Chapter 28 - Share WebSphere Application Server V6 Product Binaries across Nodes

Introduction

This chapter describes a process that enables you to share one WebSphere Application Server V6 installation (product binaries) among many nodes (and even cells). This chapter is for the architects and administrators installing WebSphere V6 on UNIX/Linux/AIX based platforms in large enterprises that have a number of WebSphere server farms and want to reduce administrative time and effort (and save disk space) by reusing the product binaries across multiple nodes.

Software installed on the WebSphere nodes must be serviced or updated, and there is no way to do this other than servicing each image as if it were a standalone server. Most operating systems (and hardware) support the ability to share the file system across many nodes, so there ought to be a way to install WebSphere once and use that installation for many nodes. This was previously impractical in earlier versions of WebSphere (V5.1 and below), because the installed WebSphere program files and the installation specific data files were not decoupled sufficiently to allow general reuse of the installation.

WebSphere Version 6 introduced the concept of "profiles", which are created by the wasprofile command or through the profile creation wizard. Profiles must be created for all installations of WebSphere V6, and contain all the user-writable files (configuration, logs, installed applications, etc) for WebSphere. All the profiles on a single host can share the same WebSphere binaries from the WebSphere installation directory. Software updates applied to the common WebSphere product binaries apply to all the profiles using those binaries. With WebSphere version 6, the WebSphere binaries are separate from the installation specific data files. We can use this separation to allow profiles to share the WebSphere binaries not only on the same server, but across many servers. Refer to chapters 1, 3 and 7 for more information about product binaries, profiles and installation procedures.

The advantages you get with this procedure are:

- Install WebSphere only once.
- Save about 1GB of disk space per WebSphere node.
- Update many profiles easily to the latest WebSphere maintenance level.

Since a single installation of the product binaries are supporting potentially many WebSphere nodes in your server farm, you will want to make the shared file system highly available using a technology that is suitable for your platform to avoid a single point of failure (SPOF). You will also need to work closely with the system and network administrators to successfully configure this environment as you are going to install the product binaries on a highly available shared file system. (On UNIX machines you can dedicate a file system to install WebSphere binaries.)

<begin note>
You need to configure and verify this scenario in development and test environments before setting this up in the production environment. Check with IBM support if they support this configuration for your platform, WebSphere version/fix pack level in production environment. The authors did not test this configuration for performance in a production environment, so do this at your own risk.
<end note>

Configure Shared Product Binaries Environment

As shown in the diagram below (Figure 28-1), we will install WebSphere V6 and apply fix packs on binary-host and share the product binaries that are installed on binary-host among all the WebSphere nodes. We performed the steps for this chapter on a **Linux** environment by mounting a file system on the WebSphere installation directory (<WASV6-ROOT>) on binary-host. We also set up this file system (NFS mount) with read-only access from all the WebSphere nodes (was-host1, was-host2, dmgr-host). We are assuming that you are using the Network Deployment package, but the procedure is similar for the Base or Express packages. We performed all the steps logging in as root user.

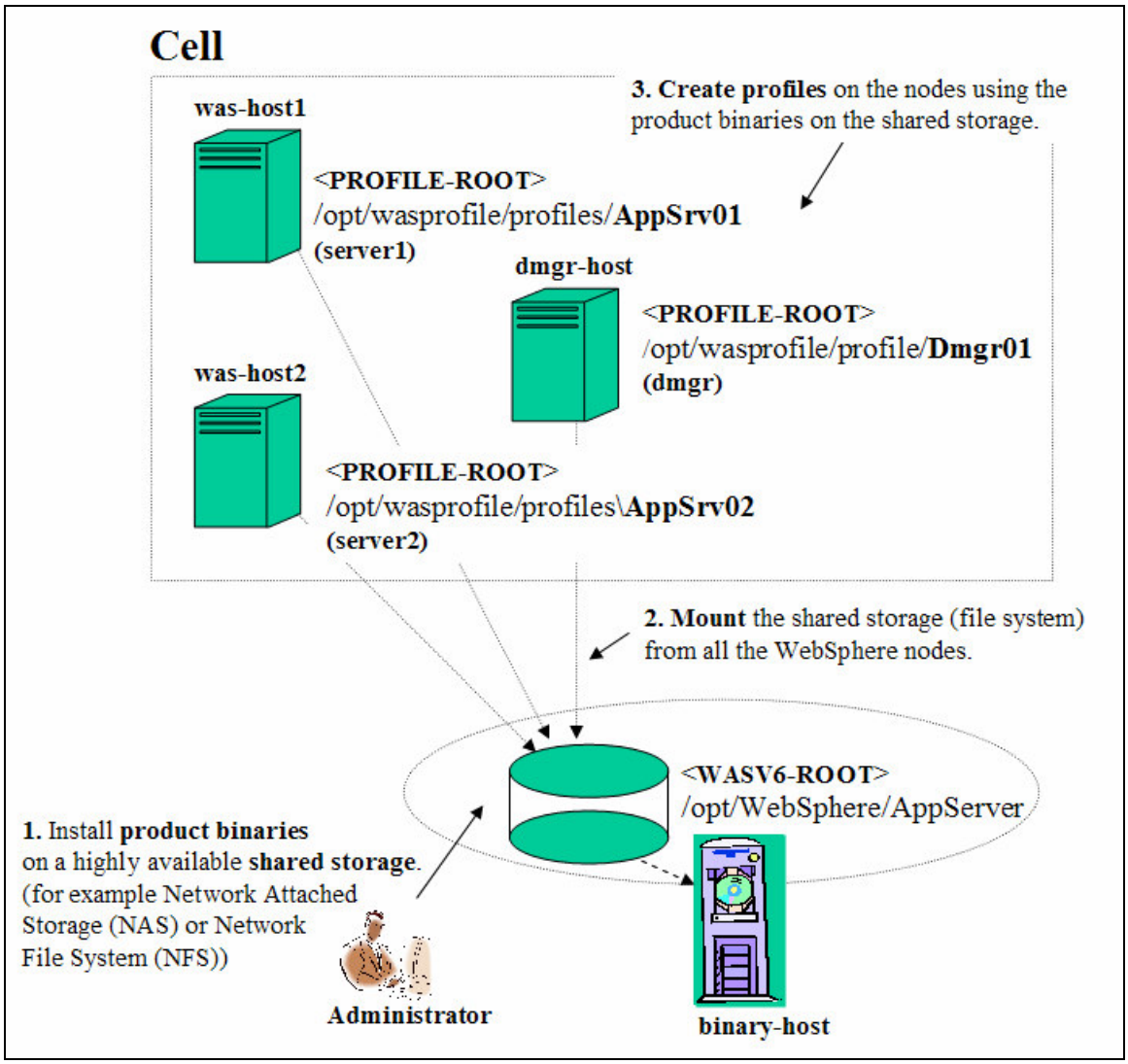

Figure 28-1: Sharing WebSphere Product Binaries across Nodes in a Cell

Task 1: Install WebSphere product binaries on a shared file system. Use Chapter 7 to install and verify product binaries (phase-1 only) on binary-host. In our sample configuration we installed the product binaries in the /opt/WebSphere/AppServer directory. (A separate file system had been mounted on this directory to install the product binaries alone. Creating a separate file system is recommended but not required. If you are configuring this for testing purposes, simply create a directory /opt/WebSphere/AppServer). This file system will have read/write (R/W) access when you log on to the binary-host. Refer to chapter 7 for system and disk space requirements. From now on we represent it as <WASV6-ROOT>. Do not create any profiles (phase-2) on this node and skip the profile creation wizard during the ND installation.

I. The procedure below is done just once after successfully installing and verifying WebSphere Application Server V6 product binaries (phase-1) on binary-host, and will be used for all profiles you create on WebSphere nodes in the future.
1.1. Log on to the binary-host

1.2. Edit the file *<WASV6-ROOT>/properties/wasprofile.properties*
The wasprofile command or the profile creation wizard tries to write information about the profiles it creates into the WebSphere install directory (which is read-only from the WebSphere nodes). We will tell it instead to put those files into the local file system of the WebSphere node where you are creating a profile.
1.3. Change the default location of the logs written by the wasprofile command or the profile creation wizard.

- **Change:** WS_CMT_LOG_HOME=${was.install.root}/logs/wasprofile
  To: WS_CMT_LOG_HOME=/opt/wasprofile/logs
  (You will be creating this directory on each WebSphere node where you are creating a profile.)

1.4. Change the default location of the WebSphere profile registry.

- **Change:**
  WS_PROFILE_REGISTRY=${was.install.root}/properties/profileRegistry.xml
  To:
  WS_PROFILE_REGISTRY=/opt/wasprofile/properties/profileRegistry.xml
  (You will be creating this directory on each WebSphere node where you are creating a profile.)

II. As all the WebSphere nodes (was-host1, was-host2, dmgr-host) are going to use the product binaries installed in /opt/WebSphere/AppServer directory on this machine, we performed the following steps from the command line on the binary-host machine. After performing these steps all the nodes specified in the configuration can access product binary files remotely. You may have to do extra steps depending on your environment. Refer to "http://nfs.sourceforge.net/nfs-howto" for more information on this topic.

1.5 Edit /etc/exports file to include the ip-address or hostnames or DNS of each WebSphere node with read only (ro) access on product binary directory as shown in the screenshot below.

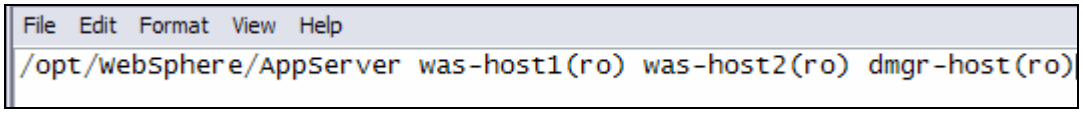

Figure 28-1a: Specifying WebSphere nodes information in /etc/hosts file

1.6 Start the following daemons on binary-host if they are not already running
- rpc.portmap
- rpc.mountd
- rpc.nfsd

<begin note>
Use chapter 3 if you are using the Base or Express package. As explained in chapter 3, these packages will create a default profile (phase-2) at the time of installing the product binaries. But you may or may not want to use this default profile and create required profiles on other WebSphere nodes instead and share only the product binaries.
<end note>

Task 2: Map shared storage on all the WebSphere nodes. Work with your System and/or Network administrator if you do not have sufficient rights or access to mount the file system of for the WebSphere product binaries on binary-host from each WebSphere node (was-host1, was-host2, dmgr-host). (In our example we are using NFS for remote file system sharing.) You are going to create profiles on these nodes. You need read-only (R/O) access from these nodes to the shared file system. Create directory /opt/WebSphere/AppServer on each node to mount the file system. You will be mounting the file system containing the product binaries to the same directory you used before (/opt/WebSphere/AppServer) on each WebSphere node. For example you use a similar command to mount the file system (having the product binaries on binary-host) from each WebSphere node (dmgr-host, was-host1 and was-host2) (all on one line):

```
mount –t nfs binary-host:/opt/WebSphere/AppServer
/opt/WebSphere/AppServer.
```

Refer to "http://nfs.sourceforge.net/nfs-howto" for more information on NFS mount.

Task 3: Create DMGR profile using the shared product binaries. Log on to dmgr-host and perform the following steps to create the deployment manager profile on dmgr-host using the product binaries on the shared file system of binary-host.

3.1. **Create the following directories** to store data for the deployment manager profile. Refer to chapter 7 for system and disk space requirements. On our Linux system we created a directory called /opt/wasprofile and mounted a separate file system on the /opt/wasprofile directory. According to the wasprofile.properties settings you modified earlier, you need to create three sub directories under /opt/wasprofile.

- profiles – to create dmgr profile under this directory.
- logs – to store log files under this directory.
- properties – to store property files under this directory.

<begin>
Creating a separate file system is recommended but not required. If you are configuring this for testing purposes simply create a directory /opt/wasprofile and the required subdirectories (profiles, logs, properties) under it.
<end>

3.2. **Copy the file ".WASRegistry"** from the binary-host system. The WebSphere install program creates a file named ".WASRegistry" in the home directory of the root user, which is the userid used to install WebSphere. This file contains only one line showing the location of WebSphere installation directory which is /opt/WebSphere/AppServer (<WASV6-ROOT>).

You need to create the same file (.WASRegistry) on each WebSphere node that will be running a WebSphere profile. The easiest way to do this is to create the file (.WASRegistry) on each WebSphere node under the required directory (~root) and add one line pointing to the installation directory (<WASV6-ROOT>). On our Linux system, we created the file using an editor (vi ~root/.WASRegistry) and inserted the line pointing to the WebSphere installation directory (/opt/WebSphere/AppServer).

3.3. **Run the profile creation wizard** (or wasprofile command) from the shared storage on the dmgr-host to create and verify a DMGR profile using the instructions given in chapter 7. You need to create the profile under /opt/wasprofile/profiles directory of dmgr-host instead of the default directory which is <WASV6-ROOT>/profiles on binary-host.

Task 4: Create Application server profile using the shared product binaries. Log on to was-host1 and perform the steps provided in Task 3 to create an application server profile on was-host1 using the product binaries on the shared file system of binary-host. Here you will be creating an application server profile instead of a deployment manager profile. Refer to Chapter 7 for instructions on creating and verifying the application server profile. You need to create the profile under /opt/wasprofile/profiles directory of was-host1 instead of the default directory which is <WASV6-ROOT>/profiles on binary-host.

Task 5: Create custom profile using the shared product binaries. Log on to was-host2 and perform the steps provided in Task 3 to create a custom profile on was-host2 using the product binaries on the shared file system of binary-host. Here you will be creating the custom profile instead of deployment manager profile. Refer to Chapter 7 for instructions on creating and verifying the custom profile. You need to create the profile under /opt/wasprofile/profiles directory of was-host2 instead of the default directory which is <WASV6-ROOT>/profiles on binary-host.

Task 6: Federate Nodes. Use the instructions provided in Chapter 8 to federate the profiles on was-host1 and was-host2 into the deployment manager cell on dmgr-host. Verify the configuration using the instructions given in Chapter 8.

Task 7: Apply product updates to WebSphere. You apply updates only to the product binaries on the shared file system on binary-host and all the profiles will be updated when the server processes are re-started. There are two ways to apply the update to WebSphere.

• **Simple:** Install the update onto the shared product binaries on the binary-host just as you would for a non-shared installation of WebSphere.
1. If possible stop all WebSphere proceses on all the nodes.
2. Log in to binary-host
3. Install the update.
4. Go to each of the WebSphere nodes and reboot the node and/or restart all the processes including the node agent.
5. If the update causes problems, then uninstall it from the binary-host (So the recommendation is to test the updated version before installing into a production environment to avoid having to uninstall it).

• **Robust (Hint):**

The following diagram shows the setup of the "robust" method of applying updates to WebSphere. In this scenario we are assuming that you are migrating from V6.0.2.5 (on binary-host1) to V6.0.2.9 (binary-host2).

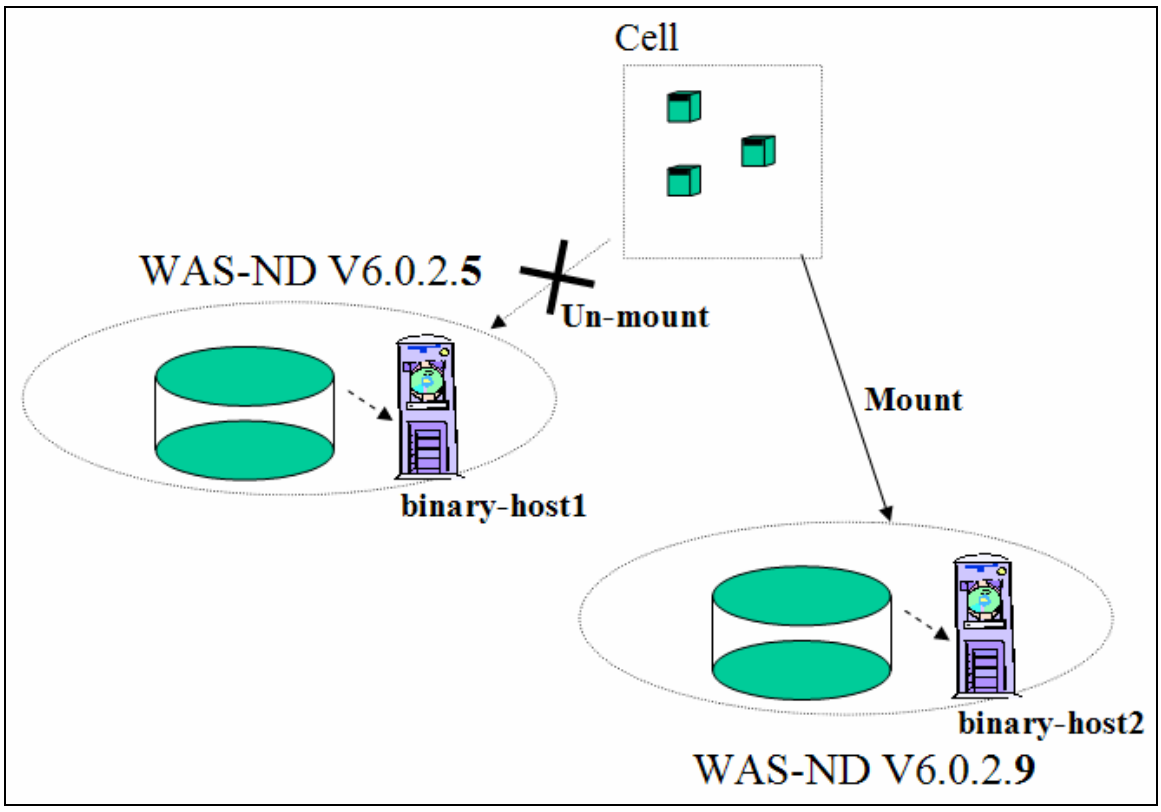

Figure 28-2: Robust way of applying product updates to WebSphere

1. Set up a file system on a second system (binary-host2). Use the same installation directory (<WASV6-ROOT>) structure used on the existing system (binary-host1).
2. Copy the contents of the present version of WebSphere on binary-host1 to the file system on binary-host2 (or re-install WebSphere product binaries on to this new system).
3. Apply required update to the product binaries on binary-host2.
4. Stop WebSphere processes on all the nodes before we un-mount the file system.
4. Un-mount the file system on binary-host1 from all the WebSphere nodes (dmgr-host, was-host1, was-host2) and mount the file system on binary-host2.
5. Reboot each WebSphere node and/or restart all the processes including the node agent.
6. If the testing uncovers a problem with the fix pack, then mount the original shared drive on binary-host1 and go through the server restart process to get back to the original state.

Task 7: Install IBM HTTP Server V6 and Plug-in. Use Chapter 5 to install and verify IBM HTTP Server V6 (IHS V6) and Plug-in (Use Chapter 6 and 9) on each HTTP server node (http-host1 and http-host2) the way you normally would. We do not recommend sharing IHS V6 binaries among HTTP server nodes at this time.

Workaround during local plug-in installation: When you install plug-in as local (deployment manager and HTTP server are on the same node), plug-in installation program will complain that it can not write on to the WebSphere installation directory (<WASV6-ROOT>) as it is read-only from the node you are installing the plug-in software. As a work around install plug-in as remote (even though it is local) and give the hostname of WebSphere application server as the local machine's hostname.

<begin note>
Even though you can share IBM HTTP Server V6 (IHS V6) binaries across HTTP server nodes just like WebShere binaries, it is more tedious and less straight forward than the sharing of WebSphere product binaries because there are more components (HTTP server, HTTP Administrative Server, Plug-in and WebSphere Application Server) involved in configuring each HTTP server instance. It gets more complicated and becomes a maintenance nightmare if you want to manage the HTTP server as an un-managed node from the deployment manager's admin console. We recommend installing and configuring IHS V6 (and IHS Administrative Server), and the plug-in on each HTTP server nodes as you do normally without sharing its binaries. We do not cover sharing IBM HTTP Server binaries in this chapter.
<end note>

Use Product Binaries across Cells

The diagram below provides a suggestion on how to create multiple cell environments using the same product binaries. Depending on the number of environments (development, test, integration, stress etc.), enterprise applications and version levels, you may want to share one WebSphere installation across all of the cells that are using the same version, or one installation per each environment, or one installation per set of applications, whichever is the best suited for your situation.
In any case it is recommended to exclude production for cells that are used for other stages of development. The screenshot below is illustrating that nodes on Cell-1, Cell-2 and Cell-3 are sharing the same product binaries (V6.0.2.5) on binary-host1, whereas nodes on Cell-4 and Cell-5 are sharing the product binaries (V6.0.2.9) on binary-host2. Consider this only as a suggestion and not a recommendation, and do it at your own risk.

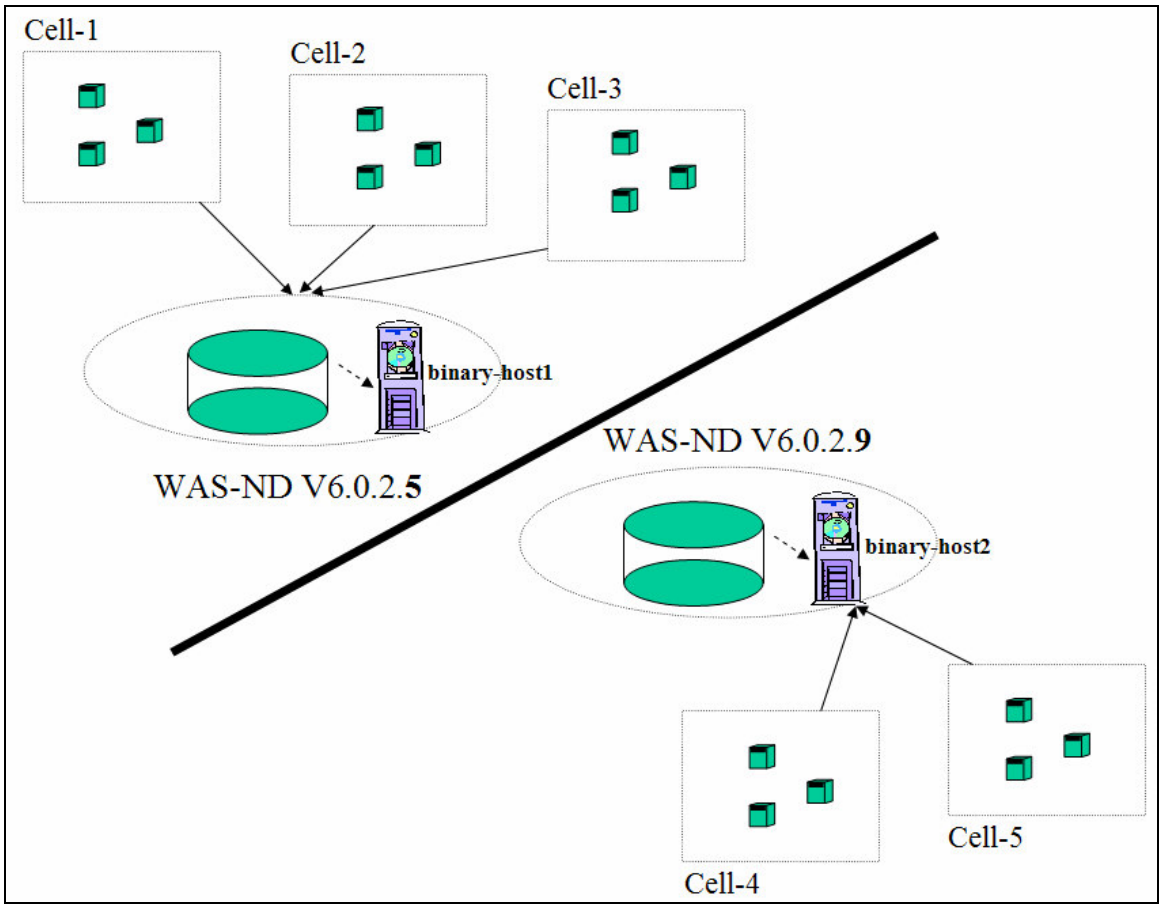

Figure 28-3: Sharing product binaries across cells

<begin note>

If you are using V6.0.2 and higher version of WebSphere and want to create and configure WebSphere Proxy Server as discussed in Chapter-27, then you have to manually edit an XML file as a workaround to use Proxy Server features under certain conditions. When you mounted the shared file system (where you have installed the product binaries) as read-only from each WebSphere node where you have created profiles, the augmentProxyServer.bat (sh) script fails (PARCIALSUCCESS) when you run the command against a deployment manager profile, even though it augments the profile successfully. In WAS 6.0.2, when a deployment manager profile needs to be augmented with the proxy templates, it also currently updates the feature.xml file (located in product binaries) to enable the proxy console plug-in. This is part of a workaround in 6.0.2.x to conditionally enable the Proxy console only when the templates have been updated. Open feature.xml file under <WASV6>-ROOT>/features/ com.ibm.ws.xd.nd_6.0.0.0 directory and add the content (highlighted text in the screenshot) to enable Proxy server entries on the admin console after augmenting the deployment manager profile.

```
    <plugin id="com.ibm.ws.console.proxy" version="6.0.0"
uri="com.ibm.ws.console.proxy/plugin.xml"/>
    <plugin id="com.ibm.ws.console.appmanagement.appedition
version="6.0.0"
```

Figure 28-5:
<end note>

Acknowledgements

This chapter is based on the document originally produced by **Steve Wehr**, zSeries New Technology Center. His document discussed the steps specific to zLinux on zSeries platform. I want to thank Steve Wehr for allowing me to use contents in his document that are common to the distributed environment. You can download the document produced by Steve Wehr by accessing the web site www.WebSphereMentor.com.