Presentation:
Moving Nodes Between MVS Images

Version Date: September 27, 2005
See "Document Change History" on page 39 for a description of the changes in this version of the document

IBM Washington Systems Center
IBM Americas Advanced Technical Support

Don Bagwell
IBM Washington Systems Center
301-240-3016
dbagwell@us.ibm.com
Overview

This presentation was originally presented at SHARE in Boston in August of 2005. The presentation is a shorter version of the WP100542 white paper found on Techdocs. That white paper is 96 pages long and very specific in its detail. This presentation takes a slightly higher-level view.

Note: The WP100542 white paper was written for WebSphere for z/OS Version 5, and this presentation is updated for Version 6. The differences aren't that great, but there are a few. Those differences are noted on page 3.

The act of moving a node from one MVS image to another can be viewed in two ways:

- **Temporarily** moving a node for a short period of time. In general, only the Deployment Manager node has this capability inherent in its design. This is covered on page 34 of this presentation, as well as more thoroughly in the WP100585 white paper on Techdocs.

- **Permanently** moving a node. Here the various system names and IP references are changed with the expectation that the node will be started on another MVS image and left there indefinitely. That’s more or less what this presentation is about.

While the work needed to update the configuration XML to move a node is not that great, the act of moving a node should not be treated as trivial. It involves going into XML and making changes, something we don’t generally recommend you do. There are complications based on the naming convention (see page 31). And moving a node to another MVS image may disrupt some automation you’ve developed.

Do not move a node casually ... do it with a careful and deliberate intent.
The Issue

WebSphere nodes have specific references to system names and IP host names. You can't *simply* restart servers on another MVS image.

Please don't misunderstand ... it *can* be done. It just requires some changes to the XML files. That's what this presentation is all about.

This presentation will be a mixture of key concepts and specific instructions.

The issue we're working with here is the fact that WebSphere V6 for z/OS nodes have configuration definitions that tie the node to a specific MVS image. You can't simply start the servers on another MVS image ... the result would be a failure of the server to start.

But that's not to say that it can't be done. It can. And it involves the modification of key XML files to reflect the IP host names and system names of the new node on which the node will operate. It's not really that complicated a process ... but it's also not an obvious process. So that's what this presentation is all about.
Based on WP100542

This presentation is based on the white paper WP100542, found out on the www.ibm.com/support/techdocs website.

To set the stage for this, let me point you to a white paper that was written for the V5 environment. This white paper provides a detailed step-by-step process for moving a node, and it provides several different scenarios.

Note: That paper was written for V5, not V6, which wasn’t available at that time. The process is very similar, but not exactly the same. The differences are highlighted below.

Here are the differences between the process outlined in that paper (the V5 process) and the V6 process:

- The introduction of "Node Groups" has moved a key variable regarding the Daemon IP host name from the cell-level variables.xml file down to a /nodegroups level nodegroups.xml file. You’ll see this touched upon in this paper on pages 18 and 23.
- The bbodc2ns.sh shell script is now gone, replaced with the wsc2n.sh shell script. Both accomplish essentially the same thing -- the “transformation” of the was.env files and the rebuilding of key properties files.
- The location of the shell script directory has moved. It used to be at /bin directory, immediately under either /DeploymentManager or /AppServer. In Version 6, the concept of a "profile" has been introduced. In z/OS, we only have one profile -- the default profile. Now, in V6, we must run the shell scripts under the /profiles/default/bin directory. Now what makes this interesting is that the higher-level /bin directory is still there, and the shell scripts are found in both locations. But we are to run the shell scripts down in the lower-level /profiles/default/bin directory. Why? Because that shell script sets up the command line interface, then it calls the other. We illustrate that on page 24.
Here's how this presentation is laid out. In the first section we'll set some essential concepts on the table so the process illustrated later will make some sense. Then we'll paint a picture of the process used to move a node by illustrating what is perhaps the most common movement scenario. And finally, we'll touch on the Deployment Manager, which is a special type of server and has more ability to be restarted on other MVS images than application servers do.
Essential Concepts
Config: HFS Directories and XML Files

Information about a WebSphere configuration is kept in a deep directory structure in the HFS. It contains many directories and XML files:

Moving a node to another MVS image involves updating these XML files to reflect the new system name and IP address.

These references are found in a relatively small number of XML files.

Just one configuration structure?

The first point we'll make is that the configuration structure for WebSphere is an HFS directory structure that goes down a fair ways. Along the way, the cell's long names -- cell, nodes and servers -- will be used as directory names. Interspersed along the way will be XML files that contain the information about the configuration.

Note: See the /profiles/default portion of that structure? That's new with V6. Everything else is pretty much the same as it was in V5.

To move a node to another MVS image involves modifying some of these XML files and updating the information contained inside. The system name and IP host name references are contained in a relatively small number of the files.

To those of you who are somewhat familiar with the WebSphere configuration structure, you may ask, "Just one directory?" No, in fact there are multiple.
Every Node Has a Config Structure

A WebSphere "ND" configuration -- Network Deployment -- consists of multiple nodes. Each node has its own configuration structure:

```
<mount_point>
  /DeploymentManager
    /profiles
      /default
</mount_point>
```

This is the key

```
<mount_point>
  /AppServer
    /profiles
      /default
</mount_point>
```

All changes made to the Deployment Manager's configuration. The "master configuration" resides there

Let's look at the Deployment Manager's configuration ...

In a WebSphere "Network Deployment" configuration, there will be multiple nodes. Each node has its own configuration structure. The Deployment Manager's configuration will, by default, be under the `/DeploymentManager` directory, and the application server nodes will be under, again by default, the `/AppServer` directory.

**Note:** Set aside the issue of having the nodes in separate HFS or one big HFS. That subject is largely irrelevant to the question of moving nodes.

The key to this whole thing is the Deployment Manager node directory structure. That contains what's known as the "Master Configuration," which is the DMGR's configuration as well as a copy of the configuration for every other node in the cell. This is where we'll make all of our changes. We will not climb into the application server node directories to update XML there. We'll use the `syncNode.sh` process for that.

**Note:** We'll talk about the `syncNode.sh` process later in this presentation.
The DMGR’s Configuration Structure

The Deployment Manager’s configuration structure contains information about the DMGR and every node in the cell ...

If the cell had six application server nodes, the DMGR’s master configuration would have a section for each.

We’ll make changes to a node’s XML in the DMGR’s master configuration, not the node configuration

These changes are copied out to the node XML with syncNode.sh ...

The Deployment Manager configuration has a section for each node that’s found in the cell. You can see evidence of this under the /nodes directory. Under that you’ll find a directory for each node, including the DMGR’s node. So if your ND cell had six application server nodes, you’d see seven directories under /nodes -- one for the DMGR and one for each application server nodes. The name of the directories under /nodes would the node long name for each node.

This is what’s known as the "Master Configuration" for the cell. It’s in here that we'll make our configuration changes. Changes made in the "Master Configuration" find their way out to the node directory structures by way of a process called syncNode.sh, which we'll run from each application server node directory.
Changes "Synched Out" to Nodes

We'll make all our changes in the Deployment Manager's master configuration, then we'll use the `syncNode.sh` utility to synchronize changes out to the nodes:

![Diagram showing system components and configuration structure]

The `syncNode.sh` shell script performs what is in essence the same thing as node synchronization performed through the Admin Console. That shell script will coordinate the synchronization of changes found in the DMGR's "Master Configuration" with the same files in the node's configuration.

At a very high level, the process goes something like this:

- The `syncNode.sh` shell script is invoked. It makes contact with the DMGR and asks, "Do you have changes for me?"

  **Note:** The `syncNode.sh` shell script takes as parameters the host and port on which the DMGR is listening. This is how the shell script makes contact with the DMGR to ask this fundamental first question.

- The Deployment Manager checks its master configuration and compiles a list of files that have changed since the last synchronization effort was made with the node. It passes that list to the `syncNode.sh` process.

- The shell script then issues a series of HTTP "GET" commands to pull the changed files down from the DMGR.

- As the `syncNode.sh` process receives each file, it then turns and updates the node's configuration with the file brought down from the master configuration. It does this for every file, thus updating the node's configuration to match the files in the master configuration.

How does it know which node to work against? Each node has a copy of the `syncNode.sh` shell script. Which node gets updated depends entirely on which copy of the `syncNode.sh` shell script you invoke. The shell script will only update the node in which the shell script itself resides.
Three "Levels" of the Config HFS

When referring to the various XML files, it's helpful to understand the three "levels" of the configuration structure: cell, node and server.

Cell Level

/tools/deployment
/profiles
/default
/config
/cells
/<cell_long>

Node Level

/nodes
/<node_long>

Server Level

/servers
/<server_long>

To move a node to another MVS image will require changes to files in the node and server level.

It may require changes to a cell level file. More on this later.

Some files exist by the same name in multiple locations, so we'll refer to the file by name and location.

Example: "variables.xml at the node level"

When considering the HFS structure, it's important to understand that there are three "levels" of the configuration structure:

- **Cell-level** -- this is immediately under the /cells/<cell_long> directory. Here you'll find a series of XML files that relate to the cell as a whole. There's one file here we may need to change.

- **Node-level** -- this is immediately under /nodes/<node_long> directory. Here you'll find more XML files, these related to the node itself. There'll be one "node-level" directory structure for every node in the cell. We'll be changing two files in here for each node.

- **Server-level** -- immediately under /servers/<server_long>. Here more XML files will be found. There'll be one of these directories for each server in each node. We'll change one file in here for each server being moved.

The reason why we bring this up is because some XML file names exist in multiple locations. The variables.xml file, for instance, exists at all three levels. So we'll reference the files by name and their level.
Information Kept in ASCII XML Files

The XML files are encoded in ASCII format, so editing with the ISPF editor will be a challenge:

```
EDIT       serverindex.xml                                 Columns 00001 00072
Command ===>                                                  Scroll ==== CSR
****** **************************** Top of Data ****************************
==MSG> -CAUTION- Data contains invalid (non-display) characters. Use command
==MSG> ===> FIND P’.’ to position cursor to these
000001   Ã_% Œ¡        ¡>ƒ?¿—>  ≈       À¡       À¡        Æ¡         Ò>¿¡Ã Ρ   Œ¡  
****** **************************** Bottom of Data ****************************
```

There are tools to help edit ASCII files on the mainframe -- viascii, utilities that convert the file to EBCDIC for editing, etc.

You can download files to PC and edit there, then upload back to z/OS

Be careful with the editor you use. Some can change the line break characters and create problems. See notes below for explanation of possible error symptom.

Key message here is that the files are in ASCII so be aware.

The XML files in the WebSphere configuration are maintained in the HFS as ASCII files. What this means is that you have to use the proper tool to edit them. You can't go in using the normal ISPF editor (by default) because it'll assume an EBCDIC code page. You'll end up with something that looks like the chart.

But there are lots of ways you can edit ASCII files up on z/OS. The "viascii" tool is one. There are also tools to quickly and automatically convert ASCII files to EBCDIC before opening them in the ISPF editor. I used the old trick of downloading the file to my PC where I edited them with natively as an ASCII file. I used the WS-FTP tool to make this relatively quick and easily.

**Note:** You should be careful with the editor you use on the PC. Some will put a CR-LF at the end of files, and that can cause problems when the file is uploaded back up to the mainframe. WordPad caused me some problems. NotePad is difficult to use because it wraps all the lines into one big bundle (it doesn't recognize the LF with the CR).
Key Assumptions

- **Target MVS image in the same Sysplex**
  
  V6 supports cells that span Sysplex, but that introduces complexity not covered in this presentation.

- **All WebSphere names stay the same (server, node)**
  
  Some names critically tied to underlying security profiles; changing would imply considerable effort.

- **Same mount point name used on target MVS**
  
  Maintained as a WebSphere variable and used as an important starting point for configuration references.
  
  WP100542 has section on how to change this value.

- **Servers on target MVS have access to JCL and product code**
  
  Code libraries and code HFS, and if no STEPLIB then modules in LPA/LNKLST.

- **Servers on target MVS have access to updated configuration HFS**
  
  Shared HFS -- should change ownership of HFS to target MVS.
  
  Unshare HFS -- move HFS to target MVS.

- **TCP port values stay the same**
  
  Remap those if you wish after you get the node up on new MVS image.

  With enough research and effort, it’s possible to change any aspect of the configuration by manipulating the XML. But we have to draw a line somewhere to keep the topic from being too complicated.

So now we lay out the key assumptions under which the rest of this presentation will operate:

- **Target MVS image in the same Sysplex.** The reason for this is because WebSphere restricts a "node group" to a given Sysplex. It's possible to have multiple node groups, but moving a node from one node group to another is beyond this presentation.

- **All WebSphere names stay the same.** This is explained on page 31. The basic problem here is that names are not just WebSphere things, but are tied to other things at the MVS level. Changing them is a bit more complicated. So we'll assume these don't change.

- **Same mount point used on both MVS images.** It would be possible to unmount a node's HFS on one MVS image and mount it on a differently-named HFS on the other MVS image. But things wouldn't work. Not only would the JCL be wrong (the SET_ROOT= value would be incorrect), but the WebSphere runtime would also not work because it maintains the mount point as a variable and uses that as the starting point for the searches for the XML files. Therefore, the assumption is that the mount point, once set initially, doesn't change.

- **Access to JCL and product code.** If your JCL is in a system-specific proclib that is accessible from one MVS image but not another, then you have to make sure the JCL is made available to the new MVS image to which the node is moved. The same applies to the product code modules. If you have the modules in LPA/LNKLST on SYSA, then you'll have to load them into SYSB as well.

- **Access to updated configuration HFS.** The configuration HFS is not necessarily a shared HFS, so updates made to an HFS that's owned by SYSA (and not shared) will not be accessible from SYSB. So the assumption here is that the configuration HFS will be available. But even if the
configuration HFS is shared, you should change the ownership to the target MVS to help performance.

- **TCP Ports the same.** This is one that could actually be accomplished fairly easily during our other changes to the XML files. The port values are contained in the `serverindex.xml` file, and we could change them while we're in editing that file for other reasons. But it would just cloud our issue, so we'll suggest you postpone remapping ports until after the node has been moved.
Preaching to the Choir

Any time you change the XML files you run the risk of breaking something.

Back Up Your HFS!

PAX
DFDSS
etc.

Also, be careful ... think about what you're doing and go slow at first.

We'll manually change a few files, but those changes will be propagated to others. So backing up the whole HFS is far better than trying to backup individual files.

Finally, since we'll be in making changes to the configuration XML files, it is strongly recommended that you back up the HFS.
Common Scenario
Our Sample Scenario

To illustrate this process, we'll use the following hypothetical scenario:

- Network Deployment configuration
- Two nodes: DMGR and AppServer
- Both nodes move from SYSC to SYSD
- Daemon instance moved as well

This scenario is useful because it illustrates some key points:

- Two nodes updated in DMGR's "Master Configuration"
- Three "servers" updated: DMGR, Node Agent and AppServer
- The regeneration of DMGR was.env with wsc2n.sh shell script
- The automatic regeneration of the Daemon instance
- The use of syncNode.sh to synchronize changes for node

Let's see what the process looks like ...

To help illustrate this process, we'll use a hypothetical configuration scenario as shown in this picture. Here we have a ND configuration with two nodes -- a Deployment Manager node and a simple application server node. We'll move both nodes from SYSC to SYSC.

This configuration is useful because it'll illustrate several key points:

- It'll allow us to show how the master configuration has different directories for each node.
- It'll allow us to show how when multiple servers exist in a node, that it'll require driving down into the server-level directories of each server.
- Since this configuration has two nodes, it'll show how we regenerate the was.env files for the DMGR and Daemon with the wsc2n.sh file, then update the application server nodes using the syncNode.sh shell script.

This model is the same as would be the case for a much larger configuration. So this scenario is helpful in illustrating the process.
High Level of Process Used

At a very high level, here's the process:

- Stop the servers
- Modify system name variables and IP host name variables for DMGR node
- Run `wsc2n.sh` to regenerate DMGR `was.env` and regenerate Daemon instance
- Start DMGR on SYSD
- Modify system name variables and IP host name variables for appserver node
- Run `syncNode.sh` from appserver node -- master config changes copied out and all `was.env` files in the node are updated
- Start application server node servers on SYSD

And here's what we'll do:

- First, we'll stop the servers. Technically speaking this is not required ... it's possible to update the XML while the servers are running. But there's always a risk some files might be updated after we've gone in and made some changes, so it's best to shut the servers down to make sure no changes are made.
- We'll modify the settings in the XML files for the Deployment Manager.
- We'll then run the `wsc2n.sh` shell script to regenerate the properties files for the Deployment Manager. We'll then start the Deployment Manager on the new MVS image.
- Next we'll modify the XML files for the application server node. We'll make those changes in the DMGR's "master configuration."
- Then run `syncNode.sh` from that node. That'll synchronize the changes in the master configuration out to the node configuration structure, and it'll update the `was.env` files for all the servers in the node.
- Lastly, we'll start the servers in the application server node on the new MVS image.
### Summary of Where Stuff is Referenced

Here's a quick reference to where the information is located:

/\*<mount_point>\*/
  /\*DeploymentManager\*/
  /\*profiles\*/
    /\*default\*/
    /\*config\*/
    /\*cells\*/
      /\*<cell_long>\*/
        /\*virtualhosts.xml\*/
        /\*nodegroups\*/
          /\*DefaultNodeGroup\*/
            /\*nodegroup.xml\*/
        /\*nodes\*/
      /\*<node_long>\*/
        /\*variables.xml\*/
        /\*serverindex.xml\*/
      /\*servers\*/
        /\*<server_long>\*/
          /\*server.xml\*/

#### File Name | Location | Description
--- | --- | ---
virtualhosts.xml | Cell Level | Contains references to IP host names. You may or may not need to modify this file, depending on whether you have aliases that are exclusive to a specific host. See page 32 for more.
nodegroup.xml | Node Group Level | Has an IP host name that's given out to external clients seeking Location Name Services from the cell. This will be the IP host name listened on by the Daemons. Ideally this would be a Sysplex Distributor generic IP host name. But if it's a system specific host name, and you've moved the Daemon off that host, you'll need to change this value.
variables.xml | Node Level | Here several references to the system name will be found. There will be one variables.xml per node.
serverindex.xml | Node Level | This file has IP host name references and the High Availability Manager (HAM) host reference. One per node.
serverserver.xml | Server Level | Has a single reference to the system name as well as the potential (but unlikely) reference to an IP host address in the event the HTTP ports are bound to a single adapter.
variables.xml for the DMGR Node

This is where several references to the system name are kept:

```
"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSC"
"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSC"
"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSC"
"WAS_DAEMON_daemonInstanceName" value="SYSC"
```

Notes:
- Lines not contiguous like shown here ... other lines separate them
- Do global "find / replace" and change from old system name to new
- Save the file

Let's start this discussion by focusing on the variables.xml file, which is located at the Node Level for the node. In our hypothetical example we have two nodes, the DMGR and the Application Server node. We'll start first with the Deployment Manager.

Down under the /DeploymentManager directory, down under /profiles/default and under /nodes you'll find a directory for each node in the cell. In this example the DMGR's node is called /g6node. It's in that directory where we'll find the variables.xml file (and the serverindex.xml file as well).

If you edit this file and do a global search for all the instance of the system name, you'll find it has four. If you're permanently moving this node to another MVS image, then you should change all four instances of the system name in the file.

Note: The instances of the reference are not contiguous as shown here. They're a bit more scattered.
**serverindex.xml for the DMGR Node**

This is located in the same directory as the variables.xml file for the DMGR node. It contains IP information about the servers in the node.

Later we'll talk about using Sysplex Distributor IP host for these values

**Notes:**
- The IP host names in the DMGR's serverindex.xml file should, ideally, be set to a Sysplex Distributor generic IP host name. The DMGR is a special server that is designed to be started on different MVS images. That's not possible if the IP host names are tied to a specific MVS image. But a Sysplex Distributor generic IP host name is not tied to a particular MVS image, so having that value coded provides maximum mobility of the DMGR. See page 34 for more on this.
- You'll notice the TCP ports. This is where you could change them if you wanted to. But that's outside the scope of this presentation, so we'll assume you're not changing them.

The serverindex.xml file is located in the same directory as the variables.xml for the node. This is a file that requires a bit more explanation as to what to do. What this file does is provide the "end point" information for all the servers in the node. For the DMGR node there is only one server -- the DMGR. But that one server has multiple ports, and this file defines the IP host address and TCP port for each port.

This file also provides an IP Host name that's related to the node as a whole. That reference is located on one of the first lines, and in an un-wrapped file exists way off to the right side of the file. That particular reference can't be updated through the Admin Console, but all the others can. But this one instance is important, so we can't just leave it set to the old IP host name. Hence our going into the XML files.

The IP host names in the DMGR's serverindex.xml file should, ideally, be set to a Sysplex Distributor generic IP host name. The DMGR is a special server that is designed to be started on different MVS images. That's not possible if the IP host names are tied to a specific MVS image. But a Sysplex Distributor generic IP host name is not tied to a particular MVS image, so having that value coded provides maximum mobility of the DMGR. See page 34 for more on this.

You'll notice the TCP ports. This is where you could change them if you wanted to. But that's outside the scope of this presentation, so we'll assume you're not changing them.

The DCS_Unicast address is used to hold the port for the "High Availability Manager" (HAM) port. There's been some maintenance updates to this function recently that we need to tell you about. That's next.
The High Availability Manager Reference

When Version 6 was first released, that port required being bound to an IP host that was single-homed. That restriction was relaxed with 6.0.1.2

<table>
<thead>
<tr>
<th>Node Built</th>
<th>Node Built</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre 6.0.1.2</td>
<td>6.0.1.2 or later</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&lt;specialEndpoints xmi:id=&quot;NamedEndPoint_6&quot; endPointName=&quot;DCS_UNICAST_ADDRESS&quot;&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;endPoint xmi:id=&quot;EndPoint_6&quot; host=&quot;9.82.24.71&quot; port=&quot;26513&quot;/&gt;</td>
</tr>
<tr>
<td>&lt;specialEndpoints xmi:id=&quot;NamedEndPoint_6&quot; endPointName=&quot;DCS_UNICAST_ADDRESS&quot;&gt;</td>
</tr>
<tr>
<td>&lt;endPoint xmi:id=&quot;EndPoint_6&quot; host=&quot;*&quot; port=&quot;26513&quot;/&gt;</td>
</tr>
</tbody>
</table>

You may see an IP address in that field or an IP host name:

- If still pre-6.0.1.2, then update with IP address of target MVS or IP host name of target MVS if single-homed address
- If WebSphere now 6.0.1.2 or later, then you may replace this value with an asterisk

ISPF customization panels changed so no host value needed for HAM port

- You'll see an asterisk (as shown above)
- No changes necessary -- leave as an asterisk

Next, the `server.xml` for the DMGR ...

There has been a change in the behavior of the "host" value associated with the DCS_Unicast port since Version 6 first came out. Prior to 6.0.1.2, the host value on that port was required to be specified, and the host name provided had to resolve to a single IP address. In other words, "multi-homed" host names weren't permitted. This is why you might see examples showing an IP address in this field -- that address was for a specific adapter. After 6.0.1.2, the "single-homed" restriction was lifted, and now the HAM function is able to discern where it's located. So now an asterisk can be used in that field.

**Note:** After the applying of service level 6.0.1.2, the ISPF customization panels don't even show the "host" field anymore. It still asks for the port, but not the host. Now an asterisk is automatically inserted.

When you open up the `serverindex.xml` file, what you find inside will depend largely on when the node was built. If it was built prior to 6.0.1.2, then what you'll see is likely a single-homed IP host name or an actual IP host address for the DCS_Unicast host. If the node was built after 6.0.1.2, you'll see only an asterisk in that spot.

So what you do is based on what you find in the file and what service level you're at right now:

- If you're not yet to V6.0.1.2, then you still have to have a single-homed address there. If you're moving the node to another MVS image, then you'll need to change the host value for the DCS_Unicast port to a valid single-homed address over on the target MVS image.
- If you're at 6.0.1.2, then you're free to set this to an asterisk.
server.xml for the DMGR Server

There are two things in here -- a system name reference and a "host" value on the HTTP ports you may or may not need to change:

```
/server.xml
```

### System name reference:

```
<configuration>
  <name>was.ConfiguredSystemName</name>
  <value>SYSC</value>
</configuration>
```

One instance, about 100 lines down. Change this to your target MVS image system name.

DMGR is a special server in that server will start even if this value is incorrect. But if move is "permanent," then change. More later.

### Possible IP address value in XML file

```
<address port="26514" host=""/>
<address port="26515" host=""/>
```

If the `host` field is null, then the HTTP port is not bound to a specific adapter. Do nothing.

But if the `host` field has an IP address, then either specify an appropriate IP address on the target MVS, or turn the field into "null"

Next, "node groups" ...

The `server.xml` file resides down in the lowest-level directory for the Deployment Manager node. It sits in the same directory as the `was.env` file and the `jvm.properties` files for the server. This file has a single reference to the system name on which the server will run. Change that value to the new MVS system name.

**Note:** See page 34 for more on the special nature of the DMGR and how it can be started on another MVS image even without this value being updated. If your move is "permanent" then you should change it. But be aware that for temporary purposes the DMGR can be started on another MVS image without this value being updated. Other planning is necessary.

There are two fields in this file that may -- but unlikely -- need updating. The HTTP ports for the DMGR are maintained in this file, and those ports may have a host address associated with them. This would only be the case if the HTTP port were being bound to a specific adapter. Otherwise, the fields will be null, implying the ports are listening on all adapters. If you find an IP address in those fields, then you have two choices for the target MVS image:

- You can change the host IP address to one that's found on the target MVS image, or
- You can clear the field, make it null, and allow the ports to listen on all adapters on the target image
With V6, WebSphere has introduced the notion of "node groups." By default there'll be one -- DefaultNodeGroup

A few notes about Node Groups:

- By default all nodes added to a cell will be members of the DefaultNodeGroup, unless you intentionally create another
- Clusters may be built within a node group, but not between
- Daemon server settings are held in the nodegroup.xml

There's one more file we need to update to make the DMGR come up okay. It's something new with V6, and it's called the nodegroup.xml file. Before V6 we had no notion of a "node groups" -- a cell had to stay within a cell, and all the nodes were part of the same group. But with V6 the cell can now jump Sysplexes or even jump to non-z/OS servers. So the idea of a "node group" was invented to be able to group up nodes that were roughly similar to one another.

Note: The main purpose of this is to provide a boundary in which clusters can be formed. Clusters can be formed within a node group, but not across node groups.

Though multiple node groups are permitted, your cell will probably be operating with one node group -- the default node group called DefaultNodeGroup. That was the case for the sample scenario used in this presentation. All the nodes were members of that single node group, so that meant only one nodegroup.xml file to update.

The file is located under /profiles/default/nodegroups/DefaultNodeGroup. In that file is an IP host address that defines the host on which the Daemons of the node group listen. The purpose of this property is so when an external client requests information on how to contact a Daemon to exercise the "location name service" function of WebSphere, this IP host value can be given out.

It is a fact that an external client may access any Daemon in the cell. So this value need not be specific to an MVS image. The alternative is to have this defined as a Sysplex Distributor generic IP host value, and VipaDistribute the Daemon ports. If that's what you've done, then make no changes to this file. But if you find a system-specific IP host value here, and you're moving the Daemon away from that host, then change this value to be equal to the host value of the target MVS image.
Run \texttt{wsc2n.sh} for DMGR

With the DMGR's XML changed, it's necessary to run \texttt{wsc2n.sh} to update all the settings for the node to reflect those changes.

This shell script does many things:
- Reads all the XML, including your changes
- Regenerates the DMGR's \texttt{was.env} file
- Regenerates the JVM properties files
- Regenerates other properties files
- Updates Daemon (next chart)

Key Points
- "Switch User" to your WebSphere Admin ID to run this shell script. Do \textbf{not} run as UID=0.
  Will result in ownership and permission problems. If you accidentally do run \texttt{wsc2n.sh} under UID=0, then re-run BBODHFSB to "fix" HFS.
- Run copy under /\texttt{profiles}/default/bin, \textbf{not the copy found under} /\texttt{bin}
  The one under /\texttt{profiles}/default/bin performs critical command line setup things and then calls other one.

BBOJ0056I Transformer Processing Complete, RC=0.

What about the Daemon server? ...

With all the XML files updated, it's now time to run the \texttt{wsc2n.sh} shell script. That file will read the updated XML files and propagate that information out to the other files in the node that aren't XML files -- \texttt{was.env}, \texttt{control.jvm.properties}, etc. Running this utility is a critical part of updating the DMGR.

There are two copies of the shell script in the DMGR's directory structure: one under the /\texttt{bin} directory, and one further down under /\texttt{profiles}/default/bin. Run the one lower in the directory. That one sets up the command line environment then calls the other one.

Before invoking this shell script, switch users to the WebSphere Admin ID. Do \textbf{not} run this under a UID=0 ID. That will result in several key files being owned by that UID=0 ID, which may make subsequent operations troublesome. If you find that has happened, go back to your DMGR's CNTL data set (the one you allocated when creating the customized jobs that created the DGMR) and re-run the BBODHFSB job. That'll sweep through and update the DMGR's config ownership back to the Admin ID.

Look for the BBOJ0056I message as an indication of success.
Daemon Instance Updated

In addition to transforming the DMGR node, the wsc2n.sh shell script will determine which Daemon instance supports it and then creates a new Daemon instance based on the updates to the DMGR XML:

![Diagram showing the structure of the directory for Daemon instances](image)

Interesting points:
- Daemons have no XML files themselves. The only configuration file is was.env
- You should never hand-modify a Daemon's was.env because your changes may be overwritten by other WebSphere processes
- The wsc2n.sh shell script will actually make another Daemon instance -- with the new system name -- and leaves the old instance in place

What about the Daemon server for the Deployment Manager? Well, do you remember changing those system references in the variables.xml file at the node level? In so doing that, it provided the information needed for wsc2n.sh to create a new Daemon instance. The new Daemon instance will be written into the /Daemon directory, under the structure illustrated in the chart. The only configuration file a Daemon has is the was.env file -- there are no XML files associated with a Daemon.

So with the new Daemon instance created, you're ready to start the DMGR on the target MVS image. How that's done is next.
Start DMGR on New System

Your Deployment Manager is capable of being started on the new MVS image. *Your AppServer node is not yet capable of it.*

```
S  G6DCR,ENV=G6CELL.G6NODE.G6DMGR,JOBNAME=G6DMGR
```

**Assumptions:**

- JCL start procedure available to new system
- HFS mount point has not changed
- Configuration HFS available to new system

Having DMGR running is necessary pre-req for next step ...

With the changes made to the Deployment Manager's XML, you can now start that server. You can use the exact same START command as you did before. Doing so, however, implies some assumptions:

- You're using the same JCL start procedure.
- That has the `SET ROOT=` value that points to the mount point for the configuration HFS. We stated earlier that a key assumption was that the mount point would be the same value. The reason for that assumption is not only so the JCL doesn't need to be change (after all, that change would be easy), but more importantly because WebSphere maintains the mount point as a variable inside its configuration and uses that variable as a starting point for all references to underlying XML files.
- The configuration HFS is available to the target MVS image. If the HFS was shared across MVS images, this would be automatically possible. But if you don't have shared HFS, then you'll have to mount the HFS over on the target MVS image. Even if the HFS was shared, we'd still recommend you at least change the ownership to the target system so the performance would be better.

We're about to show how to change the configuration information for the application server node. Recall that we said that would be done in the DMGR's "master configuration" as well. Therefore, to get those changes "synched out" to the node, the DMGR would have to be running. So with the DMGR now running (but on a new system), we're ready to make the changes to the application server node and then invoke `syncNode.sh` to get those changes out to the node configuration.
Make Changes to AppServer Node

Now update your application server node in the Master Configuration. The files and the updates are largely the same as with the DMGR:

```
<mount_point>
  /DeploymentManager
  /profiles
  /default
  /config
  /cells
  /<cell_long>
  /nodes
  /<dmgr_node>
  /<appserver_node>
   serverindex.xml
   variables.xml
   servers
   /<server_1>
   /<server_2>
   /nodeagent
  /<appserver_node>
```

A few notes about the serverindex.xml file for multi-server node ...

The process is essentially the same as for the Deployment Manager. What you’d do is locate the node’s directory in the DMGR’s master configuration, then modify the variables.xml file, the serverindex.xml file and all the server.xml files, one for each server in the node.

**Note:** Provided the application server node is in the same "node group" as the Deployment Manager, you won’t have to modify the nodegroup.xml file ... it was already done. If per chance you have multiple node groups, then locate the node group directory that this node belongs to and modify that copy of nodegroup.xml.

Here’s a quick recap of the files:

**variables.xml at the node level** -- there’ll be several references to the system name in this file, just like there was for the Deployment Manager. Change those references to the system name for the target MVS image and save the file.

**serverindex.xml at the node level** -- here there’ll be the IP host name references. The primary difference between this copy of the file and the one for the DMGR node is that this will have multiple "sections," one for each server in the node. The DMGR node had only one server. We’ll explore the structure of the serverindex.xml file on the next chart.

**server.xml at the server level** -- you’ll find a directory for each server in the node, including the Node Agent. This file will be found in each, and each will need updating to change the one instance of the system name reference in the file (and possibly, though not likely, the HTTP ports and the host="" reference made there, just like the DMGR had).

Let’s explore this copy of the serverindex.xml file a bit ...
Multi-Server Node serverindex.xml

The Deployment Manager's serverindex.xml had only one "server entry" -- that's because that node had only one server. But a node with multiple servers will have a block of XML for each server, including Node Agent:

```
<?xml version="1.0" encoding="UTF-8"?>
<serverindex>
  <serverEntries serverDisplayNames="nodeagent">
    <End_point/>
  </serverEntries>
  <serverEntries serverDisplayNames="server_1">
    <End_point/>
  </serverEntries>
  <serverEntries serverDisplayNames="server_n">
    <End_point/>
  </serverEntries>
</serverindex:ServerIndex>
```

Notes:
- All the IP host name references in this file would need to be changed
  Performing a "global change" might be best way.
- Node Agent "DCS_Unicast" end point (the HAM) might be specific IP address
  From pre-6.0.1.2 construction. Same rules apply as mentioned a few charts back about High Avail. Mgr.
  Application servers have DCS_Unicast end point as well, but those will always be asterisk
- Node Agent IP host names may be Sysplex Distributor generic host name
  Be aware and be careful. If they are Sysplex Distributor generic host name, then leave as is.

You're now ready to "sync" the changes out to the node ...

The serverindex.xml file for an application server node will look a bit different than that for the Deployment Manager node. The difference lies in the way the file is structured: one <ServerEntries> block per server in the node. Each block will contain an IP host name reference, so be sure to update all the references in the whole file.

The Node Agent will have a DCS_Unicast port which servers at the HAM for that node. The same issues we outlined back on page 21 applies here as well.

Finally, Node Agents are like Daemons in that one recommended practice is have Sysplex Distributor distribute traffic between all the Node Agents in a cell. If that's how yours has been configured, then the IP host names may be Sysplex Distributor generic IP host names. If that's the case, then be careful updating the Node Agent section of the file ... leave generic IP host name references as is. Change only the system-specific host name references to match the IP host name of the target MVS image.

Now you're ready to synchronize the changes out to the node.
Run **syncNode.sh** from Node

The shell script `syncNode.sh` will pull changes from the Deployment Manager and update the entire node with those changes:

Each application server node has a shell script in its `/profiles/default/bin` directory called `syncNode.sh`. This shell script will invoke a node synchronization process with the Deployment Manager and pull all the changes made in the DMGR's "Master Configuration" out to the node itself. In addition, the `syncNode.sh` shell script will sweep through the node, updating all the `was.env` files and other properties files and prepare the servers to start on the new system.

Here are the keys to running this shell script:

- Make sure you're invoking the shell script from the proper node. The shell script will figure out which node is being synchronized based on which copy of the `syncNode.sh` shell script is invoked. If you're accidentally in another node's directory structure and invoke that node's copy of `syncNode.sh`, it'll synchronize that node. So be aware of which copy of `syncNode.sh` you invoke.

- Before invoking the shell script, switch users to the WebSphere Admin ID. The shell script process is going to update a lot of files, and if you're running under a different ID, either the process will fail (insufficient authority) or, if you're running under a UID=0 ID, it'll create files with ownership that'll prove troublesome later. (You can always run the `BBOWHFSB` customized job to sweep through the node and set all the ownerships back to the WebSphere Admin ID if you need to.)

  **Note:** If the DMGR has global security enabled, then running `syncNode.sh` under the Admin ID is very important -- that ID by default has the "CA certificate" in its keyring, which is what allows an SSL connection to be established. Some other UID=0 ID may not have that CA certificate in its ring.

- Finally, the syntax of the command is as shown in the chart above. The key is that the first parameter is the host name where the DMGR is running, and the second parameter is the SOAP port on which the DMGR is listening. There are other parameters, but those two are required. If global security is enabled for your cell, then you'll need to pass in the WebSphere Admin ID and password to get in the front door of the SOAP port. Look for the message as shown in the chart for an indication of success.
Start Servers in Node

Your application server node servers may now be started on the new MVS image.

S G6ACR, ENV=G6CELL.G6NODEC.G6AGNTC, JOBNAME=G6AGNTC

Assumptions:
- JCL start procedure available to new system
- HFS mount point has not changed
- Configuration HFS available to new system

What about the system identifier portion of the naming convention?

Now you can start the servers, just as you did the Deployment Manager. The same assumptions apply. See page 26 for more.
A Known Limitation -- Names

If your naming convention includes a "System Identifier," then moving the node may create some confusion. But changing names not straight-forward.

WebSphere "long" names
- Are not open to change in Admin Console
- Used as directory names in configuration structure
- Are used throughout the WebSphere configuration to identify locations and to relate one configuration object to another

Unless there was a proven way to change all instances of a long name, you should be very careful making ad hoc changes to the configuration

WebSphere "short" names
- May be changed in Admin Console
- Found in the symbolic link structure used by ENV= on MVS START
- Are related to underlying security profiles!
  - Server short -- tied to servant STARTED profile; possibly controller STARTED if using qualified
  - Cluster transition -- tied to CBIND and SERVER profiles

Understand interrelationship and be careful

Final note on virtual host aliases ...

Earlier we mentioned that a key assumption in this process is that the node and server names will not change when they are moved to another MVS image. That may be troubling to some, as their naming convention may have included a "system identifier". That system identifier is now invalid on the new MVS image. As we look at this issue, it's useful to remember there are two kinds of names: long and short.

Long Names -- these are used by WebSphere as the display names on the Admin Console and in the directory structure of the HFS. They are not open to change by the Admin Console, which means they are generally not considered something to be changed. I've not explored exactly how the long names might be changed. The key point here is that unless a proven method is outlined, it is inadvisable to try changing long names on an ad hoc basis.

Short Names -- these are z/OS-only constructs meant to overcome length limitations in MVS. Many of these can be changed through the Admin Console. But they're also used as ties to the underlying security structure, which means changing them may introduce a breakage in the security operations. Specifically, the server short name is related to the servant STARTED profile. Change a server short name and it's possible the servant region won't start properly. Similarly, the Cluster Transition Name (admittedly not something that typically carries a system identifier) is related to the CBIND and SERVER profiles.

Note: Though not necessarily. It's possible to have STARTED, CBIND and SERVER profiles that are "generic" enough to work despite minor changes to the name.

Bottom line: change names only if you know what the downstream effect will be.
Virtual Host Aliases

Final note ... virtual host aliases may need changing, depending on whether you have users accessing applications with a host-specific alias.


WebSphere will compare the host/port pair on the URL to all the "aliases" in the virtual host the application is bound to. (The default VH is default_host.)

/DeploymentManager
   <config>
      /cells
         /<cell_long>
            virtualhosts.xml
               <aliases xmi:id="HostAlias_2" hostname="*" port="1234"/>
               <aliases xmi:id="HostAlias_2" hostname="wsc3.washington.ibm.com" port="1234"/>
         
         If "hostname" is an asterisk, you're okay -- any IP host name on URL will be permitted
         But if the host name is specific, and you've changed the host to which users point, you will need to update the alias

Note: if you have both asterisk and specific host reference (two aliases -- a common thing), the wildcard asterisk takes precedence and the request will be honored

One final point ... there's a chance that one other file may need updating. It's the virtualhosts.xml file, and it's located at the cell level. This file is used to permit or restrict access to an application based on the host value found on the URL. Inside that file are "aliases" that consist of a host value and a port. Combined, the two make up the alias.

If the host value is an asterisk, that means that it doesn't matter what host value is on the URL. The request will be permitted and the application run. But if the host value is an explicit host reference, then the URL needs to contain that same host or the application request will be denied.

By moving the node you may have created the need for users to modify their request URLs and specify a different host value. The impact of this is that if you have an application that's mapped to a virtual host where the alias is restricted to an explicit IP host value, then the new application requests may not work. It would require climbing in the serverindex.xml file (or accessing this through the Admin Console) and changing the host value on the alias.

Exceptions:

- If you have an alias of asterisk and an alias with an explicit IP host, both with the same port, then the asterisk alias takes precedence and the request is permitted.

- If your users access the system through a front-end webserver that then forwards the request on, then the users may be using a common host value that routes to the front-end device. In that case, the movement of the node won't affect virtual host processing. You'll likely have an alias with that front-end device's host name, and that value doesn't change.
Special Case:

The Deployment Manager node
The DMGR is Mobile

Unlike application servers, the DMGR is designed to be started temporarily on other MVS images in the cell.

![Diagram showing SYSC and SYSD with DMGR and other components]

If SYSC is down for maintenance, DMGR can be started on SYSD so configuration updates can be made. DMGR will ignore system name references and allow DMGR to start on another MVS image.

**Key:** If this is planned properly, the DMGR can be started on another MVS image without any updates to XML files.

To do this involves the use of several z/OS features:
- DVIPA addresses
- VipaDistribute definitions
- Sysplex Distributor Generic IP host names

Written up in white paper ...

The Deployment Manager is a special kind of server in that it is designed to be started on another MVS image as a way of providing flexibility and failover. If the system name reference in the server.xml file doesn't match the actual system name, the DMGR will start anyway, provided the rest of the configuration is okay.

If this is planned properly, you can make sure the DMGR will start on another MVS image quite easily. It involves the use of a DVIPA address, VipaDistribute statements and a Sysplex Distributor generic IP host name. In other words, it involves taking advantage of the dynamic capabilities of z/OS TCP routing to provide the mobility of the DMGR. How this is done is written up in great detail in a white paper.
WP100585 on Techdocs

The full details on how to plan for and implement maximum mobility of the V6 Deployment Manager is provided in Techdoc WP100585:

20 pages long

Two sections:
- Creating new configuration with maximum mobility
- Changing existing configuration so DMGR will be mobile


Starting the DMGR on a Different MVS Image

We'll give you a chart here to give you the main concepts ...

White paper WP100585 on Techdocs was written for V6, so there's no worry about things that are different in the white paper from how the product itself works. It's 20 pages long and has two main sections: configuring for maximum DMGR mobility when you're building a cell fresh, and changing an existing configuration to incorporate the Sysplex Distributor definitions.
As mentioned, the key to this is Sysplex Distributor. By using a DVIPA address and a generic IP host name associated with that address, it's possible to provide clients with a IP host name that is generic to the Sysplex. TCP on the Sysplex figures out where that IP address is actually hosted. So if a MVS image in the Sysplex goes down, that DVIPA is quickly and automatically moved to another MVS image, and clients have no idea this occurred.

In addition to this, VipaDistribute statements provide a way to route traffic to whatever MVS image the ports defined on the VipaDistribute are open. If the ports are open on multiple MVS images, then the VipaDistribute balances the traffic based on WLM-based recommendations. So in the case of Daemon ports defined on a VipaDistribute, where multiple Daemons are listening on the same ports but on different MVS images, the Sysplex Distributor will balance inbound traffic to the best Daemon to service the request.

But let's say it's a Deployment Manager, and you have only one in your cell. So there's no need to "load balance" between multiple DMGR's. What's the value? The value is that Sysplex Distributor will route requests to the MVS image where that port is open. Clients point their browsers at the generic IP host name, and Sysplex Distributor routes the request to whatever MVS image the DMGR is running on. And here's the key: a "client" may be an external browser client or an internal client like a Node Agent. Therefore, for internal communications back to the DMGR, the rest of the WebSphere configuration can be made aware of the DMGR living on the generic IP host name, which means that DMGR can live anywhere in the Sysplex.
Concluding Points
Concluding Points

- Nodes can be moved to other MVS images in the Sysplex
- Process involves modifying XML and transforming was.env files
- There are certain key restrictions (see "Key Assumptions" earlier)
- Known limitation is the issue of names that contain system identifier
- The Deployment Manager is a special server -- properly configured, it can start on other MVS images without modifying XML
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2, 2005</td>
<td>Original document.</td>
</tr>
<tr>
<td>September 7, 2005</td>
<td>Document number assigned and added to the document.</td>
</tr>
<tr>
<td>September 27, 2005</td>
<td>Fixes a few instances where directory reference was to /profiles/default/config/bin when in reality there is no &quot;config&quot; in the path.</td>
</tr>
</tbody>
</table>

End of Document