WebSphere Application Server for z/OS Version 5.1

Moving Nodes To Another MVS Image within a Sysplex

Or:

"How to change the System Name and IP Host Names within a Node's Configuration Files"

With a bonus section on how to change the mount point information for a node

This document can be found on the web at:
www.ibm.com/support/techdocs
Search for document number WP100542 under the category of "White Papers"

Version Date: February 12, 2005
See "Document Change History" on page 95 for a description of the changes in this version of the document

IBM Washington Systems Center
Don Bagwell
IBM Washington Systems Center
301-240-3016
dbagwell@us.ibm.com
Many thanks go to Chris Vignola of the WebSphere for z/OS development group, and Mike Cox and John Hutchinson of the Washington Systems Center. And Mike Loos of the WebSphere Technical Sales Support team.
Table of Contents

Executive Overview .................................................................................................................. 5
Basic issue being addressed ..................................................................................................... 5
Where the challenges exist ....................................................................................................... 5
What's involved ......................................................................................................................... 6
Relationship to Disaster Recovery testing ............................................................................. 6
Important note about system-specific naming conventions .................................................... 6
Difference between this document and WP100415 ................................................................. 6
Disclaimer .................................................................................................................................. 7

Sample Starting Configuration and Different Scenarios .............................................................. 8
Starting Sample Configuration .................................................................................................. 8
What about a Base Application Server node? .......................................................................... 8
Scenario 1 -- Move node and Daemon on SYSD to SYSB .......................................................... 9
Scenario 2 -- Move node and Deployment Manager to a new MVS image ............................. 10
Scenario 3 -- Move node to a new MVS image where Daemon does not already exist .......... 11
Scenario 4 -- Move node to a new MVS image where Daemon already exists ....................... 12
Scenario 5 -- Move Base Application Server node and its Daemon to a new MVS image ....... 13
The requirement that basic WebSphere requirements "be there" on the target MVS image ....... 13
A note about duplication of information in this document ...................................................... 13

The Essential Knowledge About the Configuration Structure ................................................ 14
Everything kept in the HFS ....................................................................................................... 14
Configuration hierarchy for each node structure ...................................................................... 14
Each node may have its own mount point, or all nodes under one mount point ....................... 14
The mount point name is maintained in the configuration XML files ...................................... 15
XML files maintained in ASCII format .................................................................................... 15
The different "levels" of the configuration hierarchy .................................................................. 16
Deployment Manager's configuration structure has "Master Configuration" ............................ 16
Synchronization involves copying changes from Master Configuration out to nodes ............. 17
The was.env file and how it's updated to reflect changes in the other XML files .................... 18
The Daemon's was.env file and how that's updated ................................................................. 18
The Deployment Manager's was.env file and how that's updated .......................................... 18
A view of the process ... from a high level ................................................................................ 18

Scenario 1 in Detail .................................................................................................................... 20
Stop the servers in the application server node and the Daemon that supports that node ......... 20
Change system names .............................................................................................................. 20
Change system name reference in the node-level variables.xml file .................................... 20
Change system name reference in application server's server.xml file ................................ 21
Change system name reference in the Node Agent's server.xml file ....................................... 22
Change IP host names .............................................................................................................. 22
IP host names in variables.xml at the cell level .................................................................... 23
IP host names in serverindex.xml at the node level ............................................................... 24
IP host addresses in server.xml at the server level ............................................................... 25
IP host names in virtualhosts.xml at the cell level .............................................................. 25
Make certain the Deployment Manager is up and running .................................................... 27
Run syncNode.sh to synchronize changes and rebuild node's was.env files ....................... 27
Start servers ............................................................................................................................ 28
Task completed ....................................................................................................................... 28

Scenario 2 in Detail .................................................................................................................... 29
Stop the Deployment Manager, the servers in the application server node, and the Daemon .... 29
Change system names .............................................................................................................. 29
Change system name reference in node-level variables.xml file for DMGR node ................. 29
Change system name reference in node-level variables.xml file for appserver node .......... 30
Change system name reference in Deployment Manager's server.xml file .......................... 31
### Scenario 3 in Detail

#### Change system names

- Update the Deployment Manager node's `serverindex.xml`.
- Update the application server node `serverindex.xml`.
- Change IP host names in `virtualhosts.xml` at the cell level.
- Change IP host addresses in `server.xml` at the server level.
- For the Deployment Manager node.
- For the application server node.

#### The dilemma

- IP host names in `variables.xml` at the cell level.

#### Decision

- Possible solutions

#### Key message: we created new Daemon instance; we need unique "specific UUID" value

- Capture "specific UUID" value of Daemon that supports Deployment Manager

#### Background on "specific UUID" value found in Daemon `was.env`

- Calculating a unique UUID for the new Daemon
- Change system name reference in the node-level `variables.xml`
- Change system name reference in application server's `server.xml`
- Change system name reference in the Node Agent's `server.xml`

#### Change IP host names

- IP host names in `variables.xml` at the cell level
- IP host names in `serverindex.xml` at the node level
- IP addresses in application server `server.xml` at the server level
- IP host names in `virtualhosts.xml` at the cell level

#### Regenerate the Deployment Manager's `was.env` file with `bbodc2ns.sh` shell script

- Start the Deployment Manager on the target MVS image
- Run `syncnode.sh` to synchronize changes and rebuild node's `was.env` files
- Start servers
- Task completed

#### Scenario 4 in Detail

#### A challenge we faced -- you may not

- The dilemma
- Possible solutions
- Decision
- The port remapping challenge

#### Node Agent ports remapped

- Edit `serverindex.xml` and change the Node Agent TCP port values
- Run `syncNode.sh` from node's `/bin` directory

#### Stop the servers in the node

- Change system names
- Capture target MVS image Daemon "specific UUID" value
- The short story -- steps required to capture Daemon specific UUID value
- The longer story -- why we need to do this
- Change system name reference in the node-level `variables.xml`
Appendix A - Other Information

Scenario 5 in Detail

Stop the server and the Daemon

Change system names

Change system name reference in application server's server.xml file

Change system name reference in the Node Agent's server.xml file

Change IP host names

IP host names in variables.xml at the cell level

IP host names in serverindex.xml at the node level

IP host addresses in server.xml at the server level

IP host names in virtualhosts.xml at the cell level

Make certain the Deployment Manager is up and running

Run syncNode.sh to synchronize changes and rebuild node's was.env files

Start servers

Task completed

Easiest thing -- maintain same mount point

Single HFS for all nodes?

Common JCL in use across different nodes?

Scenario 1 -- remapping mount point for single node

Create mount point, provide permissions/ownership and mount HFS

Update node-level variables.xml in DMGR master configuration

Update setupCmdLine.sh in application server node's /bin directory

Run syncNode.sh to synchronize master configuration changes out to node

Modify JCL start procedures to update SET ROOT= value

Start servers on new MVS image

Scenario 2 -- remapping mount point for multiple nodes

Create mount point, provide permissions/ownership and mount HFS

Update node-level variables.xml in DMGR master configuration

Update setupCmdLine.sh in both nodes' /bin directory

Run bbowc2ns.sh to regenerate the Deployment Manager node

Modify JCL start procedures to update SET ROOT= value

Start the Deployment Manager

Run syncNode.sh to synchronize master configuration changes out to node

Modify JCL start procedures to update SET ROOT= value

Start servers on new MVS image

Scenario 5 -- remapping mount point for Base Application Server node

Create mount point, provide permissions/ownership and mount HFS

Update node-level variables.xml in BaseApp's configuration

Update setupCmdLine.sh in node's /bin directory

Run bbowc2ns.sh to regenerate the Base Application Server node

Modify JCL start procedures to update SET ROOT= value

Start the server
Scenario 3 -- copying out a node structure to a new HFS on a new mount point ........................................... 89
Create mount point, provide permissions/ownership and mount HFS ................................................................. 90
Copy application server node structure over to new HFS ..................................................................................... 90
Copy the Daemon structure over to new HFS ........................................................................................................ 90
CHOWN both new directory structures .............................................................................................................. 90
Update node-level variables.xml in DMGR master configuration ........................................................................ 91
Update setupCmdLine.sh in application server node's /bin directory ................................................................. 91
Run syncNode.sh to synchronize master configuration changes out to node ..................................................... 91
Modify JCL start procedures to update SET ROOT=value .............................................................................. 92
Start servers on new MVS image ...................................................................................................................... 92
The difference between changing all system name reference in variables.xml vs. just one ................................. 92
Changing just WAS_DAEMON_ONLY_server_configured_system_name value .................................................. 93
Changing all instances of the system name string in the file .............................................................................. 94

Document Change History ............................................................................................................................... 95
Executive Overview

Basic issue being addressed

WebSphere Application Server for z/OS Version 5.1 has a configuration structure that ties a "node" to a specific system. If you initially create a node on SYSA, you can't just start it on SYSB. The configuration XML files for that node will have references within them to the MVS system and the TCP Host Name of the MVS image on which it sits. Try to start the node on a different MVS image and WebSphere will detect the difference and fail to come up:

Simply moving a node over to another MVS image isn't enough -- servers won't start

However, if you edit a few XML files, change a few values and do a few other things, the node will come up on another MVS image.

Servers made to understand they now live on a different MVS image within the Sysplex

Where the challenges exist

Editing the XML files is a fairly easy thing to do. They're in ASCII within the HFS, so you'll need to take that into account. But once you've got an XML file open in an editor, the act of changing the necessary fields is pretty straight-forward.

Knowing where the files are located is quite another thing. And if you're unfamiliar with the hierarchy of the configuration directory structure, then this topic can be challenging. This paper will strive to provide that information.
What's involved

Moving a WebSphere for z/OS Version 5.1 node to another MVS image within the same Sysplex involves the following three broad activities:

- Changing the System Name identifier in the variables.xml files and the server.xml files
- Changing the IP Host Name string in the following files: variables.xml, serverindex.xml, server.xml and virtualhosts.xml.
- Synchronizing the changes from the "Master Configuration" to the node structure and then starting the servers on the new MVS image.

Note: What about a "Base Application Server node?" Thought a BaseApp node has by definition only one node, and the idea of synchronizing with other nodes doesn't apply to it, the basic concepts are the same. We cover the Base Application Server node scenario at "Scenario 5 in Detail" on page 70.

Relationship to Disaster Recovery testing

Testing a disaster recovery plan typically involves bringing up a configuration on another system. Sometimes that target system is not exactly like the source -- particularly with respect to system names and IP host names. What's covered in this white paper is very similar -- changing system names and IP host names. So while this paper doesn't directly speak to the topic of Disaster Recovery testing, the concepts are directly applicable to that subject.

Important note about system-specific naming conventions

This document does not address the issue of changing the names of your servers and nodes after you move them. The desire to change the names may occur if your naming convention is designed with a "system identifier" somewhere in the name itself. If so, then the name of a server moved from one MVS image to another may no longer be meaningfully associated with the system.

This is a difficult challenge to overcome. The short names of the servers and nodes can be changed easily enough within the Admin Console function. But the long names are more difficult -- the Admin Console function has no facility to change long names. And whether the long names can be easily changed through another means is unclear at the present time.

Key Point: Therefore, the whole topic of naming conventions is simply not addressed in this document. We'll assume that when a node and its servers are moved from one MVS image to another, the names of those things remain the same.

Difference between this document and WP100415

The WP100415 white paper was dedicated to the process used to temporarily bring the Deployment Manager node only up on another MVS image. The process documented in that white paper involved changing the Deployment Manager's "End Point" host names through the Admin Console. This results in changes to the serverindex.xml file only, and not the other files we document in this white paper. Most importantly, the system name references in the Deployment Manager's XML files are not changed. That means when the Deployment Manager is started on another MVS image, it detects that it's starting on another MVS image. It then goes into what's known as "Peer Restart Recovery Mode."

This white paper documents the "permanent" movement of a node to another MVS image. We use the term "permanent" here not to mean you can never move it again, but rather to suggest that the node -- its XML files updated to indicate the new system name -- "thinks" the new MVS image is its correct and proper home.
Disclaimer

Some of the scenarios presented here have the potential to involve some complex modifications to the XML files. **Back up your configuration HFS prior to performing these modifications.** Also, it's always possible that your environment has some unique quality that requires an additional step, or a slightly different approach -- one not anticipated by this document, or not specifically documented here.

So consider this document to be one that provides the essential points in a near-cookbook fashion. But please do not think the steps provided here are 100% applicable to all environments.
Sample Starting Configuration and Different Scenarios

To best illustrate how this is done, we'll use a sample configuration. (But it's an actual operating configuration at the Washington Systems Center.) Then we'll show several different scenarios where the nodes were moved.

Starting Sample Configuration

The starting sample configuration consisted of a Network Deployment configuration with two application server nodes. It spanned two MVS images -- SYSC and SYSD of a four image Sysplex:

```
SYSA
wsc1.washington.ibm.com
SYSB
wsc2.washington.ibm.com
SYSC
1
G5DEMN
G5DMGR
2
G5AGNTC
G5SR01C
SYSD
G5DEMN
3
G5AGNTD
4
G5SR02D
wsc3.washington.ibm.com
wsc4.washington.ibm.com
```

"G5CELL" configuration that served as the starting point for the moving of nodes

Notes:

1. Daemon server instance for nodes on SYSC
2. Deployment Manager node (G5NODE) and server (G5DMGR)
3. Application Server node on SYSC. Node name is G5NODEC; Node Agent name is G5AGNTC and application server name is G5SR01C.
4. Daemon server instance for nodes on SYSD. Same name as Daemon on SYSC.
5. Application Server node on SYSD. Node name is G5NODED; Node Agent name is G5AGNTD and application server name is G5SR02D.

Note: If your nodes have more application servers in them than just one like we have here, don't worry ... the process and concepts still apply. It involves more editing in the serverindex.xml file, and more copies of server.xml. So it's really just more work, but not different work from what's written about in this paper.

What about a Base Application Server node?

The concepts and the process are nearly identical. There just isn't a "Master Configuration" and there's no need (or ability) to "synchronize" the nodes. We cover the Base Application Server node scenario separately. (See "Scenario 5 in Detail" on page 70.)
Scenario 1 -- Move node and Daemon on SYSD to SYSB

Here we moved the entire structure on SYSD to another MVS system:

Scenario 1 -- application server node on SYSD moved to SYSB

This is probably the most common scenario. It's also one of the simpler ones. The fact that both the node structure and the Daemon are moving at the same time makes this easier than, say, moving just the node.

Since both are moving to a new MVS image, the system name references and IP host name references change.

For more, see "Scenario 1 in Detail" on page 20.
**Scenario 2 -- Move node and Deployment Manager to a new MVS image**

The Deployment Manager node and the application server node on SYSC were moved to SYSA.

---

**Scenario 2 -- application server node and DMGR on SYSC moved to SYSA**

This is very similar to Scenario 1, except that two nodes are being moved instead of one (the application server node and the Deployment Manager node). The Daemon server that supported both moved as well. Also, this scenario calls for regenerating the Deployment Manager’s was.env file, which was not required in Scenario 1.

This is a bit more complicated than Scenario 1, but not terribly so. The key thing is that two nodes are moved, not just one.

For more, see "Scenario 2 in Detail" on page 29.
Scenario 3 -- Move node to a new MVS image where Daemon does not already exist

The application server node on SYSC was "split away" from the Deployment Manager and moved to SYSA. A new Daemon instance was created.

This was fairly complicated because it called for the creation of a new Daemon server instance. That involved copying the configuration structure of an existing Daemon instance and then making changes in the XML files so the new instance would be unique. In other ways its similar to what's done in Scenario 4.

For more, see "Scenario 3 in Detail" on page 41.
Scenario 4 -- Move node to a new MVS image where Daemon already exists

The application server node on SYSC was "split away" from the Deployment Manager and moved to SYSD. But it was placed on an MVS image that already had a node for this cell, which meant that a Daemon server instance was already present.

---

Scenario 4 -- application server node on SYSC moved to SYSD, alongside another node

This scenario turned out to be somewhat complex, primarily due to the fact that in our sample scenario our Node Agents -- one on SYSC and one on SYSD initially -- were configured with the same port values (as per the Washington System Center's recommendations). However, when the application server node was moved to SYSD that meant two Node Agents on the same MVS image vying for the same TCP ports. So the ports for one of the Node Agents had to be remapped.

Another complicating factor had to do with how syncNode.sh updated the Daemon's was.env, and what we had to do to maintain the unique "UUID" value for the Daemon on SYSD. The details of this are documented in the section dedicated to this scenario.

**Note:** As a general rule, we don't recommend hosting two different application server nodes on the same MVS image within the same cell. It can be done ... there's no technical reason why it can't. It's just that it doesn't provide the isolation and separation many people think it will. Nevertheless, we'll show how to do this just in case someone is interested.

For more, see "Scenario 4 in Detail" on page 55.
**Scenario 5 -- Move Base Application Server node and its Daemon to a new MVS image**

Very simple -- a Base Application Server node moved from SYSC to SYSD.

![Diagram showing SYSC and SYSD with wsc3.washington.ibm.com and wsc4.washington.ibm.com connected by an arrow to G5SR01C and G5DEMNC.]

---

**Scenario 5 -- Base Application Server node configuration moved from SYSC to SYSD**

They key difference between this scenario and Scenario 1 is that here we don't have a "Master Configuration" to synchronize. So we made changes directly to the application server node's HFS structure. The server's was.env needed to be updated (something done by syncNode.sh in the other scenarios, but not here because there's nothing to synchronize with), so we accomplished that with the bbowc2ns.sh shell script.

For more, see "Scenario 5 in Detail" on page 70.

**The requirement that basic WebSphere requirements "be there" on the target MVS image**

One of the basic assumptions we're making throughout this document is that when a node is moved from one MVS image to another, everything WebSphere needs to operate is available on that target MVS image. For instance, the WebSphere installation data sets are there; or all the other WebSphere prerequisites are met.

The chances of this being the case when moving within a Sysplex is higher than when moving a whole cell between Sysplexes.

If you're interested in a more thorough review of the things we're thinking of here, pull PRS1137 from www.ibm.com/support/techdocs.

**A note about duplication of information in this document**

There is quite a bit of overlap between the five scenarios. In writing this document, we faced the choice of putting all the common stuff in one spot and then branching you off for the unique things in each scenario. But that's often very confusing. So instead we opted to document each scenario from front to back. That resulted in a good deal of duplicate text in the document. If you read this document all the way through, you'll see a lot of the same things discussed over and over again.
The Essential Knowledge About the Configuration Structure

If we were to say, "locate and edit the variables.xml file in the node level of the master configuration," would you know what we were talking about? If so, then you might be able to skip over this section of the document. But if you're not clear on how the configuration HFS is structured, you might benefit from at least skimming through this section.

Everything kept in the HFS

In WebSphere Application Server for z/OS Version 5.1, all the configuration information is kept in the HFS. Nothing is kept in other, external data sources such as DB2 or LDAP, as was the case in WebSphere Version 4 on z/OS.

Why is this important? Because it says that to move a node from one MVS image to another, it involves changing information in files in the HFS. But nowhere else. That's good news.

Configuration hierarchy for each node structure

The configuration information is maintained in a directory "tree" (or hierarchy), which extends down quite a ways. We'll discuss the structure of it in just a bit. For now the point we're making is that each node in a cell has its own configuration hierarchy:

Each node in a cell has its own configuration hierarchy

This may mean each has its own HFS; it may not. It's possible to have all the nodes in a cell contained in the same HFS under the same mount point. It's equally possible to have separate HFS for each under their own mount points.

Each node may have its own mount point, or all nodes under one mount point

A multi-node Network Deployment configuration will have multiple directory structures, as illustrated in the prior section. Whether all the nodes reside under the same mount point, or under separate mount points, is entirely up to you. During initial customization, you specify the mount point for the node being built.

Whether your nodes are under a common, shared mount point or under separate mount points doesn't really impact how to change the system names and IP host names in the XML files.
The process is the same either way. But it may impact how you make the configuration structure available to the new MVS image:

- **Common mount point** -- this implies a shared HFS across the MVS images in the Sysplex. Here you need not copy or move the HFS to start the node's servers on the new MVS image: the HFS is shared and already available.

- **Separate mount points** -- this implies separate HFS data sets, which means when the node is moved from one MVS image to another, you'll need to copy or move that HFS to the new MVS image. We bring this up as a lead-in to the next topic, which has to do with the name of the mount point, and specifically how it's important that mount point name *not change* when the node is moved.

**The mount point name is maintained in the configuration XML files**

The name of the mount point is imbedded in the XML files of the configuration HFS. So you can't simply mount the configuration HFS at an entirely different mount point and have the configuration work. It won't -- WebSphere will fail to operate properly if it expects to see its stuff under `/wasy51config/g5cell`, for instance, and instead the files are *really* located under something different, like `/xyz`.

This paper was written with the assumption that the mount point would *not* change when a node was moved from one MVS image to another. That would be the case if you have a common, shared HFS across all MVS images, such as we did for our sample configuration.

However, if you have unshared HFS, and more specifically if your mount point names are system specific, then moving a node from one MVS image to another may imply changing the name of the mount point as kept in the XML files.

**Note:** If you have unshared HFS but the mount point on each MVS image is some common name -- the same on each MVS image -- then you're okay. No need to change the name in the XML files.

Rather than dwell on this topic in each scenario write-up, we opted to split the topic of changing the mount point information to its own section, way at the back of the document. So if you're interested in this topic, then see "Changing the mount point information in node-level variables.xml" on page 77.

You'll be responsible for knowing whether the topic applies to your environment.

**XML files maintained in ASCII format**

To move a node to another MVS image will require editing XML files and changing the values of definitions within the files. The XML files are in ASCII encoding in the HFS. That means that the traditional methods of editing EBCDIC files -- OEDIT, for example -- will not work. You'll see what looks like gibberish. So you'll have to use an ASCII editor.

There are lots of different ways you can accomplish editing ASCII files in the HFS. For the sake of this document, we'll assume you have one you prefer to use.

**Message:** Use whatever tool you are most comfortable with to edit the ASCII files.
The different "levels" of the configuration hierarchy

Under each of the configuration hierarchy structures for a node there exists three "levels" of directories, with XML files in each:

- **Cell Level** -- XML files that are related to the cell as a whole, and not to this particular node
- **Node Level** -- XML files that are related to that particular node
- **Server Level** -- XML files that are related to each server in the node

The "levels" of a configuration hierarchy

The reason we point this out is because depending on what aspect of the configuration you're changing, you may have to go to the "cell," "node" or "server" level.

Another reason to understand this structure is because some files -- the variables.xml file in particular -- exists at all three levels. Knowing which copy to edit is critical.

Deployment Manager's configuration structure has "Master Configuration"

The Deployment Manager server has its own node structure, and it has a configuration hierarchy maintained in the HFS ... just like other nodes. But there's a key difference: the Deployment Manager configuration contains what's known as the "Master Configuration." You can think of the "Master Configuration" as a copy of the configuration files for each of the application server nodes in the cell:
The "Master Configuration" contains a copy of the application server node configuration

Why is this important? Because it's in the "Master Configuration" that we'll make our XML updates. We won't make the updates in the actual node structures. If we made the changes in the nodes themselves and not in the Master Configuration, then the next time "synchronization" takes place the old values in the Master Configuration will get copied out to the nodes. That would mean your updates would be overwritten.

We could ask you to update the values in multiple places ... but that exposes the process to human error.

Instead, we'll make the updates in the Master Configuration only, then manually invoke synchronization. That'll result in your updates being copied out to the node.

**Note:** Manually invoking synchronization has the added benefit of rebuilding the `was.env` files for the servers as well as the Daemon server. More on that later.

**Synchronization involves copying changes from Master Configuration out to nodes**

To manually invoke synchronization, we're going to use a shell script that comes with WebSphere. The shell script is called `syncNode.sh`, and it's located in the `/bin` directory of each node structure. This shell script will do three fundamentally important things:

1. Copy updates seen in the "Master Configuration" out to the node, updating the XML files in that node's configuration hierarchy
2. Rebuild the `was.env` files for all the servers in the node
3. Rebuild the `was.env` file for the Daemon server that supports the node

It's a very useful utility. It has one key requirement: the Deployment Manager must be up and running for `syncNode.sh` to work. Why? Because `syncNode.sh` will connect across the network to the SOAP port of the Deployment Manager and "pull" the changes found in the Master Configuration down to the node.

When synchronization is performed from the Admin Console, the same thing takes place -- the Deployment Manager signals to the Node Agent to invoke synchronization, and then the Node Agent turns around and requests from the DMGR the changes found in the Master Configuration. It's a "pull" operation, not a "push."
The was.env file and how it’s updated to reflect changes in the other XML files

The was.env file is found in the "server level" directory of every server in the node. It is not an XML file ... it is a parameter file, encoded in EBCDIC, that's made available to the servers for initial startup. Once the server is part of the way up, it can then start parsing and understanding XML. But until that time it needs a standard parameter file.

The was.env file is really a shadow of the things specified in the XML files -- it's a distillation and compilation of the contents of a lot of different XML files, all rolled up into one file. So when changes are made to the XML files, those changes need to be propagated out to the was.env files. The shell script syncNode.sh does this for us.

The Daemon's was.env file and how that's updated

Daemon servers are funny things -- they're not technically part of any node, they don't have any XML files, and they have their own directory structure. But Daemons are critically important to the operation of the servers on an MVS image: stop a Daemon instance and all the servers on an MVS image within that Daemon's cell will also come down. So when a node and its Daemon are moved to another MVS image, making sure the Daemon comes up properly is important.

Daemon's don't have XML files, but they do have a was.env file. That's what the Daemon server process uses as its initialization parameter file. When a node and its Daemon are moved to another MVS image the Daemon's was.env needs updating. But it's not something you do manually. It's something that's done automatically.

When syncNode.sh is run for a node after it's been copied over to a new MVS image, the syncNode.sh shell script will figure out, based on the information in the XML files for that node, the Daemon that supports the node. It'll then go out to that Daemon's was.env file and update it as needed.

Message: Having an operational Daemon is critical. The Daemon's was.env will need to be updated, but it's not something you do. It's something that's done automatically.

The Deployment Manager's was.env file and how that's updated

The Deployment Manager is a server very much like an application server -- it has a controller and a servant, it has XML files and a was.env file.

We noted earlier how the was.env files for application servers in a node are updated when syncNode.sh is run. But what about the Deployment Manager's was.env file? We don't run syncNode.sh to synchronize the Deployment Manager with itself, so we can't rely on that tool.

To update the Deployment Manager's was.env we use another shell script called bbodc2ns.sh. That's in the Deployment Manager's /bin directory. That'll sweep through the Deployment Manager's XML files, see the changes you've made there, then update the was.env for the DMGR with the new information.

A view of the process ... from a high level

We've gone over a lot of things. Let's put it all in perspective so you can get a feel for the process you'd go through to move a node to another MVS image in a Sysplex:

- Copy the node's configuration HFS to the new MVS image (if necessary ... if the HFS is shared you don't have to do this, though you may wish to change the system that owns the HFS)
• Change the XML files in the "Master Configuration" to reflect the new system on which the node will reside

  **Note:** We'll spend more time discussing which files, what to change, etc.

• Regenerate the Deployment Manager's `was.env` by running `bbodc2ns.sh`

  **Note:** This is only necessary if the Deployment Manager itself is moved. If what you're moving is an application server node, but the Deployment Manager stays where it is, then this step is not necessary.

• Start the Deployment Manager (required so `syncNode.sh` can connect and pull changes)

• Run `syncNode.sh` for each node you've moved. This rebuilds the `was.env` of every server in the node as well as the Daemon server's `was.env` file.

• Start servers in the node that's been moved
Scenario 1 in Detail

Recall the scenario:

**Scenario 1 -- application server node on SYSD moved to SYSB**

### Stop the servers in the application server node and the Daemon that supports that node

- The easiest way to do this is to stop the Daemon. That'll bring down all the dependent servers.

### Change system names

References to the system name is kept in two XML files:

- The variables.xml file at the node level
- The server.xml file at the server level

Remember that we're modifying the copies of these files in the "Master Configuration" under the Deployment Manager.

#### Change system name reference in the node-level variables.xml file

- Locate the variables.xml file for the node being moved. In this case it was G5NODED. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
  /config
    /cells
      /g5cell
        /nodes
          /g5noded
            variables.xml
```

Location of node's variables.xml file, down under DMGR's "Master Configuration"
Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Perform a global change, changing all the instances of the "old" system name to the "new" system name. We changed six instances in our copy of the file:

```
"WAS_DAEMON_daemon_start_command_args"
  value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL.SYSB"
"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSB"
"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSB"
"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSB"
"WAS_DAEMON_daemon_was_env_file"
  value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSB/was.env"
"WAS_DAEMON_daemonInstanceName" value="SYSB"
```

**See Notes**

**Different instances of the system name in the node-level variables.xml file**

**Notes:**
- There is more in this file than the lines shown here.
- The lines shown above are not contiguous. Almost, but not entirely so.
- An alternative is to change just the "server_configured_system_name" variable. See "The difference between changing all system name reference in variables.xml vs. just one" on page 92 for more information.

Save the file.

**Change system name reference in application server's server.xml file**

Locate the server.xml file for the application server G5SR02D in the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5noded
            /servers
              /g5sr02d
                server.xml
```

**Location of application server's server.xml file, down under DMGR's "Master Configuration"**

Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate and change **one variable**. The variable we're interested in is:

```
name="was.ConfiguredSystemName" value="SYSD"
```

Change the value="SYSD" to the system name of where the node will be moved:

```
name="was.ConfiguredSystemName" value="SYSB"
```

Save the file.

**Note:** Repeat process for every application server in the node.
Change system name reference in the Node Agent's server.xml file

- Locate the server.xml file for the node agent in the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5noded
              /servers
                /g5sr02d
                  /nodeagent

Mount Point

Server level

server.xml
```

Location of the node agent's server.xml file, down under DMGR's "Master Configuration"

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate and change one variable. The variable we're interested in is:
  
  ```
  name="was.ConfiguredSystemName" value="SYSD"
  ```

- Change the value="SYSD" to the system name of where the node will be moved:
  
  ```
  name="was.ConfiguredSystemName" value="SYSB"
  ```

- Save the file.

Change IP host names

IP host names references are found in four different XML files in the configuration structure. Not all need to be changed.

<table>
<thead>
<tr>
<th>File</th>
<th>Level</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables.xml</td>
<td>cell</td>
<td>Pointer to the host name used by the Daemon instances for this cell. We didn't change this. See &quot;IP host names in variables.xml at the cell level&quot; on page 23 for an explanation why.</td>
</tr>
<tr>
<td>serverindex.xml</td>
<td>node</td>
<td>IP host names used by the various ports for the servers in the node. You will change these.</td>
</tr>
<tr>
<td>server.xml</td>
<td>server</td>
<td>Potential reference to a IP address, but only if an HTTP port is &quot;homed&quot; to a particular adapter. If no HTTP ports are &quot;homed&quot; like that, then you don't need to change anything.</td>
</tr>
<tr>
<td>virtualhosts.xml</td>
<td>cell</td>
<td>Virtual host aliases have IP host names in them. See &quot;IP host names in virtualhosts.xml at the cell level&quot; on page 25 for a discussion on whether you need to change these.</td>
</tr>
</tbody>
</table>
IP host names in `variables.xml` at the cell level

This file was found at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5noded
              /servers
                /g5sr02d
                /nodeagent
```

Location of `variables.xml` file at the cell level

For the simple act of moving a node to another MVS image, you will not need to modify this file. Here's why:

- All the Daemon instances in a Network Deployment configuration are copies of one another. They're not exactly identical but close. *They all have the same host listen address.*

- It does not matter which Daemon instance an external client uses; any one will work equally well. All Daemon instances have full knowledge of the entire cell configuration.

- The recommended best practice is to configure a Sysplex Distributor host name when constructing your Network Deployment Daemons. That way Sysplex Distributor can be used to route inbound requests to one of the Daemon instances.

In our sample configuration copy of that file, we found the following:

```
"WAS_DAEMON_protocol_iiopt_daemon_listenIPAddress"
  value="wsc3.washington.ibm.com"
```

We left this value as it was found. We did not change it.

**Please Read:**

- The reason a system-specific IP address appears there is because when this cell's Deployment Manager was first constructed, the person who built the cell (look no further than the name on this document 😊) failed to use the Sysplex Distributor host name of wscrb.washington.ibm.com. Instead, he used the host name of the SYSC system where the Deployment Manager was going to reside. A careless error on the part of the author of this document. But it will serve as a good instructional tool.

- In this scenario the Deployment Manager’s Daemon remained on SYSC. For that reason we left this value unchanged.

- In truth, we should have gone in and changed this to be a Sysplex Distributor address.

- Again, we left this value unchanged because the Daemon remained on SYSC.
**IP host names in serverindex.xml at the node level**

It's important to understand the structure of this file. This file provides information about the ports (or "end points") in the node. The file can be mapped out like this:

```
<serverindex ... node host address>
  <serverEntries>
    +Port type, number and host address
    +Port type, number and host address
    +etc.
  </serverEntries>
<serverEntries>
  +Port type, number and host address
  +Port type, number and host address
  +etc.
</serverEntries>
</serverindex>
```

**Schematic layout of the serverindex.xml file**

- Locate the serverindex.xml file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5noded
```

**Location of the node's serverindex.xml file, down under DMGR's "Master Configuration"**

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate all the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.

**Note:** In our sample configuration there were two `<serverEntries>` blocks in the file -- one for the G5SR02D server and one for the Node Agent.

- Take a close look at the ORB and ORB_SSL ports for each server. Check the `host=` value on each.
  - If the value is `host="*"` then do nothing
  - If the value is `host="<some IP address>"` then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

- Save the file.
**IP host addresses in server.xml at the server level**

The chances are good you will *not* have to modify this file. The only reference to an IP address is found on the HTTP port definitions, and in many cases the references will be null:

```
"EndPoint_1" host="" port="15548"
```

**Potential location of IP address in the server.xml file**

If the HTTP port (or SSL port -- a separate XML definition in the file) have a dotted-decimal address in that field, it means the ports are homed to a specific adapter. If that's the case, then you have two choices:

- If the value is `host=""` then do nothing
- If the value is `host="<some IP address>"` then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

If you need to change the file, do the following:

- Locate the server.xml file for each server in the node.
  - **Note:** Node Agents do not have HTTP ports, so you do not need to worry about that server.
- Change the HTTP port `host=""` values as appropriate
  - **Note:** There'll be two in there -- one for the non-SSL HTTP port, one for the SSL port.
- Save the file.

**IP host names in virtualhosts.xml at the cell level**

Host names in this file are used in the definition of a "virtual host alias." They are defined at the bottom of the XML file, and look something like this:
See the two aliases with a specific IP host name? That's for the Deployment Manager. When the Deployment Manager is built, by default four aliases for the DMGR HTTP ports will be built: two with asterisk, and two with the actual host name. What that means is that only people who have (in this case) `wsc3.washington.ibm.com` on their URL can access the Admin Console. When the alias has an asterisk, then any host name on the URL will be permitted to run applications behind the port.

When asterisk and specific host name alias exists, then the specific reference takes precedence.

Do you need to modify this file? Probably not, but it depends:

- If the aliases are asterisks, then no change is necessary
- If the aliases are specific, but the alias is that of a front-end load balancing device (rather than a specific MVS image host name), then no change is necessary

Note: A front-end balancing device's host name will not change when a node in WebSphere is moved. So the user's URL will still contain the same host value. Therefore, the alias will still work. You may need to update the balancing device to understand that the node's servers have in fact moved.

- If the aliases are specific, and specific to the MVS image where the note initially resides, then you'll need to change the references to the new MVS image's host name.

Note: For example, if you move the Deployment Manager, then you'll have to change the alias. Otherwise, you won't be able to access the Deployment Manager.

For this scenario we did not modify this file.

If the file does need changing, then do the following:

- Locate the `virtualhosts.xml` file. It's at the cell level:
Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.

Save the file.

Make certain the Deployment Manager is up and running

The next step -- running syncNode.sh -- depends on the Deployment Manager being up to work.

Run syncNode.sh to synchronize changes and rebuild node’s was.env files

The syncNode.sh shell script will do two important things:

- Take the changes you just made in the XML files under the Deployment Manager's "Master Configuration" and pull them out to the application server node's configuration directory structure.

  Note: Yes, indeed. Synchronization -- even when initiated from the Admin Console -- is a pull operation. The node pulls the changes down to it; the DMGR never pushes them out.

- It rebuilds the was.env file for all the servers in the node, including the Daemon server.

What's important to understand is that the syncNode.sh file is located in the /bin directory of the application server's node configuration, not the Master Configuration:

To synchronize the node, do the following:

- Make sure Deployment Manager is up.
- Make sure servers in the node being moved -- particularly the Node Agent -- are down.
- Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
- If security is enabled for the Deployment Manager, understand what the administrative userid and password is for the cell.
- Open up a telnet session, or OMVS session, and go to the /bin directory of the application server node configuration being moved.
"Switch users" to the WebSphere Admin ID.

Here's the command we entered to initiate the synchronization:

```
./syncNode.sh wsc3.washington.ibm.com 15510 -conntype SOAP -user g5admin -password xxxxxx
```

Command used to synchronize node with changes made in Master Configuration

**Note:** In our environment security was enabled.

Look for the messages that indicate success:

- ADMU0401I: Begin syncNode operation for node g5noded with Deployment Manager wsc3.washington.ibm.com: 15510
- ADMU0016I: Synchronizing configuration between node and cell.
- ADMU0402I: The configuration for node g5noded has been synchronized with Deployment Manager wsc3.washington.ibm.com: 15510

**Start servers**

The servers are now ready to start on the new target MVS image.

**Note:** The start commands for the servers are exactly the same. We haven't changed the proc names, the JOBNAMEs or the ENV= strings.

Start the servers on the new MVS image.

**Task completed**

Your node should now be running on the new MVS image:

Scenario 1 -- application server node on SYSD moved to SYSB
Scenario 2 in Detail

This is almost identical to the process illustrated in Scenario 1, except we have two nodes to move, not one (the application server node and the Deployment Manager node), and we need to regenerate the DMGR's was.env before it'll start on its new MVS image.

Recall the scenario:

```
<table>
<thead>
<tr>
<th>SYSA</th>
<th>SYSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>G5DEMN</td>
<td>G5DMGR</td>
</tr>
<tr>
<td>G5AGNTC</td>
<td>G5SR01C</td>
</tr>
</tbody>
</table>
```
```
<table>
<thead>
<tr>
<th>SYSC</th>
<th>SYSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsc1.washington.ibm.com</td>
<td>wsc2.washington.ibm.com</td>
</tr>
</tbody>
</table>
```

Scenario 2 -- application server node and DMGR on SYSC moved to SYSA

Stop the Deployment Manager, the servers in the application server node, and the Daemon

- The easiest way to do this is to stop the Daemon. That'll bring down all the dependent servers.

Change system names

References to the system name is kept in two XML files:

- The variables.xml file at the node level
  Two nodes in the sample configuration: G5NODE (the Deployment Manager node) and G5NODEC (the application server node on SYSC)

- The server.xml file at the server level
  Three servers in the sample configuration: G5DMGR (the Deployment Manager), G5AGNTC (the Node Agent on SYSC) and G5SR01C (the application server)

Remember that we're modifying the copies of these files in the "Master Configuration" under the Deployment Manager.

Change system name reference in node-level variables.xml file for DMGR node

- Locate the variables.xml file for the Deployment Manager node being moved. In this case it was G5NODE. In our sample configuration, the file was located at the following location:
Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Perform a global change, changing all the instances of the "old" system name to the "new" system name. We changed six instances in our copy of the file:

```xml
"WAS_DAEMON_daemon_start_command_args" value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL.SYSA"
"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSA"
"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSA"
"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSA"
"WAS_DAEMON_daemon_was_env_file" value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSA/was.env"
"WAS_DAEMON_daemonInstanceName" value="SYSA"
```

Different instances of the system name in the node-level variables.xml file

**Notes:**
- There is more in this file than the lines shown here.
- The lines shown above are not contiguous. Almost, but not entirely so.
- An alternative is to change just the "server_configured_system_name" variable. See "The difference between changing all system name reference in variables.xml vs. just one" on page 92 for more information.

Save the file.

**Change system name reference in node-level variables.xml file for appserver node**

Locate the variables.xml file for the application server node being moved. In this case it was G5NODEC. In our sample configuration, the file was located at the following location:

```xml
/wasv51config/g5cell
    /DeploymentManager
        /config
            /cells
                /g5cell
                    /nodes
                        /g5node
                            variables.xml
                        /g5noded
```

Location of variables.xml file for the application server node

See Notes
Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Perform a global change, changing all the instances of the "old" system name to the "new" system name. We changed six instances in our copy of the file:

```xml
"WAS_DAEMON_daemon_start_command_args"
  value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL.SYSA"
"WAS_DAEMONONLY_control_region_configured_system" value="SYSA"
"WAS_DAEMONONLY_server_configured_system_name" value="SYSA"
"WAS_DAEMONONLY_server_specific_short_name" value="SYSA"
"WAS_DAEMON_daemon_was_env_file"
  value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSA/was.env"
"WAS_DAEMON_daemonInstanceName" value="SYSA"
```

**Different instances of the system name in the node-level variables.xml file**

**Notes:**
- There is more in this file than the lines shown here.
- The lines shown above are not contiguous. Almost, but not entirely so.
- An alternative is to change just the "server_configured_system_name" variable. See "The difference between changing all system name reference in variables.xml vs. just one" on page 92 for more information.

Save the file.

**Change system name reference in Deployment Manager's server.xml file**

Locate the server.xml file for the Deployment Manager. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /gnode
              /servers
                /dmgr
                  server.xml
```

**Location of Deployment Manager's server.xml file**

Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate and change one variable. The variable we're interested in is:

```xml
name="was.ConfiguredSystemName" value="SYSC"
```

Change the value="SYSC" to the system name of where the node will be moved:

```xml
name="was.ConfiguredSystemName" value="SYSA"
```

Save the file.

**Change system name reference in application server's server.xml file**

Locate the server.xml file for the application server. In our sample configuration, the file was located at the following location:
Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate and change one variable. The variable we're interested in is:

```
name="was.ConfiguredSystemName" value="SYSC"
```

Change the value="SYSC" to the system name of where the node will be moved:

```
name="was.ConfiguredSystemName" value="SYSA"
```

Save the file.

**Note:** Repeat process for every application server in the node.

**Change system name reference in Node Agent's server.xml file**

- Locate the server.xml file for the Node Agent. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5nodec
              /servers
                /g5sr01c
                  /nodeagent
          /servers
              /g5sr01c
                  /nodeagent
```

Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate and change one variable. The variable we're interested in is:

```
name="was.ConfiguredSystemName" value="SYSC"
```

Change the value="SYSC" to the system name of where the node will be moved:

```
name="was.ConfiguredSystemName" value="SYSA"
```
Change IP host names

IP host names references are found in four different XML files in the configuration structure. Not all need to be changed.

<table>
<thead>
<tr>
<th>File</th>
<th>Level</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables.xml</td>
<td>cell</td>
<td>Pointer to the host name used by the Daemon instances for this cell. Unlike in Scenario 1, we did change this. See &quot;IP host names in variables.xml at the cell level&quot; on page 33 for an explanation why.</td>
</tr>
<tr>
<td>serverindex.xml</td>
<td>node</td>
<td>IP host names used by the various ports for the servers in the node. You will change these.</td>
</tr>
<tr>
<td>server.xml</td>
<td>server</td>
<td>Potential reference to a IP address, but only if an HTTP port is &quot;homed&quot; to a particular adapter. If no HTTP ports are &quot;homed&quot; like that, then you don't need to change anything.</td>
</tr>
<tr>
<td>virtualhosts.xml</td>
<td>cell</td>
<td>Virtual host aliases have IP host names in them. See &quot;IP host names in virtualhosts.xml at the cell level&quot; on page 37 for a discussion on whether you need to change these.</td>
</tr>
</tbody>
</table>

IP host names in variables.xml at the cell level

This file was found at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
     /config
        /cells
           /g5cell
              variables.xml
     /nodes
        /g5node
           /g5nodec
        /server
           /g5sr01c
              /nodeagent
```

Location of variables.xml file at the cell level

In general, for the simple act of moving a node to another MVS image, you would not normally need to modify this file. Here's why:

- All the Daemon instances in a Network Deployment configuration are copies of one another. They're not exactly identical but close. They all have the same host listen address.

- It does not matter which Daemon instance an external client uses; any one will work equally well. All Daemon instances have full knowledge of the entire cell configuration.

- The recommended best practice is to configure a Sysplex Distributor host name when constructing your Network Deployment Daemons. That way Sysplex Distributor can be used to route inbound requests to one of the Daemon instances.

In our sample configuration copy of that file, we found the following:
We changed this value to wsc1.washington.ibm.com. We did this because the Daemon on SYSC was moving to SYSA. Otherwise we would have left it unchanged.

**Please Read:**

- The reason a system-specific IP address appears there is because when this cell's Deployment Manager was first constructed, the person who built the cell (look no further than the name on this document 😊) failed to use the Sysplex Distributor host name of wscrb.washington.ibm.com. Instead, he used the host name of the SYSC system where the Deployment Manager was going to reside. A careless error on the part of the author of this document. But it will serve as a good instructional tool.

- In *every other scenario of this white paper* the Deployment Manager remained on SYSC. In this scenario the DMGR and its Daemon moved to SYSA. For that reason we had to go in and change this to wsc1.washington.ibm.com.

- In truth, we should have gone in and changed this to be a Sysplex Distributor address.

- Again, we're changing the value to wsc1.washington.ibm.com because the Daemon on SYSC is moving to SYSA.

To modify the value, do the following:

- Locate the variables.xml file at the cell level.
- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate the occurrence of the following variable:
  
  "WAS_DAEMON_protocol_iiop_daemon_listenIPAddress"

- Change the value="" property to the host name of target MVS image. In this scenario we changed the value from wsc3.washington.ibm.com to wsc1.washington.ibm.com

- Save the file.

**IP host names in serverindex.xml at the node level**

It's important to understand the structure of this file. This file provides information about the ports (or "end points") in the node. The file can be mapped out like this:

```xml
<serverindex ... node host address>
  <serverEntries>
    <Port type, number and host address
    <Port type, number and host address
    etc.
  </serverEntries>
  <serverEntries>
    <Port type, number and host address
    <Port type, number and host address
    etc.
  </serverEntries>
  : :
</serverindex>
```

*Schematic layout of the serverindex.xml file*
Update the Deployment Manager node's serverindex.xml

☐ Locate the serverindex.xml file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
   /config
    /cells
     /g5cell
      /nodes
       /g5node
serverindex.xml
```

Location of DMGR node's serverindex.xml file

☐ Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
☐ Locate all the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image. There'll be four for the Deployment Managers server, and one on the very first line of the file.
☐ Take a close look at the ORB and ORB_SSL ports for each server. Check the host= value on each.
  • If the value is host="*" then do nothing
  • If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.
☐ Save the file.

Update the application server node serverindex.xml

☐ Locate the serverindex.xml file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
   /config
    /cells
     /g5cell
      /nodes
       /g5nodec
serverindex.xml
```

Location of application server node's serverindex.xml file

☐ Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
☐ Locate all the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.
In our sample configuration there were two <serverEntries> blocks in the file -- one for the G5SR01C server and one for the Node Agent.

- Take a close look at the ORB and ORB_SSL ports for each server. Check the host= value on each.
  - If the value is host="*" then do nothing
  - If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

- Save the file.

**IP host addresses in server.xml at the server level**

The chances are good you will not have to modify this file. The only reference to an IP address is found on the HTTP port definitions, and in many cases the references will be null:

```
"EndPoint_1" host="" port="15548"
```

**Potential location of IP address in the server.xml file**

If the HTTP port (or SSL port -- a separate XML definition in the file) have a dotted-decimal address in that field, it means the ports are homed to a specific adapter. If that’s the case, then you have two choices:

- If the value is host="" then do nothing
- If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

If you need to change the file, do the following:

**For the Deployment Manager node**

- Locate the server.xml file for the Deployment Manager server
- Change the HTTP port host="" values as appropriate

**Note:** There'll be two in there -- one for the non-SSL HTTP port, one for the SSL port.

- Save the file
For the application server node
- Locate the server.xml file for each of the application servers in the node

  Note: Node Agents do not have HTTP ports, so you do not need to worry about it.

- Change the HTTP port host="" values as appropriate

  Note: There'll be two in there -- one for the non-SSL HTTP port, one for the SSL port.

- Save the file

IP host names in virtualhosts.xml at the cell level

Host names in this file are used in the definition of a "virtual host alias." They are defined at the bottom of the XML file, and look something like this:

```
"HostAlias_1088555533034" hostname="*" port="15518"/
"HostAlias_1088555533035" hostname="*" port="15519"/
"HostAlias_1088555533036" hostname="wsc3.washington.ibm.com" port="15518"/
"HostAlias_1088555533037" hostname="wsc3.washington.ibm.com" port="15519"/
"HostAlias_1088555533038" hostname="*" port="15538"/
"HostAlias_1088555533039" hostname="*" port="15539"/
"HostAlias_1088555533040" hostname="*" port="15548"/
"HostAlias_1088555533041" hostname="*" port="15549"/
"HostAlias_1088555533042" hostname="*" port="15558"/
"HostAlias_1088555533043" hostname="*" port="15559"/
"HostAlias_1094597555644" hostname="*" port="15800"/
"HostAlias_1094597584468" hostname="*" port="15443"/
```

Host alias values coded inside the virtualhosts.xml file

See the two aliases with a specific IP host name? That's for the Deployment Manager. When the Deployment Manager is built, by default four aliases for the DMGR HTTP ports will be built: two with asterisk, and two with the actual host name. What that means is that only people who have (in this case) wsc3.washington.ibm.com on their URL can access the Admin Console. When the alias has an asterisk, then any host name on the URL will be permitted to run applications behind the port.

Note: When asterisk and specific host name alias exists, then the specific reference takes precedence.

Do you need to modify this file? Probably not, but it depends:

- If the aliases are asterisks, then no change is necessary

- If the aliases are specific, but the alias is that of a front-end load balancing device (rather than a specific MVS image host name), then no change is necessary

Note: A front-end balancing device's host name will not change when a node in WebSphere is moved. So the user's URL will still contain the same host value. Therefore, the alias will still work. You may need to update the balancing device to understand that the node's servers have in fact moved.

- If the aliases are specific, and specific to the MVS image where the note initially resides, then you'll need to change the references to the new MVS image's host name.

Note: For example, if you move the Deployment Manager, then you'll have to change the alias. Otherwise, you won't be able to access the Deployment Manager.
For this scenario we did modify this file. That's because we moved the Deployment Manager. Therefore, do the following:

- Locate the virtualhosts.xml file. It's at the cell level:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell

Mount Point
```

**Location of cell-level virtualhosts.xml file, down under DMGR's "Master Configuration"**

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.
- Save the file.

**Regenerate the Deployment Manager's was.env file with bbodc2ns.sh shell script**

The Deployment Manager's was.env file needs to be regenerated to pick up the changes you've made in the XML files. The was.env files for the servers in the application server node will be updated when you run syncNode.sh. But we can't synchronize the Deployment Manager with itself, so we have a different shell script that's used to regenerate the was.env for it.

The bbodc2ns.sh shell script will do two important things:
- It rebuilds the was.env file for the Deployment Manager server, and
- It rebuilds the was.env file for the Daemon server

The bbodc2ns.sh file is located in the /bin directory of the Deployment Manager node configuration:

```
/wasv51config/g5cell
  /DeploymentManager
    /bin
      bbodc2ns.sh

Mount Point
```

**Location of the bbodc2ns.sh shell script**

To regenerate the was.env, do the following:
- Open up a telnet session, or OMVS session, and go to the /bin directory of the Deployment Manager node configuration being moved
- "su" (switch user) to the WebSphere Admin ID (in our case, G5ADMIN)

**Note:** You could use a UID=0 ID, but don't. That might leave file ownership and permissions in a funny state. Better to use the WebSphere Admin ID.

- Invoke the shell script:

```
./bbodc2ns.sh
```
Look for the indication of success:

BBOJ0056I Transformer Processing Complete, RC=0

**Start the Deployment Manager on the target MVS image**

With the Deployment Manager’s XML files updated and the `was.env` regenerated, it should now start on the new target MVS image. The next step -- running `syncNode.sh` -- depends on the Deployment Manager being up to work.

**Note:** The start command for the Deployment Manager is exactly the same. We haven't changed the proc name, the JOBNAME or the `ENV=` string.

Start the Deployment Manager on the new target MVS image

**Run `syncNode.sh` to synchronize changes and rebuild node’s `was.env` files**

The `syncNode.sh` shell script will do two important things:

- Take the changes you just made in the XML files under the Deployment Manager's "Master Configuration" and pull them out to the application server node's configuration directory structure.

**Note:** Yes, indeed. Synchronization -- even when initiated from the Admin Console -- is a pull operation. The node pulls the changes down to it; the DMGR never pushes them out.

- It rebuilds the `was.env` file for all the servers in the node, including the Daemon server.

**Note:** Although that's less important in this scenario because we already did that with the running of `bbodc2ns.sh`. Since the Deployment Manager and the application server node share the same Daemon, either thing -- `bbodc2ns.sh` or `syncNode.sh` -- will result in the same thing.

What's important to understand is that the `syncNode.sh` file is located in the `/bin` directory of the application server's node configuration, not the Master Configuration:

```
/wasv51config/g5cell
  /AppServerNodeC
    /bin
      syncNode.sh
```

**Location of the `syncNode.sh` shell script**

To synchronize the node, do the following:

- Make sure Deployment Manager is up.
- Make sure servers in the application server node -- particularly the Node Agent -- are down.
- Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
- If security is enabled for the Deployment Manager, understand what the administrative userid and password is for the cell.
- Open up a telnet session, or OMVS session, and go to the `/bin` directory of the application server node configuration being moved.
- "Switch users" to the WebSphere Admin ID.
Here's the command we entered to initiate the synchronization:

```bash
./syncNode.sh wsc1.washington.ibm.com 15510 -conntype SOAP -user g5admin -password xxxxxx
```

**Command used to synchronize node with changes made in Master Configuration**

**Notes:**
- The use of `wsc1.washington.ibm.com` is unique to this scenario. The DMGR moved from SYSC to SYSA. In all other scenarios the DMGR stayed on SYSC.
- In our environment security was enabled.

**Look for the messages that indicate success:**

- **ADMU0401I:** Begin syncNode operation for node g5noded with Deployment Manager `wsc1.washington.ibm.com`: 15510
- **ADMU0016I:** Synchronizing configuration between node and cell.
- **ADMU0402I:** The configuration for node g5nodec has been synchronized with Deployment Manager `wsc1.washington.ibm.com`: 15510

**Start servers**

The servers are now ready to start on the new target MVS image.

**Note:** The start commands for the servers are exactly the same. We haven't changed the proc names, the JOBNAMEs or the ENV= strings.

**Start the servers on the new MVS image.**

**Task completed**

Your two nodes -- Deployment Manager node and application server node -- should now be running on the new MVS image:

**Scenario 2 -- application server node and DMGR on SYSC moved to SYSA**
Scenario 3 in Detail

This scenario involved the "splitting away" of the application server node from the Deployment Manager node on SYSC and moving it to another MVS image. In this scenario we moved it to SYSA, which had no other part of this cell defined on it. That meant there was no Daemon server there to support the application server node. And that meant we had to create a new Daemon instance on SYSA.

This scenario shares some common things with the next scenario. Both involved the moving of the application server node away from the Daemon that supported it originally. In this scenario we created a new Daemon instance on SYSA to support the node; in the next scenario we made use of the already-created Daemon instance on SYSD.

Recall the scenario:

```
Scenario 3 -- application server node on SYSC moved to SYSA; new Daemon instance created
```

**Stop the servers in the application server node**

- This includes the Node Agent and all the application servers in the node.

  **Note:** Leave the Deployment Manager running.

**Copy Daemon directory structure and update ownership/permissions**

A Daemon has a directory structure separate from /DeploymentManager and /AppServer. If you map out the structure beneath /Daemon, you'll find it slightly different from the others. For our sample configuration, it looked like this:
A Daemon's configuration directory structure

Notes:
- A Daemon has no XML files. It has only a single was.env file.
  - We constructed our cell, we used a single shared HFS. That's why you see /SYSCE and /SYSD down under /G5CELL. If we had used a separate HFS for each node, we would not see all the system names like we do here.
  - The directory in which the was.env resides appears to be equal to the system name, but that's not precisely correct. It's actually the Daemon "instance" name.
  - A Base Application Server node's Daemon has a slightly different structure. There, rather than "System Name" for a directory, it uses the "JOBNAME" value.

For this scenario we had to create a new Daemon on SYSA. That involved copying one of the existing Daemon's directories to form a new directory.

Copy Daemon directory
- Copy the Daemon directory as illustrated in this picture (this is what we did):

Copying the SYSC Daemon structure to create a new Daemon structure

Notes:
- Make certain the system name directory -- /SYSA -- is equal to the actual system name where the node and its new Daemon will reside.
  - Back under "The mount point name is maintained in the configuration XML files" on page 15, we wrote of the importance of maintaining the same mount point name when you copy or move configuration information. Here's a good example. Even though we created a whole new directory, it's still under the /wasv51config/g5cell mount point. In this case it was in the same HFS. We could have created the directory in a different HFS on a different MVS image, provided the mount point name for that HFS was the same.
It is possible to change the mount point information in the configuration files. See "Changing the mount point information in node-level variables.xml" on page 77.

Fix ownership and permissions

- Set the new directory and file ownership to that of your "WebSphere Admin ID"
- Set the new directory so the group owner is the "WebSphere Config Group"
- Set the new directory and file permissions to 770.

Change system names

References to the system name is kept in two XML files:
- The variables.xml file at the node level
- The server.xml file at the server level

Remember that we're modifying the copies of these files in the "Master Configuration" under the Deployment Manager.

But first, we had to do a little maintenance on the Daemon we just copied.

Background on "specific UUID" value found in Daemon was.env

In a multi-system Network Deployment configuration cell, there exists a Daemon per MVS image over which the cell spans. The Daemons are very closely related to one another. They are almost -- but not exactly -- copies of one another. One of the definitions within the Daemon servers is something called the "UUID" value. It's a long hex string.

There are really two different UUID values:
- A generic UUID value -- this is the same for Daemons in a Network Deployment cell.
- A specific UUID value -- this is unique for each Daemon in the cell. Any new Daemon that's created must have a unique "specific UUID" value.

And the way WebSphere maintains the uniqueness of the "specific UUID" values is to increment the last byte of the long hex string by 1 for each node that's federated from another MVS image:
Upon federation, a new Daemon is created with a new "specific UUID" value

Notes:

1. When a Base Application Server node on the same MVS image as the DMGR is federated, the newly federated node is instructed to make use of the existing DMGR Daemon. The BaseApp's Daemon is "set aside" -- not deleted, but no longer used. Since no new Daemon is created, no new "specific UUID" is needed.

2. However, when a Base Application Server node on another MVS image from the DMGR is federated, a new Daemon is required. The BaseApp's Daemon is also "set aside" -- not deleted, but no longer used.

3. During the act of federation, a new Daemon instance is created on the MVS image where the newly federated node resides. This new Daemon instance is a close-copy of the Daemon created for the Deployment Manager. The "generic UUID" inside the new Daemon is the same as the DMGR's Daemon. The "specific UUID" is unique. And if this is the first node from another MVS image federated into the DMGR cell, the specific UUID value will be: original value + 1

Key message: we created new Daemon instance; we need unique "specific UUID" value

As the heading indicates, since we have a new Daemon instance, we need to supply a unique "specific UUID" value. It's a simple process -- we need only add one to the highest value specific UUID in the cell.

But where do we find the highest value specific UUID in the cell?

It'll be in the last "other MVS image" node federated into the Deployment Manager cell.

Note: The "other MVS image" part of that is key. Nodes on the same MVS image as the DMGR use the DMGR's Daemon. We're only interested in last node federated from some other MVS image.

But you may not recall which was the last. So instead we'll calculate the next value to use by determining the DMGR's Daemon "specific UUID" value and then using this equation:

DMGR's Daemon "specific UUID": BB6D33042266E8450000164000000809521847

+ Number of "other MVS image" nodes federated: 1

+ Increment by 1: 1

= Next "specific UUID" to use: BB6D33042266E8450000164000000809521849

Calculating the next "specific UUID" value to use for new Daemon instance
With that in mind, follow the instructions provided next.

**Capture "specific UUID" value of Daemon that supports Deployment Manager**

This step is needed so when we update the variables.xml file (the next step), we can provide a new, unique "specific UUID" value for the new Daemon we just created.

**Calculating a unique UUID for the new Daemon**

- Browse the Daemon instance was.env for the MVS image where the Deployment Manager resides. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
    /Daemon
        /config
            /cells
                /g5cell
                    /G5CELL
                        /G5CELL
                            /SYSC
```

*Location of Daemon was.env for SYSC -- where our DMGR resided*

*Note:* The file is in EBCDIC in the HFS, so you can use ISHELL or OMVS and OBROWSE.

- Scroll to the bottom and locate the following variable:

```
server_generic_short_name=G5CELL
server_generic_uuid=BB6D36776B5EE602000001640000000809521847
server_specific_short_name=SYSD
server_specific_uuid=BB6D33042266E845000001640000000809521847
```

*Note "specific" as part of the variable name*

*Hex string to capture*

**Example of "specific UUID" value for Daemon server, as found in was.env**

*Note:* For the Deployment Manager Daemon the "generic" and "specific" UUID values **look** the same, but on closer inspection you'll see they're **not exactly the same**.

- Take note of the last two characters ... in our example 47.
- Count up the number of nodes **on other MVS images** you federated into the Deployment Manager cell to create the Network Deployment configuration.
- Add that value from the previous bullet to the last two characters of the DMGR Daemon's specific UUID number.
- Add one to that number -- that'll be the "specific UUID" of the new Daemon.

*Note:* It's a hex value, so 47 .. 48 .. 49 .. 4A .. 4B .. etc.

- Write the number down and keep it handy for the next step.

New "last two characters" of Daemon "specific UUID" value: _____________
Change system name reference in the node-level variables.xml file

- Locate the variables.xml file for the node being moved. In this case it was G5NODEC. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
        /nodes
          /g5nodec
            variables.xml
```

**Location of appserver node's variables.xml file, down under DMGR's "Master Configuration"**

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Perform a global change, changing all the instances of the "old" system name to the "new" system name. We changed six instances in our copy of the file:

```
"WAS_DAEMON_daemon_start_command_args"
  value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL.SYSA"

"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSA"

"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSA"

"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSA"

"WAS_DAEMON_daemon_was_env_file"
  value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSA/was.env"

"WAS_DAEMON_daemonInstanceName" value="SYSA"
```

**See Notes**

**Different instances of the system name in the node-level variables.xml file**

**Notes:**
- There is more in this file than the lines shown here.
- The lines shown above are not contiguous. Almost, but not entirely so.
- Unlike in Scenarios 1, 2 and 5 where there was an option to change all the values or just one value in the file, here you must change all. That's because we need to tell this node about using an entirely new Daemon.

- Locate the variable in the file called:

```
"WAS_DAEMON_ONLY_server_specific_uuid"
```

and modify the last two characters to be equal to the value you calculated back on page 45.

**Note:** The key is making this value unique in the cell. Also, make sure the ending double-quote is there.

- Save the file.

Change system name reference in application server's server.xml file

- Locate the server.xml file for the application server in the node being moved. In our sample configuration, the file was located at the following location:
Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate and change one variable. The variable we’re interested in is:

name="was.ConfiguredSystemName" value="SYSC"

Change the value="SYSD" to the system name of where the node will be moved:

name="was.ConfiguredSystemName" value="SYSA"

Save the file.

Note: Repeat process for every application server in the node.

**Change system name reference in the Node Agent’s server.xml file**

Locate the server.xml file for the Node Agent in the node being moved. In our sample configuration, the file was located at the following location:
Change the value="SYSD" to the system name of where the node will be moved:

```xml
name="was.ConfiguredSystemName" value="SYSA"
```

Save the file.

**Change IP host names**

IP host names references are found in four different XML files in the configuration structure. Not all need to be changed.

<table>
<thead>
<tr>
<th>File</th>
<th>Level</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables.xml</td>
<td>cell</td>
<td>Pointer to the host name used by the Daemon instances for this cell. We didn't change this. See &quot;IP host names in variables.xml at the cell level&quot; on page 48 for an explanation why.</td>
</tr>
<tr>
<td>serverindex.xml</td>
<td>node</td>
<td>IP host names used by the various ports for the servers in the node. You will change these.</td>
</tr>
<tr>
<td>server.xml</td>
<td>server</td>
<td>Potential reference to a IP address, but only if an HTTP port is &quot;homed&quot; to a particular adapter. If no HTTP ports are &quot;homed&quot; like that, then you don't need to change anything.</td>
</tr>
<tr>
<td>virtualhosts.xml</td>
<td>cell</td>
<td>Virtual host aliases have IP host names in them. See &quot;IP host names in virtualhosts.xml at the cell level&quot; on page 51 for a discussion on whether you need to change these.</td>
</tr>
</tbody>
</table>

**IP host names in variables.xml at the cell level**

This file was found at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
  /config
  /cells
    /g5cell
      /nodes
        /g5nodec
        /servers
          /g5sr01c
          /nodeagent
```

*Location of variables.xml file at the cell level*

For the simple act of moving a node to another MVS image, you will not need to modify this file. Here's why:

- All the Daemon instances in a Network Deployment configuration are copies of one another. They are virtually identical, including the host listen address.
- It does not matter which Daemon instance an external client uses; any one will work equally well. All Daemon instances have full knowledge of the entire cell configuration.
- The recommended best practice is to configure a Sysplex Distributor host name when constructing your Network Deployment Daemons. That way Sysplex Distributor can be used to route inbound requests to one of the Daemon instances.

In our sample configuration copy of that file, we found the following:
We left this value as it was found. We did not change it.

Please Read:

- The reason a system-specific IP address appears there is because when this cell's Deployment Manager was first constructed, the person who built the cell (look no further than the name on this document 😊) failed to use the Sysplex Distributor host name of wsccb.washington.ibm.com. Instead, he used the host name of the SYSC system where the Deployment Manager was going to reside. A careless error on the part of the author of this document. But it will serve as a good instructional tool.
- In this scenario the Deployment Manager's Daemon remained on SYSC. For that reason we left this value unchanged.
- In truth, we should have gone in and changed this to be a Sysplex Distributor address.
- Again, we left this value unchanged because the Daemon remained on SYSC.

IP host names in serverindex.xml at the node level

It's important to understand the structure of this file. This file provides information about the ports (or "end points") in the node. The file can be mapped out like this:

```
<serverindex ... node host address>
  <serverEntries>
    +Port type, number and host address
    +Port type, number and host address
    +etc.
  </serverEntries>
  <serverEntries>
    +Port type, number and host address
    +Port type, number and host address
    +etc.
  </serverEntries>
</serverindex>
```

Schematic layout of the serverindex.xml file

- Locate the serverindex.xml file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5nodec
```

Location of appserver node’s serverindex.xml file, under DMGR’s "Master Configuration"

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
Locate all the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.

Note: In our sample configuration there were two <serverEntries> blocks in the file -- one for the G5SR01C server and one for the Node Agent.

Take a close look at the ORB and ORB_SSL ports for each server. Check the host= value on each.
• If the value is host="*" then do nothing
• If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

Save the file.

IP host addresses in application server server.xml at the server level

The chances are good you will not have to modify this file. The only reference to an IP address is found on the HTTP port definitions, and in many cases the references will be null:

```
"EndPoint_1" host="" port="15548"
```

Potential location of IP address in the server.xml file

If the HTTP port (or SSL port -- a separate XML definition in the file) have a dotted-decimal address in that field, it means the ports are homed to a specific adapter. If that's the case, then you have two choices:
• If the value is host="" then do nothing
• If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

If you need to change the file, do the following:

Locate the server.xml file for each application server in the node.

Note: Node Agents do not have HTTP ports, so you do not need to worry about that server.

Change the HTTP port host="" values as appropriate

Note: There'll be two in there -- one for the non-SSL HTTP port, one for the SSL port.

Save the file.
IP host names in `virtualhosts.xml` at the cell level

Host names in this file are used in the definition of a "virtual host alias." They are defined at the bottom of the XML file, and look something like this:

```
"HostAlias_1088555533034" hostname="*" port="15518"/
"HostAlias_1088555533035" hostname="*" port="15519"/
"HostAlias_1088555533036" hostname="wsc3.washington.ibm.com" port="15518"/
"HostAlias_1088555533037" hostname="wsc3.washington.ibm.com" port="15519"/
"HostAlias_1088555533038" hostname="*" port="15538"/
"HostAlias_1088555533039" hostname="*" port="15539"/
"HostAlias_1088555533040" hostname="*" port="15548"/
"HostAlias_1088555533041" hostname="*" port="15549"/
"HostAlias_1088555533042" hostname="*" port="15558"/
"HostAlias_1088555533043" hostname="*" port="15559"/
"HostAlias_1094597555644" hostname="*" port="15800"/
"HostAlias_1094597584468" hostname="*" port="15443"/
```

Host alias values coded inside the `virtualhosts.xml` file

See the two aliases with a specific IP host name? That's for the Deployment Manager. When the Deployment Manager is built, by default four aliases for the DMGR HTTP ports will be built: two with asterisk, and two with the actual host name. What that means is that only people who have (in this case) `wsc3.washington.ibm.com` on their URL can access the Admin Console. When the alias has an asterisk, then any host name on the URL will be permitted to run applications behind the port.

Note: When asterisk and specific host name alias exists, then the specific reference takes precedence.

Do you need to modify this file? Probably not, but it depends:

- If the aliases are asterisks, then no change is necessary
- If the aliases are specific, but the alias is that of a front-end load balancing device (rather than a specific MVS image host name), then no change is necessary

Note: A front-end balancing device's host name will not change when a node in WebSphere is moved. So the user's URL will still contain the same host value. Therefore, the alias will still work. You may need to update the balancing device to understand that the node's servers have in fact moved.

- If the aliases are specific, and specific to the MVS image where the note initially resides, then you'll need to change the references to the new MVS image's host name.

Note: For example, if you move the Deployment Manager, then you'll have to change the alias. Otherwise, you won't be able to access the Deployment Manager.

For this scenario we did not modify this file.

If the file does need changing, then do the following:

- Locate the `virtualhosts.xml` file. It's at the cell level:
To synchronize the node, do the following:

1. Make sure Deployment Manager is up.
2. Make sure servers in the node being moved -- particularly the Node Agent -- are down.
3. Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
4. If security is enabled for the Deployment Manager, understand what the administrative user id and password is for the cell.
5. Open up a telnet session, or OMVS session, and go to the /bin directory of the application server node configuration being moved.
"Switch users" to the WebSphere Admin ID.

Here's the command we entered to initiate the synchronization:

```bash
./syncNode.sh wsc3.washington.ibm.com 15510 -conntype SOAP -user g5admin -password xxxxxx
```

**Note:** In our environment security was enabled.

- Look for the messages that indicate success:
  
  ADMU0401I: Begin syncNode operation for node g5noded with Deployment Manager wsc3.washington.ibm.com: 15510
  
  ADMU0016I: Synchronizing configuration between node and cell.
  
  ADMU0402I: The configuration for node g5noded has been synchronized with Deployment Manager wsc3.washington.ibm.com: 15510

**Start servers**

The servers are now ready to start on the new target MVS image.

**Note:** The start commands for the servers are exactly the same. We haven't changed the proc names, the JOBNAMEs or the ENV= strings.

- Start the servers on the new MVS image.

**Note:** Make sure to start both Node Agents ... this will test to make sure there's no port contention. Look into the job output to make sure they're starting on the new target system.
Task completed

Your node should now be running on the new MVS image:

Scenario 3 -- application server node on SYSC moved to SYSA; new Daemon instance created
**Scenario 4 in Detail**

This scenario calls for the "splitting away" -- if you will -- of an application server node from the Deployment Manager and Daemon on one MVS image, and moving it to another. For this scenario we're assuming the node is being moved to an MVS image where a node *for this cell* already exists. The key there is that a Daemon instance for this cell already exists.

**Note:** A variation on this is moving the node to an MVS image where no Daemon instance for this cell already exists. That implies the creation of a new Daemon instance. That's covered in Scenario 3, starting on page 0.

So let's recall what this scenario looks like:

```
SYSA
   wsc1.washington.ibm.com

SYSC
   G5DEMN
   G5DMGR
   wsc3.washington.ibm.com

SYSD
   G5DEMN
   G5AGNTD
   G5SR02D
   G5AGNTC
   G5SR01C
   wsc4.washington.ibm.com

SYSB
   wsc2.washington.ibm.com
```

**Scenario 4 -- application server node on SYSC moved to SYSD, alongside another node**

**Please Read!** If you know your node agents are not configured with the same port values, then skip ahead to "Stop the servers in the node" on page 59.

**A challenge we faced -- you may not**

In the WP100367 white paper we -- the Washington Systems Center -- recommended that all Node Agents in a cell be configured with the same port values. There was good reason for recommending this: external clients looking to bootstrap into the namespace could do so through any Node Agent in the cell. All Node Agents are fully aware of the entire namespace. Therefore, it made good sense to configure all the Node Agents with the same ports and use Sysplex Distributor to distribute inbound IIOP requests from *external clients* across all available Node Agents. It was not a hard requirement to have the ports the same; it was just a recommendation.
When the Deployment Manager communicates with a Node Agent it does not go through Sysplex Distributor. The Deployment Manager is aware of the specific host IP names for each MVS image on which a Node Agent resides. It uses that host name, along with the port numbers it knows about, to communicate with a specific Node Agent.

Note: When the Deployment Manager communicates with a Node Agent it does not go through Sysplex Distributor. The Deployment Manager is aware of the specific host IP names for each MVS image on which a Node Agent resides. It uses that host name, along with the port numbers it knows about, to communicate with a specific Node Agent.

So our sample configuration looked like this:

```
SYSC
  G5DEMN
  G5DMGR
  G5AGNTC
  G5SR01C

SYSD
  G5DEMN
  G5AGNTD
  G5SR02D

Port Assignments

Bootstrap: 15522
DRS: 15521
Node Disc: 15524
Node Multi: 15525
ORB Listen: 15524
ORB SSL: 15523
SOAP: 15520
```

Both Node Agents had the same ports defined

The dilemma

The problem is that if we moved the application server node from SYSC to SYSD, we'd have two Node Agents on the same MVS image, contending for the same set of TCP ports:

```
SYSC
  G5DEMN
  G5DMGR
  G5AGNTC
  G5SR01C

SYSD
  G5DEMN
  G5AGNTD
  G5SR02D

Port Assignments

Bootstrap: 15522
DRS: 15521
Node Disc: 15524
Node Multi: 15525
ORB Listen: 15524
ORB SSL: 15523
SOAP: 15520
```

Port contention on the same MVS image

The result would be that it would not be possible to have both Node Agents up at the same time.
Possible solutions

We saw four possible solutions:

1. Configure multiple TCP stacks and bind one Node Agent to one and the other Node Agent to the other

   When more than one TCP stack is configured on an MVS image, the TCP ports used are separate from one another. That way one Node Agent could use port 15520 on one TCP stack, for instance, while the other used 15520 on the other TCP stack. It's a nice solution ... just more complex than we want to show in this document.

2. "Bind specific" one of the node agents to a particular adapter, and the other node agent to a different adapter.

   Again, more complex than we want to show in this document.

3. Resign ourselves to having no more than one Node Agent up at a time

   Potentially effective, but definitely awkward. This was rejected pretty quickly.

4. Reconfigure the ports for one of the two Node Agents so their ports would be different

   This defeats the benefit of Sysplex Distributor for external IIOP requests, but it allows us to accomplish the move of the node.

   **Note:** And this also permits us the opportunity to illustrate how this is done.

Decision

We opted to go with #3 -- remap the ports for one of the Node Agents and then move the node over.

The port remapping challenge

The WP100367 white paper also provides a rather elegant port allocation scheme that calls for blocks of ten TCP ports to be set aside for each server. So in our sample configuration, we set aside the ports in the following way:

<table>
<thead>
<tr>
<th>Server Type</th>
<th>Port Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daemon Servers</td>
<td>1550x range</td>
</tr>
<tr>
<td>Deployment Manager</td>
<td>1551x range</td>
</tr>
<tr>
<td><strong>Node Agents</strong></td>
<td><strong>1552x range</strong></td>
</tr>
<tr>
<td>G5SR01C</td>
<td>1553x range</td>
</tr>
<tr>
<td>G5SR02D</td>
<td>1554x range</td>
</tr>
</tbody>
</table>

So the challenge was: if we're going to remap the SYSC Node Agent's ports, what 10-port range would we give it? It would have been great to give it the 1553x range to keep it contiguous with the other Node Agent, but that range was already taken by G5SR01C.

So we opted to remap the ports into the 1555x range.

**Quick Sidestory:** Ever watch those home improvement shows where everything always works perfectly? Don't you wish just once they'd show when something goes wrong, so you could at least see how it was fixed? Well, you can't accuse this document of hiding the mistakes! 😊

Node Agent ports remapped

This process involves two steps:

1. Editing the `serverindex.xml` file and changing the port values
2. Running `syncNode.sh` to synchronize the change out of the Master Configuration into the node, and to alert all other nodes of the change.

**Edit `serverindex.xml` and change the Node Agent TCP port values**

**Note:** This editing of this file could be done concurrent with "IP host names in `serverindex.xml` at the node level" on page 64. We broke it out here to keep this issue -- duplicate ports -- separate. You may not have duplicate ports.

It’s important to understand the structure of this file. This file provides information about the ports (or "end points") in the node. The file can be mapped out like this:

```xml
<serverindex ... node host address>  
  <serverEntries>
    • Port type, number and host address
    • Port type, number and host address
    • etc.
  </serverEntries>
  <serverEntries>
    • Port type, number and host address
    • Port type, number and host address
    • etc.
  </serverEntries>
</serverindex>
```

**Schematic layout of the `serverindex.xml` file**

- Locate the `serverindex.xml` file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5nodec
```

**Location of appserver node's `serverindex.xml` file, under DMGR's "Master Configuration"**

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate the `<serverEntries>` block for the Node Agent. It'll say `serverDisplayName="nodeagent"` on that line ... that's how you can tell.

**Note:** Do not change the application server port values. There's no need.

- Update the Node Agent's port values. You can tell which port is which in the XML:
### Example of port name and port value assignments in serverindex.xml file

**Run synchNode.sh from node's /bin directory**

Rather than stepping you through this process here, we'll wait until after we've made all the other changes. Then under "Run synchNode.sh to synchronize changes and rebuild node's was.env files" on page 67 we'll actually perform the process.

**Stop the servers in the node**

This includes the Node Agent and all the application servers in the node.

**Note:** But leave the Deployment Manager up.

### Change system names

References to the system name is kept in two XML files:

- The variables.xml file at the node level
- The server.xml file at the server level

Remember that we're modifying the copies of these files in the "Master Configuration" under the Deployment Manager.

### Capture target MVS image Daemon "specific UUID" value

**Note:** We did something similar in Scenario 3, back on page 43. But there we had a different reason for doing it. When considering the totality of this white paper, please don't consider the "specific UUID" things done in Scenario 3 and Scenario 4 to be the same. They weren't.

This step is needed so when we update the variables.xml file (the next step), we have a key piece of information available so synchNode.sh doesn't "break" the Daemon on the target MVS image.

**The short story -- steps required to capture Daemon specific UUID value**

- Browse the target MVS image's Daemon instance was.env for this cell. In our sample configuration, the file was located at the following location:
Location of Daemon was.env for SYSD

Note: The file is in EBCDIC in the HFS, so you can use ISHELL or OMVS and OBROWSE.

- Scroll to the bottom and locate the following variable:

```plaintext
:  
  server_generic_short_name=G5CELL
  server_generic_uuid=BB6D36776B5EE602000001640000000809521847
  server_specific_short_name=SYSD
  server_specific_uuid=BC7BCA2DE935221C0000015C0000000809521848
```

- Hex string to capture

Example of “specific UUID” value for Daemon server, as found in was.env

- Copy that hex string to the clipboard, or paste into Notepad so you can use it during the next step.

The longer story -- why we need to do this

When a Network Deployment cell configuration spans MVS images, there'll be a Daemon server "instance" for every MVS image included in the cell. In our sample configuration the cell spanned SYSC and SYSD, so we had two Daemon server instances.

All the Daemons are very similar to one another, but there are some unique things about each:

- Each has its own directory structure where its own was.env resides
- Each has its own "instance name," which was set equal to the system name when the Daemon was first constructed
- Each has a unique "specific UUID" value

This scenario is unique in that rather than moving the node and the Daemon, we're splitting the node away from its original Daemon and putting it under a different, already-existing Daemon. In the next step we're going to accomplish that by doing a global change in the variables.xml file to change all SYSC strings to SYSD.

But there's a dilemma: the variables.xml file will contain the "specific UUID" value for the original SYSC Daemon server. When we run syncNode.sh to synchronize the Master Configuration change out to the node, that shell script -- syncNode.sh -- will...
read the stuff inside variables.xml and then go update the was.env for whatever Daemon is pointed to in the file.

And there's the dilemma: when we did the global change of SYSC to SYSD, we changed the pointer so it no longer pointed at the Daemon on SYSC, but rather started pointed to the Daemon on SYSD. So when syncNode.sh updates the Daemon's was.env, it'll update the SYSD Daemon.

Note: If there was a switch that told syncNode.sh not to update the Daemon, we'd use it here to avoid this "copy-paste" of the specific UUID value. But to my knowledge there is no such switch. So we did what's described here.

If we did not set the "specific UUID" value inside the variables.xml file, the value that'd be in there is the hex string for the SYSC Daemon server. When syncNode.sh runs it would update the SYSD Daemon and change its specific UUID value to be the same as SYSC. That's not good -- all Daemons need a unique "specific UUID" value.

To fix that, what we'll do is go in ahead of time and set the "specific UUID" value in the variables.xml file for the node being moved to be equal to the Daemon's on the target system. That way, when syncNode.sh runs it'll still update the was.env file, and it'll still update the "specific UUID" value, but it'll replace "X" with "X" so everything stays as it was before.

Got that?

Remember, we didn't claim this was a pretty scenario.

Change system name reference in the node-level variables.xml file

- Locate the variables.xml file for the node being moved. In this case it was G5NODEC. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5nodec
```

Location of appserver node's variables.xml file, down under DMGR's "Master Configuration"

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Perform a global change, changing all the instances of the "old" system name to the "new" system name. We changed six instances in our copy of the file:

```
"WAS_DAEMON_daemon_start_command_args"
  value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL_SYSD"

"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSD"

"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSD"

"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSD"

"WAS_DAEMON_daemon_was_env_file"
  value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSD/was.env"

"WAS_DAEMON_daemonInstanceName" value="SYSD"
```

Different instances of the system name in the node-level variables.xml file

© 2005, IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD

Section: Scenario 4 Detail
Version Date: Saturday, February 12, 2005
There is more in this file than the lines shown here.
The lines shown above are not contiguous. Almost, but not entirely so.
Unlike in Scenarios 1, 2 and 5 where there was an option to change all the values or
just one value in the file, here you must change all. That's because we need to tell
this node about using a different Daemon.

- Locate the variable in the file called:
  "WAS_DAEMON_ONLY_server_specific_uuid"
and replace the long hex string found there with the hex string captured back in "The
short story -- steps required to capture Daemon specific UUID value" on page 59. (In
other words, replace the specific UUID in the XML file with the specific UUID captured
from the target image's Daemon was.env file.)

  **Note:** Please make certain the hex string is enclosed in double quotes as shown.

- Save the file.

**Change system name reference in application server's server.xml file**

- Locate the server.xml file for the application server in the node being moved. In our
  sample configuration, the file was located at the following location:

  ```
  /wasv51config/g5cell
  /DeploymentManager
  /config
  /cells
  /g5cell
  /nodes
  /g5nodec
  /servers
  /g5sr01c
  server.xml
  /nodeagent
  ```

  "Location of appserver's server.xml file, down under DMGR's "Master Configuration"

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate and change one variable. The variable we're interested in is:
  name="was.ConfiguredSystemName" value="SYSC"
- Change the value="SYSD" to the system name of where the node will be moved:
  name="was.ConfiguredSystemName" value="SYSD"
- Save the file.

  **Note:** Repeat process for every application server in the node.

**Change system name reference in the Node Agent's server.xml file**

- Locate the server.xml file for the Node Agent in the node being moved. In our
  sample configuration, the file was located at the following location:
Location of the node agent's `server.xml` file, down under DMGR's "Master Configuration"

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate and change one variable. The variable we're interested in is:
  ```xml
  name="was.ConfiguredSystemName" value="SYSC"
  ```
- Change the value="SYSD" to the system name of where the node will be moved:
  ```xml
  name="was.ConfiguredSystemName" value="SYSD"
  ```
- Save the file.

**Change IP host names**

IP host names references are found in four different XML files in the configuration structure. Not all need to be changed.

<table>
<thead>
<tr>
<th>File</th>
<th>Level</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables.xml</td>
<td>cell</td>
<td>Pointer to the host name used by the Daemon instances for this cell. We didn't change this. See &quot;IP host names in variables.xml at the cell level&quot; on page 64 for an explanation why.</td>
</tr>
<tr>
<td>serverindex.xml</td>
<td>node</td>
<td>IP host names used by the various ports for the servers in the node. You will change these.</td>
</tr>
<tr>
<td>server.xml</td>
<td>server</td>
<td>Potential reference to a IP address, but only if an HTTP port is &quot;homed&quot; to a particular adapter. If no HTTP ports are &quot;homed&quot; like that, then you don't need to change anything.</td>
</tr>
<tr>
<td>virtualhosts.xml</td>
<td>cell</td>
<td>Virtual host aliases have IP host names in them. See &quot;IP host names in virtualhosts.xml at the cell level&quot; on page 66 for a discussion on whether you need to change these.</td>
</tr>
</tbody>
</table>
IP host names in `variables.xml` at the cell level

This file was found at the following location:

```
/wasv51config/g5cell
  /DeploymentManager
    /config
      /cells
        /g5cell
          /nodes
            /g5nodec
              /servers
                /g5sr01c
                /nodeagent
```

Location of `variables.xml` file at the cell level

For the simple act of moving a node to another MVS image, you will *not* need to modify this file. Here’s why:

- All the Daemon *instances* in a Network Deployment configuration are copies of one another. They are virtually identical, including the host listen address.

- It does not matter which Daemon instance an external client uses; any one will work equally well. All Daemon instances have full knowledge of the entire cell configuration.

- The recommended best practice is to configure a Sysplex Distributor host name when constructing your Network Deployment Daemons. That way Sysplex Distributor can be used to route inbound requests to one of the Daemon instances.

In our sample configuration copy of that file, we found the following:

```
"WAS_DAEMON_protocol_iioop_daemon_listenIPAddress"
  value="wsc3.washington.ibm.com"
```

We left this value as it was found. We did not change it.

**Please Read:**
- The reason a system-specific IP address appears there is because when this cell’s Deployment Manager was first constructed, the person who built the cell (look no further than the name on this document 😊) failed to use the Sysplex Distributor host name of `wsccb.washington.ibm.com`. Instead, he used the host name of the SYSC system where the Deployment Manager was going to reside. A careless error on the part of the author of this document. But it will serve as a good instructional tool.

- In *this scenario* the Deployment Manager's Daemon *remained on SYSC*. For that reason we left this value *unchanged*.

- In truth, we should have gone in and changed this to be a Sysplex Distributor address.

- Again, we left this value unchanged because the Daemon remained on SYSC.

IP host names in `serverindex.xml` at the node level

It’s important to understand the structure of this file. This file provides information about the ports (or "end points") in the node. The file can be mapped out like this:
Locate the serverindex.xml file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
   /DeploymentManager
      /config
         /cells
            /g5cell
                /nodes
                   /g5nodec
                      serverindex.xml
```

Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate all the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.

**Note:** In our sample configuration there were two <serverEntries> blocks in the file -- one for the G5SR01C server and one for the Node Agent.

Take a close look at the ORB and ORB_SSL ports for each server. Check the host= value on each.

- If the value is host="*" then do nothing
- If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

Save the file.

**IP host addresses in application server's server.xml at the server level**

The chances are good you will not have to modify this file. The only reference to an IP address is found on the HTTP port definitions, and in many cases the references will be null:
Null value means port is not bound to any particular adapter

"EndPoint_1" host="" port="15548"

**Potential location of IP address in the server.xml file**

If the HTTP port (or SSL port -- a separate XML definition in the file) have a dotted-decimal address in that field, it means the ports are homed to a specific adapter. If that's the case, then you have two choices:

- If the value is `host=""` then do nothing
- If the value is `host="<some IP address>"` then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

If you need to change the file, do the following:

- Locate the `server.xml` file for each server in the node. Its location is the exact same as we covered back on page 62.
  
  **Note:** Node Agents do not have HTTP ports, so you do not need to worry about that server.

- Change the HTTP port `host=""` values as appropriate
  
  **Note:** There'll be two in there -- one for the non-SSL HTTP port, one for the SSL port.

- Save the file.

**IP host names in virtualhosts.xml at the cell level**

Host names in this file are used in the definition of a "virtual host alias." They are defined at the bottom of the XML file, and look something like this:

```
"HostAlias_1088555533034" hostname="*" port="15518"/>
"HostAlias_1088555533035" hostname="*" port="15519"/>
"HostAlias_1088555533036" hostname="wsc3.washington.ibm.com" port="15518"/>
"HostAlias_1088555533037" hostname="wsc3.washington.ibm.com" port="15519"/>
"HostAlias_1088555533038" hostname="*" port="15538"/>
"HostAlias_1088555533039" hostname="*" port="15539"/>
"HostAlias_1088555533040" hostname="*" port="15540"/>
"HostAlias_1088555533041" hostname="*" port="15541"/>
"HostAlias_1088555533042" hostname="*" port="15542"/>
"HostAlias_1088555533043" hostname="*" port="15543"/>
"HostAlias_1088555533044" hostname="15800"/>
"HostAlias_1088555533045" hostname="15443"/>
```

**Host alias values coded inside the virtualhosts.xml file**

See the two aliases with a specific IP host name? That's for the Deployment Manager. When the Deployment Manager is built, by default four aliases for the DMGR HTTP ports will be built: two with asterisk, and two with the actual host name. What that means is that
only people who have (in this case) wsc3.washington.ibm.com on their URL can access the Admin Console. When the alias has an asterisk, then any host name on the URL will be permitted to run applications behind the port.

**Note:** When asterisk and specific host name alias exists, then the specific reference takes precedence.

Do you need to modify this file? Probably not, but it depends:

- If the aliases are asterisks, then no change is necessary
- If the aliases are specific, but the alias is that of a front-end load balancing device (rather than a specific MVS image host name), then no change is necessary

**Note:** A front-end balancing device's host name will not change when a node in WebSphere is moved. So the user's URL will still contain the same host value. Therefore, the alias will still work. You may need to update the balancing device to understand that the node's servers have in fact moved.

- If the aliases are specific, and specific to the MVS image where the note initially resides, then you'll need to change the references to the new MVS image's host name.

**Note:** For example, if you move the Deployment Manager, then you'll have to change the alias. Otherwise, you won't be able to access the Deployment Manager.

For this scenario we did *not* modify this file.

If the file does need changing, then do the following:

- Locate the virtualhosts.xml file. It's at the cell level:

  ```plaintext
  /wsv51config/g5cell
  /DeploymentManager
  /config
  /cells
  /g5cell
  virtualhosts.xml
  ```

  *Location of cell-level virtualhosts.xml file, down under DMGR's "Master Configuration"*

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image.
- Save the file.

**Make certain the Deployment Manager is up and running**

- The next step -- running `syncNode.sh` -- depends on the Deployment Manager being up to work.

**Run `syncNode.sh` to synchronize changes and rebuild node's `was.env` files**

The `syncNode.sh` shell script will do two important things:

- Take the changes you just made in the XML files under the Deployment Manager's "Master Configuration" and pull them out to the application server node's configuration directory structure.
Yes, indeed. Synchronization -- even when initiated from the Admin Console -- is a pull operation. The node pulls the changes down to it; the DMGR never pushes them out.

- It rebuilds the was.env file for all the servers in the node.

What's important to understand is that the syncNode.sh file is located in the /bin directory of the application server's node configuration, not the Master Configuration:

```
/wasv51config/g5cell
/AppServerNodeC
/bin
  syncNode.sh
```

**Location of the syncNode.sh shell script**

To synchronize the node, do the following:

- Make sure Deployment Manager is up.
- Make sure servers in the node being moved -- particularly the Node Agent -- are down.
- Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
- If security is enabled for the Deployment Manager, understand what the administrative userid and password is for the cell.
- Open up a telnet session, or OMVS session, and go to the /bin directory of the application server node configuration being moved.
- "Switch users" to the WebSphere Admin ID
- Here's the command we entered to initiate the synchronization:

```
./syncNode.sh wsc3.washington.ibm.com 15510 -conntype SOAP -user g5admin -password xxxxxx
```

**Command used to synchronize node with changes made in Master Configuration**

- Look for the messages that indicate success:
  
  ADMU0401I: Begin syncNode operation for node g5noded with Deployment Manager wsc3.washington.ibm.com: 15510
  ADMU0016I: Synchronizing configuration between node and cell.
  ADMU0402I: The configuration for node g5noded has been synchronized with Deployment Manager wsc3.washington.ibm.com: 15510

**Start servers**

The servers are now ready to start on the new target MVS image.

- The start commands for the servers are exactly the same. We haven't changed the proc names, the JOBNAMEs or the ENV= strings.
Start the servers on the new MVS image.

**Note:** Make sure to start both Node Agents ... this will test to make sure there's no port contention. Look into the job output to make sure they're starting on the new target system.

**Task completed**

Your node should now be running on the new MVS image:

**Scenario 4 -- application server node on SYSC moved to SYSD, alongside another node**
Scenario 5 in Detail

A "Base Application Server node" is WebSphere V5's most basic configuration. It consists of a single node, a single application server and a Daemon server. By definition a Base Application Server can't span MVS images; it must reside entirely on one MVS image.

In this scenario we moved a Base Application Server node from SYSC to SYSD:

- **Syntetic:** G5SR01C
- **Public:** wsc3.washington.ibm.com
- **System:** G5DEMNC
- **Remote:** wsc4.washington.ibm.com
- **Remote:** G5SR01C

**Scenario 5 -- Base Application Server node configuration moved from SYSC to SYSD**

The process is quite simple -- perhaps even more simple than Scenario 1 or Scenario 2 presented earlier. There is no "Master Configuration" to worry about because there is no Deployment Manager. Therefore there's no "synchronization" to perform. We do need to regenerate the was.env file, but that's a relatively simple process as well.

**Stop the server and the Daemon**

- The easiest way to do that is to stop the Daemon. That'll bring down the dependent server.

**Change system names**

References to the system name is kept in two XML files:

- The variables.xml file at the node level
- The server.xml files at the server level

Because this is a Base Application Server node, we don't have a "Master Configuration." So we modified the XML under the node's directory structure.

**Change system name reference in the variables.xml file**

- Locate the variables.xml file for the node being moved. In this case it was G5NODEC. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /BaseAppServer
    /config
      /cells
        /g5basec
          /nodes
            /g5nodec
```

**Location of Base Application Server node's variables.xml file**

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
Perform a global change, changing all the instances of the "old" system name to the "new" system name. We changed six instances in our copy of the file:

"WAS_DAEMON_daemon_start_command_args"
value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL.SYSD"
"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSD"
"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSD"
"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSD"
"WAS_DAEMON_daemon_was_env_file"
value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSD/was.env"
"WAS_DAEMON_daemonInstanceName" value="SYSD"

Different instances of the system name in the node-level variables.xml file

Notes:
• There is more in this file than the lines shown here.
• The lines shown above are not contiguous. Almost, but not entirely so.
• An alternative is to change just the "server_configured_system_name" variable. See "The difference between changing all system name reference in variables.xml vs. just one" on page 92 for more information.

Save the file.

Change system name reference in application server's server.xml file

Locate the server.xml file for the application server in the node being moved. In our sample configuration, the file was located at the following location:

/wasv51config/g5cell
  /BaseAppServer
    /config
      /cells
        /g5basec
          /nodes
            /g5nodec
              /servers
                /g5sr01c

Mount Point
Under the node's configuration structure

Location of Base Application Server's server.xml file

Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.

Locate and change one variable. The variable we're interested in is:

name="was.ConfiguredSystemName" value="SYSC"

Change the value="SYSD" to the system name of where the node will be moved:

name="was.ConfiguredSystemName" value="SYSD"

Save the file.
**Change IP host names**

IP host names references are found in four different XML files in the configuration structure:

<table>
<thead>
<tr>
<th>File</th>
<th>Level</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables.xml</td>
<td>cell</td>
<td>Pointer to the host name used by the Daemon instance for this cell.</td>
</tr>
<tr>
<td>serverindex.xml</td>
<td>node</td>
<td>IP host names used by the various ports for the servers in the node.</td>
</tr>
<tr>
<td>server.xml</td>
<td>server</td>
<td>Potential reference to an IP address, but only if an HTTP port is “homed” to a particular adapter. If no HTTP ports are “homed” like that, then you don’t need to change anything.</td>
</tr>
<tr>
<td>virtualhosts.xml</td>
<td>cell</td>
<td>Virtual host aliases have IP host names in them. See &quot;IP host names in virtualhosts.xml at the cell level&quot; on page 74 for a discussion on whether you need to change these.</td>
</tr>
</tbody>
</table>

**IP host names in variables.xml at the cell level**

Our approach to this file was quite a bit different in Scenario 1 through 4 -- there we had a Network Deployment configuration with multiple Daemon instances.

**Note:**  See "IP host names in variables.xml at the cell level" on page 23 for more on Daemon instances in an ND configuration.

For a Base Application Server node, however, there will be only one Daemon. And if this BaseApp server was ever federated into a Deployment Manager cell, the BaseApp’s Daemon would **cease being used**.

Therefore, it’s very likely that when configuring the Base Application Server node the host value for the Daemon was a system-specific IP host name as opposed to a Sysplex Distributor host name. Therefore, if the Daemon is being moved to another MVS image, that value must be changed in the variables.xml file.

Do the following:

- Locate the variables.xml file at the cell level. In our sample scenario it was located at the following location:

```
/wasv51config/g5cell
  /BaseAppServer
    /config
      /cells
        /g5basec
```

**Location of variables.xml file at the cell level**

- Locate the instance of the variable:

  "WAS_DAEMON_protocol_iip daemon listenIPAddress"

- Change value= to the host name of the target MVS image. In our scenario that implied a change from wsc3.washington.ibm.com to wsc4.washington.ibm.com.

- Save the file.
IP host names in serverindex.xml at the node level

It's important to understand the structure of this file. This file provides information about the ports (or "end points") in the node. The file can be mapped out like this:

```
<serverindex ... node host address>
  <serverEntries>
    - Port type, number and host address
    - Port type, number and host address
    etc.
  </serverEntries>
</serverindex>
```

A Base Application Server node's serverindex.xml will have only one <serverEntries> block. That'll define the ports for the one server in the node.

**Schematic layout of the serverindex.xml file**

- Locate the serverindex.xml file for the node being moved. In our sample configuration, the file was located at the following location:

```
/wasv51config/g5cell
  /BaseAppServer
    /config
      /cells
        /g5basec
          /nodes
            /g5nodec
```

**Location of serverindex.xml file in our Base Application Server node**

- Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
- Locate all the instances of the host name string of the original MVS image, and replace them with the host name string of the target MVS image. In our sample scenario, that meant changing them from wsc3.washington.ibm.com to wsc4.washington.ibm.com
- Take a close look at the ORB and ORB_SSL ports for each server. Check the host= value on each.
  - If the value is host="*" then do nothing
  - If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.
- Save the file.

**IP host addresses in server.xml at the server level**

The chances are good you will not have to modify this file. The only reference to an IP address is found on the HTTP port definitions, and in many cases the references will be null.
Null value means port is not bound to any particular adapter

"EndPoint_1" host="" port="15558"

**Potential location of IP address in the server.xml file**

If the HTTP port (or SSL port -- a separate XML definition in the file) have a dotted-decimal address in that field, it means the ports are homed to a specific adapter. If that's the case, then you have two choices:

- If the value is host="" then do nothing
- If the value is host="<some IP address>" then it means those ports were bound to a particular adapter. If you wish to do the same on the target MVS image, then change the address to the adapter address on the target MVS image. If you wish to change it to listen across all ports, then change it to an asterisk.

If you need to change the file, do the following:

- Locate the server.xml file for the server in the Base Application Server node. In our sample scenario the location was:

```
/wasv51config/g5cell
  /BaseAppServer
    /config
      /cells
        /g5basec
        /nodes
          /g5nodec
            /servers
              /g5sr01c
```

**Location of Base Application Server node's server.xml file**

- Change the HTTP port host="" values as appropriate
  
  **Note:** There'll be two in there -- one for the non-SSL HTTP port, one for the SSL port.

- Save the file.

**IP host names in virtualhosts.xml at the cell level**

Host names in this file are used in the definition of a "virtual host alias." They are defined at the bottom of the XML file, and look something like this:
Host alias values coded inside the virtualhosts.xml file

The aliases with the specific host name (wsc3.washington.ibm.com) were put there when the Admin Console was installed. What this does is prevent anyone from accessing the Admin Console with a host value on their browser URL of anything but wsc3.washington.ibm.com.

Note: When asterisk and specific host name alias exists -- as is the case here -- then the specific reference takes preference.

Do you need to modify this file? For a Base Application Server node, yes. In order to keep being able to access the Admin Console when the node is moved to the new MVS image, you'll need to update the virtualhost alias.

Note: Or you could remove the two aliases that explicitly point to the host name. That would leave the two with asterisks for the alias. That would then work on any MVS image.

Do the following:

☐ Locate the virtualhosts.xml file. It's at the cell level:

```
/wasv51config/g5cell
  /BaseAppServer
    /config
      /cells
        /g5basec
          virtualhosts.xml
```

Location of virtualhosts.xml file

☐ Open the file for edit. Remember that the file is in ASCII in the z/OS HFS.
☐ Scroll to the bottom where the virtual host aliases are found
☐ Change the explicit host name references to the host name of the new target MVS image.
☐ Save the file.

Regenerate the BaseApp node's was.env files with bbowc2ns.sh shell script

The bbowc2ns.sh shell script will do two important things:

- It rebuilds the was.env file for the application server, and
- It rebuilds the was.env file for the Daemon server

Note: This is a slightly different named shell script than the one used to regenerate the was.env for the Deployment Manager. Here it's bbowc2ns.sh. There it was bbodc2ns.sh.
The `bbowc2ns.sh` file is located in the `/bin` directory of the application server node configuration:

```
/wasv51config/g5cell
  /BaseAppServer
    /bin
      /bbowc2ns.sh
```

**Location of the `bbowc2ns.sh` shell script**

To regenerate the `was.env`, do the following:

- Open up a telnet session, or OMVS session, and go to the `/bin` directory of the Base Application Server node configuration being moved
- "su" (switch user) to the WebSphere Admin ID (in our case, G5ADMIN)

**Note:** You *could* use a UID=0 ID, but don't. That might leave file ownership and permissions in a funny state. Better to use the WebSphere Admin ID.

- Invoke the shell script:
  ```bash
  ./bbowc2ns.sh
  ```
- Look for the indication of success:
  ```
  BBOJ0056I Transformer Processing Complete, RC=0
  ```

**Start the application server on the target MVS image**

With the application server's XML files updated and the `was.env` regenerated, it should now start on the new target MVS image.

**Note:** The start command for the application server is exactly the same. We haven't changed the proc name, the `JOBNAME` or the `ENV=` string.

- Start the Base Application Server node server on the new target MVS image

**Task completed**

Your Base Application Server node should now be running on the new MVS image:

```
SYSC
  wsc3.washington.ibm.com

SYSD
  G5DEMNC
  G5SR01C
  wsc4.washington.ibm.com
```

Scenario 5 -- Base Application Server node configuration moved from SYSC to SYSD
Appendix A - Other Information

Changing the mount point information in node-level variables.xml

A basic assumption used in this document was that when a node was moved from one MVS image to another, the mount point on which the configuration HFS was mounted remained the same. It is not possible to simply mount the configuration HFS at a different mount point and start the servers. There are two reasons why this is not possible:

1. The configuration files for each node contain explicit references to the mount point.
   Specifically in the node-level variables.xml file and in the /bin directory's setupCmdLine.sh

   Note: Actually, there's more references than even that, but those are the ones you must change manually. The others will be automatically regenerated.

1. The JCL start procedures contain an explicit reference to the mount point.
   Specifically the SET ROOT= variable at the top of each procedure.

Thankfully it is possible to change these values, and the process is not that difficult.

Easiest thing -- maintain same mount point

If you're moving a node to another MVS image, the best and easiest thing to do is maintain the same mount point.

- If the HFS is shared across MVS images in the Sysplex, then you do not need to do anything. Leave the HFS where it is and simply start the node's servers on the other MVS image.
- If the HFS is not shared, then you have to move the HFS over to the other MVS image. If possible, insure it's on the same mount point there. Then it's simply a matter of starting the servers.

However, if for whatever reason you find you must change the mount point, then read on ...

Single HFS for all nodes?

When building a Network Deployment configuration, it's quite possible to construct it in a way where multiple nodes are contained within the same HFS. You might use one large HFS for all the nodes in a cell:

Single shared HFS with all nodes contained within it

Or perhaps you chose to have one HFS per MVS image, which still may imply multiple nodes in the same HFS:
Still find multiple nodes in this HFS, even though limited to one MVS image

HFS per MVS image, but still have multiple nodes in one of the HFS.

And while a Base Application Server node is, by definition, limited to one MVS image, there is the possibility that multiple BaseApp nodes are contained in the same HFS:

Possible to have multiple BaseApp Server nodes under the same HFS as well

**Key Point:** Mounting an HFS at a different mount point might affect multiple nodes. If you intend to update all the nodes affected, then everything is okay. But if you desire to only move one node to a different mount point, that implies copying out the node structure to another HFS. Copying out a node structure is a topic handled separately under "Scenario 3 -- copying out a node structure to a new HFS on a new mount point" on page 89.

**Common JCL in use across different nodes?**

WebSphere Application Server for z/OS Version 5.1 allows you to use a single set of procs to start different servers. This is a nice feature, but it can create other problems. The JCL has a variable called `SET ROOT=` that points to the mount point of the configuration where the server being started resides. If you move a node to a different mount point, that implies updating the JCL to reflect the new `SET ROOT=` pointer to the configuration mount point.

If you've opted to have different JCL procs per node, then you're all set -- simply update the JCL for the node being moved to a different mount point, and continue on.
But what if you have a common set of JCL used for servers across multiple nodes? In that case updating the `SET ROOT=` value will affect not only the servers you intended to affect, but other servers as well.

Now we have a dilemma. The JCL proc names used by the servers are maintained in the configuration as well. So you can't just create a new set of procs for a node. You have to tell WebSphere about the new procs.

**Note:** You could get away with simply copying the controller procs to a different member in PROCLIB, but only if you never intended to start a server from the Admin Console. Daemon procs and servant procs are more troublesome -- they're started automatically by WebSphere. So telling WebSphere about them _would_ be necessary.

My dilemma is more basic ... I have to "fence" this document somewhere or I'll never finish it. So I'm going to draw a line here -- I will assume if you're using a common set of JCL that you're also maintaining the same mount point value when you move the node. That way the same JCL can be used without modification.

**Scenario 1 -- remapping mount point for single node**

We'll use our "Scenario 1" to illustrate the process of changing the mount point for a node. Recall the scenario:

Let's settle on some basic assumptions for this illustration:

- The node originally on SYSD had its own HFS. So we _don't_ have to worry about affecting multiple nodes, as was discussed under "Single HFS for all nodes?" on page 77.
- For whatever reason the mount point for this HFS on SYSB must be different than on SYSD.
- You're familiar with all the other steps necessary to move a node to another MVS image, as detailed under "Scenario 1 in Detail" starting on page 20.

With those in mind, let's walk through the process:
Create mount point, provide permissions/ownership and mount HFS

- Create the mount point. Permissions and ownership must be as follows:

  - Directories higher than mount point may be owned by anybody, but must have "other" permissions of at least "5" (read and execute)
  - Mount point for configuration:
    - Owner: WebSphere Admin ID
    - Group: WebSphere Config Group
    - Perm: 775

Ownership and permissions for WebSphere configuration mount point

- Mount the HFS on the new mount point

Update node-level variables.xml in DMGR master configuration

- Go into the Deployment Manager's "Master Configuration"
- Locate the variables.xml file for the node being moved. This will be at the node-level. In this scenario that was at the following location:

  /wasv51config/g5cell
  /DeploymentManager
  /config
  /cells
  /g5cell
  /nodes
  /g5noded
  variables.xml

Location of node's variables.xml file, down under DMGR's "Master Configuration"

- Edit the file and change all instances of the old mount point value string to be equal to the new mount point value. You should find four instances of the string in the file.
- Save the file.

Update setupCmdLine.sh in application server node's /bin directory

- Go into the application server node's directory structure (under the new mount point)
- Go to the /bin directory
- Edit the setupCmdLine.sh file
- Update the WAS_HOME variable to reflect the new mount point.
- Save the file.
**Run `syncNode.sh` to synchronize master configuration changes out to node**

What's important to understand is that the `syncNode.sh` file is located in the `/bin` directory of the application server's node configuration, *not the Master Configuration*:

![Diagram showing the location of `syncNode.sh`]

*Location of the `syncNode.sh` shell script*

To synchronize the node, do the following:

- Make sure Deployment Manager is up.
- Make sure servers in the node being moved -- particularly the Node Agent -- are down.
- Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
- If security is enabled for the Deployment Manager, understand what the administrative userid and password is for the cell.
- Open up a telnet session, or OMVS session, and go to the `/bin` directory of the application server node configuration being moved.
- "Switch users" to the WebSphere Admin ID
- Here's the command we entered to initiate the synchronization:

```
./syncNode.sh wsc3.washington.ibm.com 15510 -conntype SOAP -user g5admin -password xxxxxx
```

*Command used to synchronize node with changes made in Master Configuration*

**Notes:**
- The command must be entered as one line.
- In our environment security was enabled.
Look for the messages that indicate success:

ADMU0401I: Begin syncNode operation for node g5noded with Deployment Manager wsc3.washington.ibm.com: 15510
ADMU0016I: Synchronizing configuration between node and cell.
ADMU0402I: The configuration for node g5noded has been synchronized with Deployment Manager wsc3.washington.ibm.com: 15510

Modify JCL start procedures to update SET Root= value

- Edit the JCL for the following servers and update SET Root= to reflect the new mount point:
  - Application server controller JCL
  - Application server servant JCL
  - Node Agent controller JCL
  - Daemon JCL

Note: Be alert regarding the Daemon JCL proc. Be certain the same proc isn't used with the other nodes in the cell. If it is, you may not be able to mount the node at a different mount point without also changing the Daemon’s JCL start proc (the reference to which is held in the node-level variables.xml, and is named on the variable WAS_DAEMON_daemon_start_command.)

Start servers on new MVS image

- Start the servers on the target MVS image, now mounted at the new mount point.

Scenario 2 -- remapping mount point for multiple nodes

We'll use our “Scenario 2” to illustrate the process of changing the mount point for a node. Recall the scenario:

Scenario 2 -- application server node and DMGR on SYSC moved to SYSA

Let's settle on some basic assumptions for this illustration:

- The nodes originally on SYSC were both in the same HFS, but no other nodes were contained in that HFS. So we don't have to worry about affecting multiple nodes beyond the two we intend to affect, as was discussed under “Single HFS for all nodes?” on page 77.
For whatever reason the mount point for this HFS on SYSA must be different than on SYSC.

You're familiar with all the other steps necessary to move a node to another MVS image, as detailed under "Scenario 2 in Detail" starting on page 29.

With those in mind, let's walk through the process:

**Create mount point, provide permissions/ownership and mount HFS**

- Create the mount point. Permissions and ownership must be as follows:

  - Directories higher than mount point may be owned by anybody, but must have "other" permissions of at least "5" (read and execute).

  - Mount point for configuration:
    - Owner: WebSphere Admin ID
    - Group: WebSphere Config Group
    - Perm: 775

  - Ownership and permissions for WebSphere configuration mount point

- Mount the HFS on the new mount point

**Update node-level variables.xml in DMGR master configuration**

- Go into the Deployment Manager's "Master Configuration"

- Locate the variables.xml file for the Deployment Manager node being moved. This will be at the node-level. In this scenario that was at the following location:

  - Location of DMGR node's variables.xml file, under DMGR's "Master Configuration"

- Edit the file and change all instances of the old mount point value string to be equal to the new mount point value. You should find four instances of the string in the file.

- Save the file.

- Locate the variables.xml file for the application server node being moved. This will be at the node-level. In this scenario that was at the following location:
Edit the file and change all instances of the old mount point value string to be equal to the new mount point value. You should find four instances of the string in the file.

Save the file.

Update `setupCmdLine.sh` in both nodes' `/bin` directory

Go into the Deployment Manager node's directory structure (under the new mount point)

Go to the `/bin` directory

Edit the `setupCmdLine.sh` file

Update the WAS_HOME variable to reflect the new mount point.

Save the file.

Go into the application server node's directory structure (under the new mount point)

Go to the `/bin` directory

Edit the `setupCmdLine.sh` file

Update the WAS_HOME variable to reflect the new mount point.

Save the file.

Run `bbodc2ns.sh` to regenerate the Deployment Manager node

Open up a telnet session, or OMVS session, and go to the `/bin` directory of the Deployment Manager node

"su" (switch user) to the WebSphere Admin ID (in our case, G5ADMIN)

Note: You could use a UID=0 ID, but don't. That might leave file ownership and permissions in a funny state. Better to use the WebSphere Admin ID.

Invoke the shell script:

```
./bbodc2ns.sh
```

Look for the indication of success:

```
BBOJ0056I Transformer Processing Complete, RC=0
```
Modify JCL start procedures to update SET \texttt{ROOT=} value

- Edit the JCL for the following servers and update \texttt{SET ROOT=} to reflect the new mount point:
  - Deployment Manager controller JCL
  - Deployment Manager servant JCL
  - Daemon JCL

\textbf{Note:} Be alert regarding the Daemon JCL proc. Be certain the same proc isn't used with the other nodes in the cell. If it is, you may not be able to mount the node at a different mount point without also changing the Daemon's JCL start proc (the reference to which is held in the node-level variables.xml, and is named on the variable \texttt{WAS_DAEMON_daemon_start_command}.)

Start the Deployment Manager

- Start the Deployment Manager.

\textbf{Run} \texttt{syncNode.sh} \textbf{to synchronize master configuration changes out to node}

What's important to understand is that the \texttt{syncNode.sh} file is located in the /\texttt{bin} directory of the application server's node configuration, \textit{not the Master Configuration}:

$\langle$\texttt{new mount}$\rangle$
$\downarrow$
/AppServerNodeC
$\downarrow$
/bin

\begin{center}
\begin{tikzpicture}
  \node (mount) {Mount Point};
  \node (appserver) at (mount.west) {Under Application Server node's configuration structure};
  \node (syncnode) at (appserver.south) {syncNode.sh};
  \draw[->] (mount) -- (appserver);
  \draw[->] (appserver) -- (syncnode);
\end{tikzpicture}
\end{center}

\textit{Location of the \texttt{syncNode.sh} shell script}

To synchronize the node, do the following:

- Make sure Deployment Manager is up.
- Make sure servers in the node being moved -- particularly the Node Agent -- are down.
- Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
- If security is enabled for the Deployment Manager, understand what the administrative userid and password is for the cell.
- Open up a telnet session, or OMVS session, and go to the /\texttt{bin} directory of the application server node configuration being moved.
- "Switch users" to the WebSphere Admin ID
- Here's the command we entered to initiate the synchronization:
Command used to synchronize node with changes made in Master Configuration

Notes:
- The command must be entered as one line.
- Scenario 2 is unique in that it was the only one where the DMGR moved. That's why you see "wsc1" rather than "wsc3".
- In our environment security was enabled.

☐ Look for the messages that indicate success:
ADMU0401I: Begin syncNode operation for node g5noded with Deployment Manager wsc3.washington.ibm.com: 15510
ADMU0016I: Synchronizing configuration between node and cell.
ADMU0402I: The configuration for node g5noded has been synchronized with Deployment Manager wsc3.washington.ibm.com: 15510

Modify JCL start procedures to update SET ROOT= value

☐ Edit the JCL for the following servers and update SET ROOT= to reflect the new mount point:
- Application server controller JCL
- Application server servant JCL
- Node Agent controller JCL

Start servers on new MVS image

☐ Start the application servers and Node Agent on the target MVS image, now mounted at the new mount point.

Scenario 5 -- remapping mount point for Base Application Server node

We'll use our "Scenario 5" to illustrate the process of changing the mount point for a Base Application server node. Recall the scenario:
Note: No, we’re not out of sequence by mistake. We’re dealing with this scenario because it’s the last “easy” one before we tackle Scenario 3, which involves copying out a node structure from one HFS to another.

Let’s settle on some basic assumptions for this illustration:

- The node originally on SYSC was in an HFS with only this node, and no other nodes. So we don’t have to worry about affecting multiple nodes beyond the two we intend to affect, as was discussed under “Single HFS for all nodes?” on page 77.
- For whatever reason the mount point for this HFS on SYSD must be different than on SYSC.
- You’re familiar with all the other steps necessary to move a node to another MVS image, as detailed under “Scenario 5 in Detail” starting on page 70.

With those in mind, let’s walk through the process:

Create mount point, provide permissions/ownership and mount HFS

☐ Create the mount point. Permissions and ownership must be as follows:

Directories higher than mount point may be owned by anybody, but must have “other” permissions of at least “5” (read and execute)

Mount point for configuration:
Owner: WebSphere Admin ID
Group: WebSphere Config Group
Perm: 775

Ownership and permissions for WebSphere configuration mount point

☐ Mount the HFS on the new mount point

Update node-level variables.xml in BaseApp’s configuration

☐ Go into the Base Application server’s configuration structure

☐ Locate the variables.xml file for the application server node being moved. This will be at the node-level. In this scenario that was at the following location:

Location of DMGR node’s variables.xml file, under DMGR’s “Master Configuration”

☐ Edit the file and change all instances of the old mount point value string to be equal to the new mount point value. You should find four instances of the string in the file.

☐ Save the file.
Update `setupCmdLine.sh` in node's `/bin` directory

- Go into the Base Application server node's directory structure (under the new mount point)
- Go to the `/bin` directory
- Edit the `setupCmdLine.sh` file
- Update the `WAS_HOME` variable to reflect the new mount point.
- Save the file.

Run `bbowc2ns.sh` to regenerate the Base Application Server node

- Open up a telnet session, or OMVS session, and go to the `/bin` directory of the Base Application Server node
- "su" (switch user) to the WebSphere Admin ID (in our case, G5ADMIN)
  
  **Note:** You could use a UID=0 ID, but don't. That might leave file ownership and permissions in a funny state. Better to use the WebSphere Admin ID.
  
- Invoke the shell script:
  
  ```bash```
  ./bbowc2ns.sh
  ```

- Look for the indication of success:
  
  `BBOJ0056I Transformer Processing Complete, RC=0`

Modify JCL start procedures to update `SET ROOT=` value

- Edit the JCL for the following servers and update `SET ROOT=` to reflect the new mount point:
  - Base Application server controller JCL
  - Base Application server servant JCL
  - Daemon JCL

Start the server

- Start the Base Application server.
Scenario 3 -- copying out a node structure to a new HFS on a new mount point

We'll use our "Scenario 3" to illustrate the process of copying a node structure out of one HFS into another, then mounting that HFS at a different mount point. Recall the scenario:

Let's settle on some basic assumptions for this illustration:

- The only node we're moving is the application server node on SYSC. The Deployment Manager remains on SYSC.
  
  **Note:** Moving both nodes to SYSA on a different mount point is something we already covered under "Scenario 2 -- remapping mount point for multiple nodes" on page 82.

- For whatever reason the mount point for this HFS on SYSD must be different than on SYSC.
- You're familiar with all the other steps necessary to move a node to another MVS image, as detailed under "Scenario 3 in Detail" starting on page 41.

With those in mind, let's walk through the process:
Create mount point, provide permissions/ownership and mount HFS

- Create the mount point. Permissions and ownership must be as follows:

  / <root>
  / <higher_directories>
  / <mount_point>

  Directories higher than mount point may be owned by anybody, but must have "other" permissions of at least "5" (read and execute)

Mount point for configuration:
- Owner: WebSphere Admin ID
- Group: WebSphere Config Group
- Perm: 775

Ownership and permissions for WebSphere configuration mount point

- Create a new HFS
- Mount the HFS on the new mount point

Copy application server node structure over to new HFS

- In this example that was the /AppServerNodeC structure found under the old mount point. The copy command -- all on one line -- was:
  ```bash
cp -R /wasv51config/g5cell/AppServerNodeC /newmount/AppServerNodeC
  ```

  Note: You should be able to accomplish this under the authority of the "WebSphere Admin ID." If for whatever reason that doesn't work, use a UID=0 ID.

Copy the Daemon structure over to new HFS

- We used this command:
  ```bash
cp -R /wasv51config/g5cell/Daemon /newmount/Daemon
  ```

  Note: That copied over all the Daemon instances to the new HFS, even though we only needed one -- the new one with directory name SYSA that we created under "Copy Daemon directory" back on page 42. A Daemon instance directory is small -- copying everything under /daemon was the easiest thing to do. If you want, you can go into the new HFS later and clean out the unused Daemons -- SYSC and SYSD.

CHOWN both new directory structures

You may not need to do this, depending on whether you were successful in copying the directories while operating under the authority of the WebSphere Admin ID. But just in case:

- CHOWN the application server directory:
  ```bash
  CHOWN -R g5admin:g5cfg /newmount/AppServerNodeC
  ```

  Note: Make certain the owning group is the "WebSphere Config Group" for your cell. Having that group own the directory structure is key to later operations.

- CHOWN the Daemon directory:
  ```bash
  CHOWN -R g5admin:g5cfg /newmount/Daemon
  ```
**Update node-level variables.xml in DMGR master configuration**

- Go into the Deployment Manager's "Master Configuration"
- Locate the variables.xml file for the application server node being moved. This will be at the node-level. In this scenario that was at the following location:

```
/wasv51config/g5cell/
```

**Mount Point**

```
/DeploymentManager
```

**Under Deployment Manager's "Master Configuration"**

```
/config
```

```
/cells
```

```
/g5cell
```

```
/nodes
```

```
g5nodec
```

**Location of appserver node's variables.xml file, under DMGR's "Master Configuration"**

- Edit the file and change all instances of the old mount point value string to be equal to the new mount point value. You should find four instances of the string in the file.
- Save the file.

**Update setupCmdLine.sh in application server node's /bin directory**

- Go into the application server node's directory structure (under the new mount point)
- Go to the /bin directory
- Edit the setupCmdLine.sh file
- Update the WAS_HOME variable to reflect the new mount point.
- Save the file.

**Run syncNode.sh to synchronize master configuration changes out to node**

What's important to understand is that the syncNode.sh file is located in the /bin directory of the application server's node configuration, not the Master Configuration:

```
/<new mount>  
/AppServerNodeD
```

**Mount Point**

```
/bin
```

**Location of the syncNode.sh shell script**

To synchronize the node, do the following:

- Make sure Deployment Manager is up.
- Make sure servers in the node being moved -- particularly the Node Agent -- are down.
- Understand what the SOAP port of the Deployment Manager is. This can be found in the Admin Console under "System Administration," "Deployment Manager," "End Points," "SOAP Connector Address."
- If security is enabled for the Deployment Manager, understand what the administrative userid and password is for the cell.
Open up a telnet session, or OMVS session, and go to the /bin directory of the application server node configuration being moved.

"Switch users" to the WebSphere Admin ID

Here's the command we entered to initiate the synchronization:

```
./syncNode.sh wsc3.washington.ibm.com 15510 -conntype SOAP -user g5admin -password xxxxxx
```

Command used to synchronize node with changes made in Master Configuration

Notes:  
- The command must be entered as one line.  
- In our environment security was enabled.

Look for the messages that indicate success:

- ADMU0401I: Begin syncNode operation for node g5noded with Deployment Manager wsc3.washington.ibm.com: 15510
- ADMU0016I: Synchronizing configuration between node and cell.  
- ADMU0402I: The configuration for node g5noded has been synchronized with Deployment Manager wsc3.washington.ibm.com: 15510

Modify JCL start procedures to update SET ROOT= value

Edit the JCL for the following servers and update SET ROOT= to reflect the new mount point:

- Application server controller JCL  
- Application server servant JCL  
- Node Agent controller JCL  
- Daemon JCL

Note: Be alert regarding the Daemon JCL proc. Be certain the same proc isn't used with the other nodes in the cell. If it is, you may not be able to mount the node at a different mount point without also changing the Daemon's JCL start proc (the reference to which is held in the node-level variables.xml, and is named on the variable WAS_DAEMON_daemon_start_command.)

Start servers on new MVS image

Start the servers on the target MVS image, now mounted at the new mount point.

The difference between changing all system name reference in variables.xml vs. just one

When changing the system name reference in the node-level variables.xml file, we have on several occasions within this document suggested that two approaches work equally well:

1. Change all the instances of the system name string in the file, or  
2. Change only the "WAS_DAEMON_ONLY_server_configured_system_name" value

Do they actually have the exact same effect? No. The two approaches yield different things. Both ultimately work -- the Daemon starts on the new system and is happy because its defined
system name matches that of the system on which it starts -- but what you see in the configuratin HFS is different. Let's see what the differences are.

**Changing just WAS_DAEMON_ONLY_server_configured_system_name value**

Changing this one variable results in no obvious changes to the HFS. The change finds its way into the Daemon's was.env file, where the server_configured_system_name variable is updated. That's sufficient to allow the Daemon to be started on the new MVS image.

An inspection of the Daemon's HFS directory structure yields the following:

```
<mount point>
  /G5CELL.G5CELL.SYSC
  /Daemon
  /config
  /cells
    /g5cell
    /G5CELL
    /SYSC
  /was.env
```

**Daemon's configuration HFS after single variable changed**

**Conclusion:** Changing the WAS_DAEMON_ONLY_server_configured_system_name value only results in no visible change to the Daemon's configuration HFS structure. The "Daemon Instance Name" (which in normal circumstances is equal to the system name on which the Daemon resides) remains as it was before. The Daemon starts on the new system and the servers in the node operate properly as well.
Changing all instances of the system name string in the file

If you change all the instances of the system name string in the node-level variables.xml file, it has the effect of creating a new Daemon directory and a modified symbolic link:

```
<mount point>
  /G5CELL.G5CELL.SYSA
  /Daemon
  /config
  /cells
    /g5cell
      /G5CELL
      /SYSA was.env
      /SYSC was.env
</mount point>
```

Daemon's configuration HFS after all variables changed

The update to the WAS_DAEMON_daemonInstanceName value in the node-level variables.xml file was what caused the new directory to be created. What's important to realize is that since the Daemon's instance name has changed, some other variables in the variables.xml file must also change:

```
"WAS_DAEMON_daemon_start_command_args" value="JOBNAME=G5DEMN,ENV=G5CELL.G5CELL.SYSA"
"WAS_DAEMON_ONLY_control_region_configured_system" value="SYSA"
"WAS_DAEMON_ONLY_server_configured_system_name" value="SYSA"
"WAS_DAEMON_ONLY_server_specific_short_name" value="SYSA"
"WAS_DAEMON_daemon_was_env_file" value="/wasv51config/g5cell/Daemon/config/cells/g5cell/G5CELL/SYSA/was.env"
"WAS_DAEMON_daemonInstanceName" value="SYSA"
```

Relationship of some other variables to the Daemon's instance name value

Notes:

1. The start command string for the Daemon includes the standard ENV= pointer to the symbolic link for the server. If the directory name where the was.env for the Daemon resides has changed, then this value too much change.

2. The pointer to the was.env for the Daemon includes the directory name. We know that changing the "daemonInstanceName" variable results in a new directory being created. Therefore, this value must change as well.

Conclusion: If you change more than just the "server_configured_system_name" value, then change all the values in the file. That keeps things "in synch."

Note: Is it possible to change some of the values and have the Daemon work properly? Maybe. We didn't test all the combinations of changes to find out.
Document Change History

Check the date in the footer of the document for the version of the document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 7, 2005</td>
<td>Original document.</td>
</tr>
</tbody>
</table>
| February 12, 2005 | • Added a section in the "Executive Overview" that described the difference between this white paper and WP100415 (which described how to move the Deployment Manager).  
  • Modified the instructions regarding the node-level variables.xml file in Scenarios #1, #2 and #5. In the initial version of this document we had you modify only one instance of the system name string in the file. After further testing, it was found that doing a global change of the system name string in that file works equally as well. Either works, but instructing readers to change all is intuitively easier.  
  • To help explain the differences between changing one value vs. changing all the values, a section in Appendix A was added to do just that. |
End of WP100542