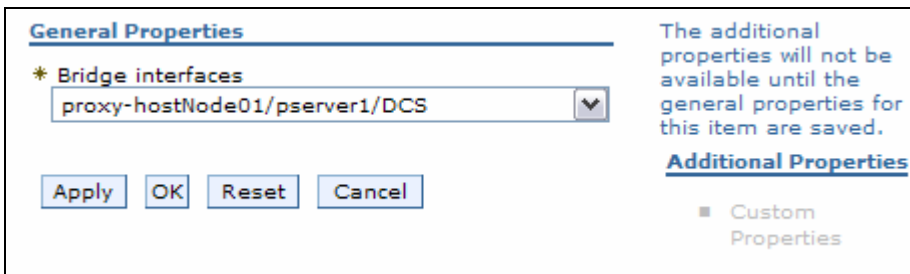


Figure 27-66

**Step 5.** After successful completion of above steps your configuration for Core Group Access Point of Cell-2 should look similar to the one shown in the screenshot below. Port numbers and names of bridge interfaces may vary depending on your configuration.

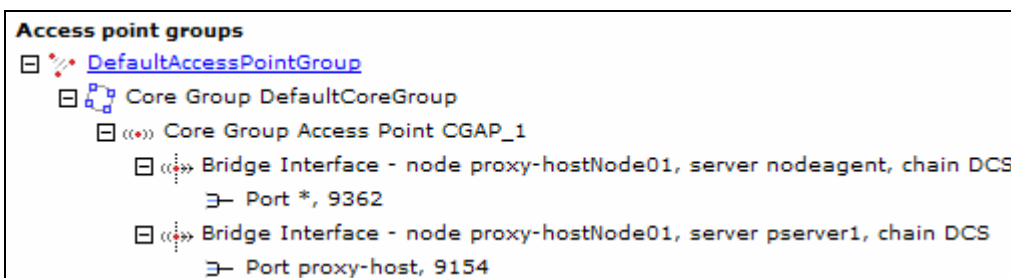


Figure 27-67

**Step 6.** Now we are done with configuring CGAP on Cell-2, let us configure CGAP on cell-1 where application servers are available. Use steps 1 thru 5 to configure

bridge interfaces on Cell-1. This time use node agent processes on was-host1 and was-host2 as bridge interfaces

After successful completion of these steps on Cell-1, your configuration should look similar to the one shown in the screenshot below. Port numbers and names of bridge interfaces may vary depending on your configuration. Note down the cell name (navigate to **System administration|Cell** to find the cell name), core group (DefaultCoreGroup), core group access point (CGAP\_1), hostname/port numbers (was-host1/9356, was-host2/9357) for this cell. You need to provide this information while configuring Peer access point on the other cell (Cell-2).

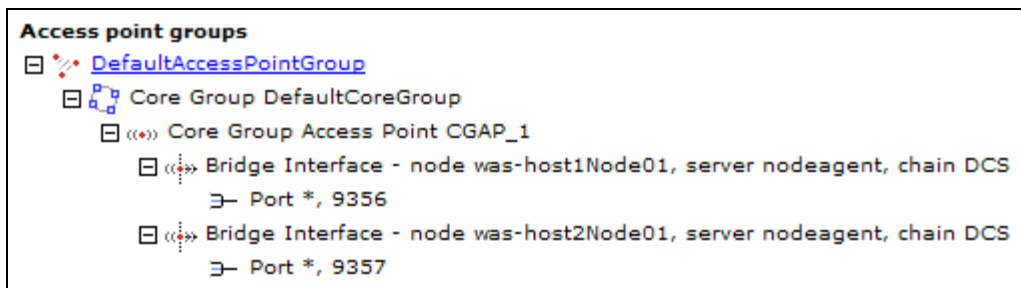


Figure 27-68

**<begin note>**

If you have more than one cell at the backend (Cell-x apart from Cell-1 for example) and you want Proxy server to route to more than one cell then you need to perform all the steps under **Configure Core Group Access Point** task on each backend cell. If you have more than one cell at the backend then you need to have a unique application (application with unique context root) on these cells. This means that Proxy server cross-cell routing does not work if the same application (with the same context root) is running on multiple backend cells.

**<end note>**

### *Configure Peer Access Point*

Our next task is to configure Peer access point on one of the cells. For our sample configuration we chose to configure it on Cell-2 where proxy server is running. We configure Peer access point on Cell-2 by giving information about the CGAP and bridge interfaces information on the other cell (Cell-1 in this example).

**Step 7.** Navigate to **Servers|Core groups|Core group bridge settings|DefaultAccessPointGroup|Peer access points**. Click on **New** button to create peer access point.

**Step 8.** Enter the configuration information about the other cell you noted down in step 6 as shown in the screenshot below. For the sample configuration we named the peer access point as PAP\_1. Click **Next**.

The screenshot shows a dialog box titled "Create new peer access point". On the left, a blue sidebar contains three steps: Step 1 (highlighted with a yellow arrow), Step 2, and Step 3. Step 1 is "Specify the name, cell, core group and core group access point". The main area contains four fields, each with a yellow asterisk icon: Name (PAP\_1), Cell (dmgr-hostCell01), Core group (DefaultCoreGroup), and Core group access point (CGAP\_1). At the bottom, there are "Next" and "Cancel" buttons.

Figure 27-69

**Step 9.** Select the **Use peer ports** radio button and, enter the hostname and port number of the first bridge interface on the other cell you noted down in step 6 as shown in the screenshot below. Click **Next**.

The screenshot shows the same dialog box, now at Step 2: "Specify either a peer port or a proxy peer access point". The "Use peer ports" radio button is selected. Below it are fields for Host (was-host1) and Port (9356). The "Use a proxy peer access point" radio button is unselected, and its associated dropdown menu is disabled. At the bottom, there are "Previous", "Next", and "Cancel" buttons.

Figure 27-70

**Step 10.** Click **Finish** on the summary screen and save the configuration.

**Step 11.** We still need to enter the second bridge interface information of the other cell. To do that, select the peer access point (PAP\_1) you just created and click on **Show Detail** button as shown in the screenshot below.

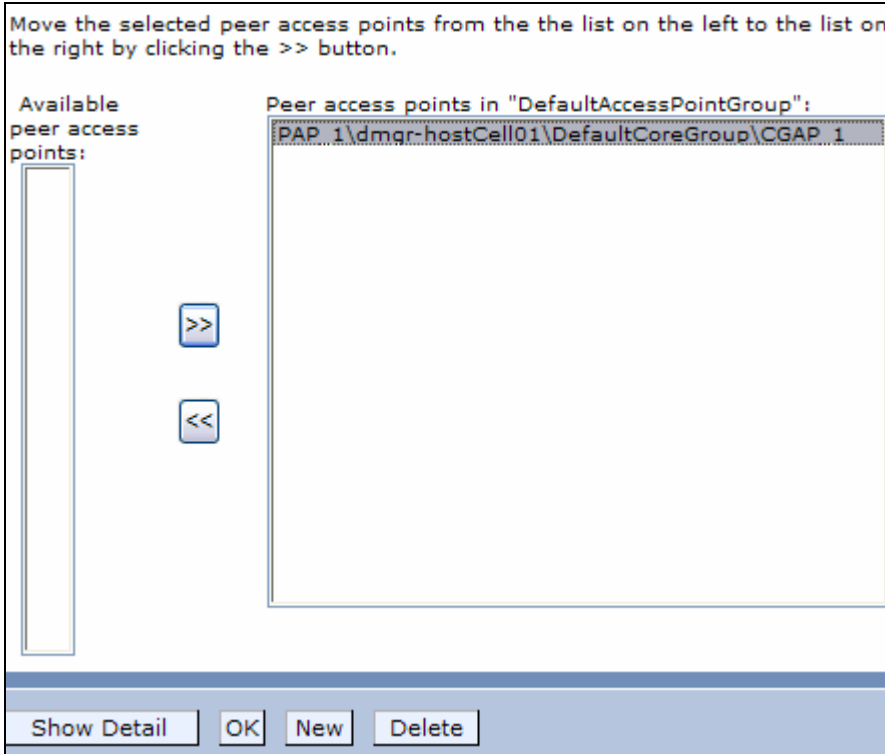


Figure 27-71

**Step 12.** In the resulting screen, click **Peer ports** under Peer addressability section as shown in the screenshot below.

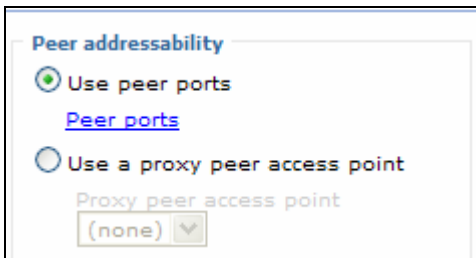


Figure 27-72

**Step 13.** On the resulting screen you will see an entry for the first bridge interface you provided in earlier steps. Click **New** to provide information about the second bridge interface.

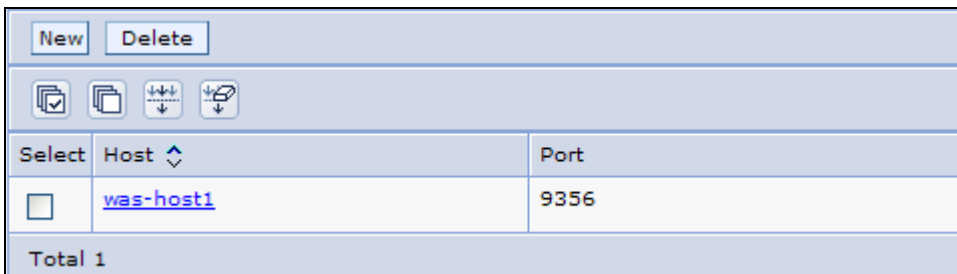


Figure 27-73

**Step 14.** Enter the hostname and port number of the second bridge interface on the other cell you noted down in step 6 as shown in the screenshot below. Click **OK** and save the configuration.

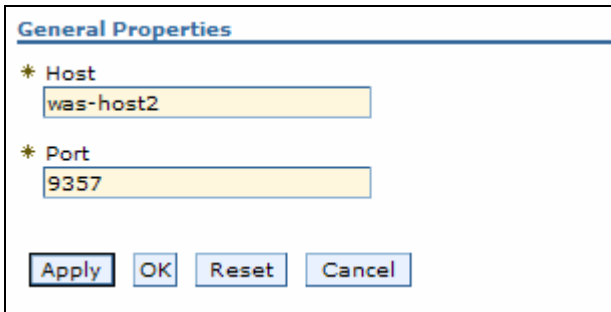


Figure 27-74

**Step 15.** After successful completion of above steps your configuration for Peer Core Group of Cell-2 should look similar to the one shown in the screenshot below. Port numbers and names of bridge interfaces may vary depending on your configuration.

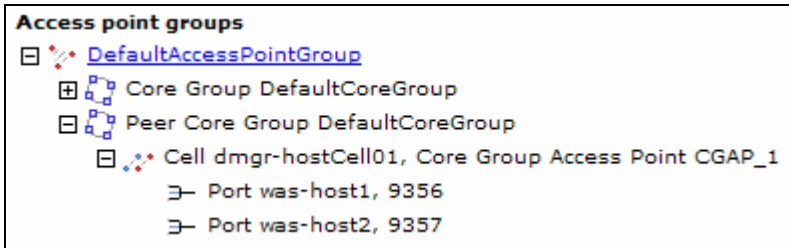


Figure 27-75:

**<begin note>**

If you have more than one cell at the backend (Cell-x for example) then you need to configure peer access point (on Cell-2) for each additional cell this proxy server is trying to route to. Unlike Core Group Access Point, you configure Peer Access Point on only one cell (Cell-2 in this example) as we are using CGB\_ENABLE\_602\_FETURES custom property. You also need to perform all the steps under **Configure Core Group Access Point** task on each backend cell before configuring peer access point on this cell.

**<end note>**

### Configure Custom Property

**Step 16.** The last step of CGBS is defining custom property named CGB\_ENABLE\_602\_FETURES for the access point group (DefaultAccessPointGroup) we worked on so far. You need to perform this step on each cell that is involved in the cross cell routing (Cell-1 and Cell-2 in our sample configuration). Navigate to **Servers|Core groups|Core group bridge settings|DefaultAccessPointGroup**, click on **Custom properties** under additional properties section. Click on **New** to define the custom property (CGB\_ENABLE\_602\_FEATUES), value (true) and, optionally, a description as shown in the screenshot below. Click on **OK** and save the configuration.

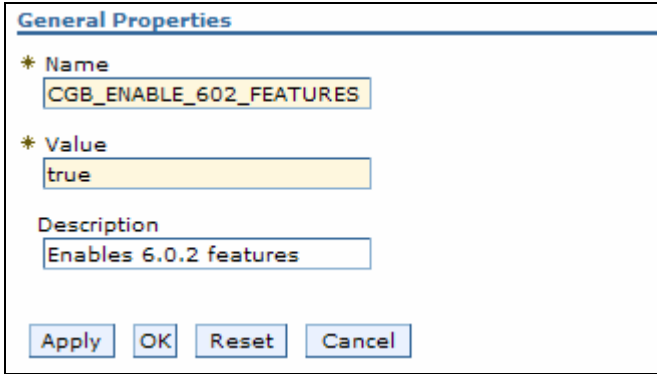


Figure 27-76

### Verify Cross-Cell Routing

After configuring Core Group Access Point (CGAP\_1) on all the cells (Cell-1 and Cell-2 in our configuration) and Peer Access Point (PAP\_1) on one of the Cells (Cell-2) in our example, you are ready to test Core Group Bridge Settings and cross-cell routing through the Proxy server.

**Step 17.** Stop all application server processes including proxy servers, node agent processes on all the cells (Cell-1 and Cell-2 in this case).

**Step 18.** Remove log files under each <PROFILE-ROOT>\logs directory. Back them up if necessary before you delete them.

**Step 19.** Re-start the node agent processes on Cell-1 where you have application servers are configured.

**Step 20.** Re-start the node agent processes on Cell-1 where you have application servers are configured.

1. Restart the cluster(s) or application servers in this cell.

**Step 21.** Re-start the node agent processes on Cell-2 where you have proxy server(s) configured.

1. Re-start the Proxy server(s).

**Step 22.** Open SystemOut.log of node agent processes and proxy server that have been used as bridge interfaces. You should see that bridge interface has been connected to the other cell as shown in the screenshot below. The log file below is indicating that bridge server (node agent) on Cell-1 has been connected to bridge server on Cell-2.

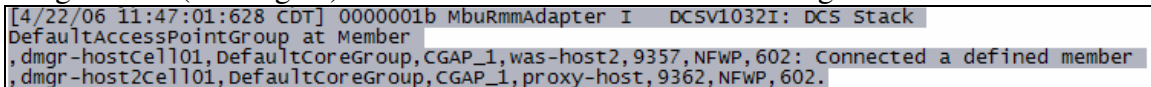


Figure 27-77

**Step 23.** Open a browser and invoke snoop servlet through the proxy server to test the cross-cell routing (<http://proxy-host/snoop>). If you have a problem invoking the snoop servlet then review the troubleshooting section given below.



Figure 27-78

### Troubleshooting Cross-Cell Routing

If you have problem routing requests then double check the configuration steps again. Also check the spelling of CGB\_ENABLE\_602\_FETURES and its value has been set as true. You can get more information by turning on tracing on bridge interfaces and proxy servers.

**Step 24.** To setup tracing on node agents that have been used as bridge interfaces, navigate to **Troubleshooting|Logs and Tracelnode agent|Diagnostic Trace|Change Log Detail Levels** and change the setting to **\*=info:com.ibm.ws.cgbridge.\*=all:com.ibm.ws.ODCTreeImpl\$Save=all**, as shown in the screenshot below. Click on OK save the configuration.

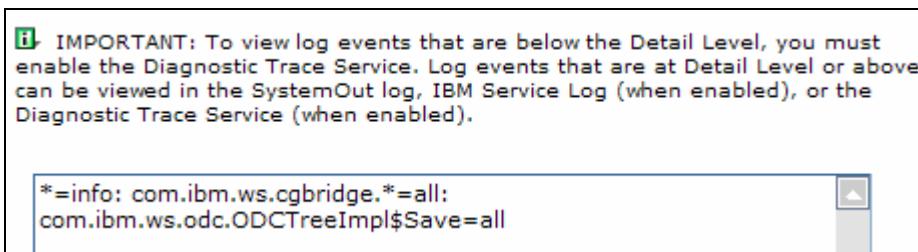


Figure 27-79

**Step 25.** To setup tracing on Proxy server(s), navigate to **Troubleshooting|Logs and Tracelproxy server|Diagnostic Trace|Change Log Detail Levels** and change the setting to **\*=info:com.ibm.ws.proxy.\*=all: com.ibm.ws.dwlm.\*=all:com.ibm.ws.odc.ODCTreeImpl\$Save=all**, as shown in the screenshot below. Click on OK save the configuration.

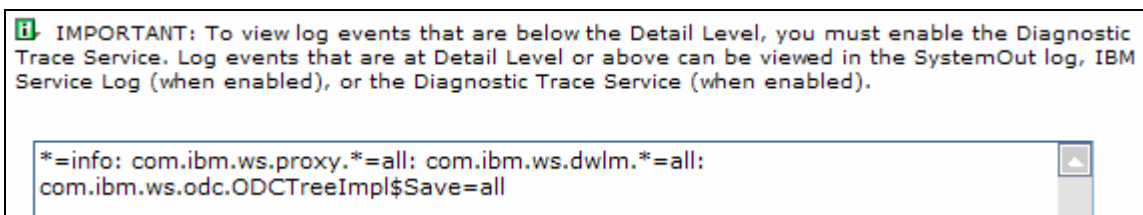


Figure 27-80

***Step 26.*** Follow the steps given under Verify Cross-Cell Routing task given above to test the routing again. Review trace.log under <PROFILE-ROOT>\logs directory for more information on the failure of cross-cell routing.

-30-  
-30-