

IBM WebSphere Web Multi-Platform Configuration

# XML Configuration Manager Tool



## About the Lecture



- The following will be presented:
  - ▲ What is the XML Configuration Manager Tool
  - ▲ Fundamental Features
  - ▲ Command syntax, XML grammar, and XML file notes
  - ▲ How to export
  - ▲ How to create a new Applications Server
  - ▲ How to make minor change



## What is XMLConfig?



- XML Configuration Management Tool (XMLConfig)
  - ▲ Allows you to import and export **configuration** data to and from the WebSphere Application Server administrative repository.
  - ▲ Uses XML documents to perform the same tasks that can be done in the WebSphere Administrative Console.
  - ▲ The tool that was available in version 3.02 of WebSphere as a technology preview is a fully functional tool in version 3.5



- The XML-based approach complements the administration you can perform through the WebSphere Administrative Console.

# What is XML?



- eXtensible Markup Language
  - ▲ Sibling to HTML
  - ▲ Derived from SGML (Standard General Markup Language)
  - ▲ HTML defines "presentation"
  - ▲ XML defines "structure"



## Extensible Markup Language (XML)

XML is considered "an essential component in the Java 2 Platform, Enterprise Edition, and will be supported throughout the Java 2 Platform" (see <http://java.sun.com/xml/> for details)

WebSphere provides standard XML server tools:

- SAX 1.0 parser
- DOM 1.0 parser
- DTD for JSPs
- XSL support



- XML is an essential component in the Java 2 Platform, Enterprise Edition and will be supported throughout as a means for enabling business-to-business information interchange using XML. For robust, synchronous data messaging, Enterprise JavaBeans (EJB) can be used to create a business service object, and related XML content can be sent over the wire using JavaServer Pages technology. Currently, JavaServer Pages can be used to generate and consume XML between n-tier servers or between server and client. Java Messaging Service provides a means for asynchronous XML data messaging. In addition, Enterprise JavaBeans uses XML to describe its deployment properties, giving Enterprise JavaBeans data portability in addition to its code portability.
- WebSphere Application Server includes complete XML support today for parsing and understanding information sources that utilize XML tags. In the next major version with J2EE and EJB 1.x support, WebSphere will include XML development/deployment descriptor support within EJBs.

# XML Properties



- User Defined tags
- Hierarchical
- DTD (Document Type Definition) defines a valid, well-formed XML document
  - ▲ (DTD is to XML as a Spell/Grammar checker is to a Word Processor document)



## XML File Notes



- XML is similar to HTML tags. It is easy to add/invent tags
- You will see a start and end tag like: `<virtual-host>` and `</virtual-host>`
- WebSphere Application Server Configuration Markup Language syntax (WASCML)
- XML documents must be coded to the WASCML syntax



- WASCML is the IBM language that will be available when the DTD is available.
- The WASCML is published in WebSphere version 3.5. The WASCML Document Type Definition (DTD) is in the **xmlconfig.dtd** file that is located in the `<server_root>/bin` directory.

## Fundamental Features



- The XMLConfig tool provides three fundamental features:
  - ▲ Full export  
Generates an XML document that describes the configuration of the entire administrative domain.
  - ▲ Partial export  
Provides an XML document that specifies the administrative objects to export from the administrative domain.
  - ▲ Import  
Imports a newly created XML document or a document that you previously exported and modified.



- Full export - In effect, the full export takes a "snapshot" of the administrative repository contents.
- Partial export - The objects and their children are exported to an XML document that you specify.
- Import - In the XML document, you can specify to add, modify, or remove administrative objects such as servlets and virtual hosts. Your XML document can replace the administrative repository contents partially or entirely.

## XML File Notes



- XML is hierarchical
- Parent XML tag is <websphere-sa-config>
- It has five supported children tags
  - <virtual-host>
  - <jdbc-driver>
  - <data-source>
  - <node>
  - <model>



- An XML viewer is available from [www.alphaworks.ibm.com](http://www.alphaworks.ibm.com).
- Each supported <websphere-sa-config> child tag contains additional child tags.

## Command Syntax



- Run the XMLConfig program `<as_root>/bin`
  - ▲ **-adminNodeName** - (mandatory parameter)
  - ▲ **-import || -export || -export -partial** (mandatory parameter)
  - ▲ **-nameServiceHost, -nameServicePort** - (Optional parameters)
  - ▲ **-traceString** - specifies the WebSphere Application Server internal code to trace
  - ▲ **-substitute** (optional parameter)



- **-adminNodeName** - Required argument that specifies the node that contains the administrative server to which you are connecting. The argument value must match the node name that is given in the topology tree on the Topology tab of the WebSphere Administrative Console.
- **-import || -export || -export -partial** - Required argument that specifies the operation to perform -- an import or export. Unless you also specify the parameter `-partial`, the export will be treated as a full export.
- **-nameServiceHost, -nameServicePort** - Optional arguments that specify the hostname of the machine that contains the name service, and the port by which to communicate with that name service.
- **-traceString** - Optional argument that specifies the WebSphere Application Server internal code that you want to trace.
- **substitute** - specifies the variables to be substituted (for example, `-substitute "NODE_NAME = admin_node; APP_SERVER = default_server"`). The above argument will substitute any occurrence of `$NODE_NAME$` with `admin_node` and `$APP_SERVER$` with `default_server`, in the input xml file.

## XML Grammar



- Each Object tag in the XML document contains
  - ▲ An object **type**, such as "node" or "virtual-host"
  - ▲ Any attributes that are associated with **type** have **name/value** pairs. Attributes are optional.



# XML Grammar



- Within WebSphere two commonly used attributes are **name** and **action**
  - ▲ The **name** attribute
    - ◆ Identifies the particular resource on which to perform the action
    - ◆ Example: `<virtual-host name="default_host">`
    - ◆ Example: `<node name="mynode">`
  - ▲ The **action** attribute
    - ◆ Controls the behavior of the import or partial export operation.
    - ◆ Example: `<virtual-host name="default host" action="update">`
    - ◆ Example: `<node name="mynode" action="create">`



## XML Grammar - action



- Currently, fourteen values for the action attribute exist.
  - ▶ **create**: Adds the specified resource to the administrative domain.
  - ▶ **update**: Updates the properties of a specified resource.
  - ▶ **delete**: Removes the specified resource and its children (recursive delete).



- Create: If the resource already exists, the create action is treated as an update action. When you are using this action, you must specify all required attributes for the object type.
- Update: You only need to specify the properties that you want to update. However, if the resource does not exist, the update action becomes a create. In such a case, if some of the properties are not specified, an error will occur.

## XML Grammar - action



- ▲ **locate**: Locates the specified resource.
- ▲ **export**: Exports the configuration of the specified resource and its children to the output XML document. Applicable for partial exports only.
- ▲ **start**: Starts the specified resource (if applicable).



- locate - Use this action to provide the containment path to specific child resources. Applies to the partial export operation as well as the import operation.
- start - You cannot start a node.

## XML Grammar - action



- ▶ **stop**: Stops the specified resource (if applicable).
- ▶ **restart**: Stops the specified resource and starts it again (if applicable).
- ▶ **stopforrestart**: Stops the specified resource and restarts it (if applicable). Node only.
- ▶ **enable**: Makes the specified resource available for user requests.
- ▶ **disable**: Makes the specified resource unavailable for user requests.



- enable - Currently applies only to servlets and Web applications.
- disable - Currently applies only to servlets and Web applications.

## XML Grammar - action



- **createclone**: Create a clone
- **associateclone**: Associates a clone to a model
- **disassociateclone**: Disassociates a clone from a model



## XML Grammar - action



- Note: Some of the operations attributes, such as start and stop, do not apply to all object types. In general, if the operation is supported in the WebSphere Administrative Console, it is supported in the XML import operation.



## Full Export



- Great way to take a snap shot of the current configuration
  - ▲ Rebuild the WebSphere Application Server to an extent
  - ▲ Undo recent changes
  
- Run the following from `<as_root>/bin`
  - ▲ `XMLConfig -adminNodeName your node -export export.xml`



- On AIX and Solaris use the `./XMLConfig.sh` command.
- On Windows NT use the `XMLConfig.bat` command.

## Partial Export



- Specify the starting level to begin the export.
  - ▲ The objects and their children are exported to an XML document that you specify.
  - ▲ Export the Big3Server Application Server to make a quick change.



## Objects you can import and export with XML Config

<b>Class</b>	<b>Summary</b>	<b>Class</b>	<b>Summary</b>
ApplicationServerConfig	Processes XML for Application Server Objects XML Structure, including immediate children	ServletConfig	Processes XML for Servlet Objects XML Structure, including immediate children
BaseConfig	Base Class for All Config Classes	ServletEngingConfig	Processes XML for ServletEngine Objects XML Structure, including immediate children
ContainerConfig	Processes XML for Container Objects XML Structure, including immediate children	ServletRedirectorConfig	Processes XML for ServletRedirector Objects XML Structure, including immediate children
DataSourceConfig	Processes XML for DataSource Objects XML Structure, including immediate children	SessionManagerConfig	Processes XML for SessionManager Objects XML Structure, including immediate children
EJBConfig	Processes XML for EJB Objects XML Structure, including immediate children	URIConfig	Processes XML for URI Objects XML Structure, including immediate children
GenericServerConfig	Processes XML for GenericServer Objects XML Structure, including immediate children	UserProfileManagerConfig	Processes XML for UserProfileManager Objects XML Structure, including immediate children
JDBCDriverConfig	Processes XML for JDBCDriver Objects XML Structure, including immediate children	VirtualHostConfig	Processes XML for VirtualHost Objects XML Structure, including immediate children
ModelConfig	Processes XML for Model Objects XML Structure, including immediate children	WebApplicationConfig	Processes XML for WebApplication Objects XML Structure, including immediate children
NodeCofig	Processes XML for Node Objects XML Structure, including immediate children	XMLConfig	Command-line WebSphere XML Import/Export Tool Usage
OLTControllerConfig	Processes XML for OLTController Objects XML Structure, including immediate children		

## Stopping Server with XML



- Simple XML file to stop the Server

```
<websphere-sa-config>  
  <node name="yournode" action="locate">  
    <application-server name="yourappserver"  
action="stop">  
      </application-server>  
    </node>  
  </websphere-sa-config>
```



## Stopping Server with XML



- Import the XML file with
  - ▲ XMLConfig -adminNodeName your node -import stop.xml
  
- Server will stop
  - ▲ WebSphere Administrative Console needs to be refreshed manually to see the effect.



## Create New Application Server

- Using an XML file, you can quickly create a new Application Server
  - ▲ See lab for complete file listing.
  - ▲ Note the use of **update**. It becomes **create** if the resource does not exist.
  - ▲ Performs all the steps at once instead of doing them one at a time in Administrative Console.



## Administrative Console



- Once you import an XML file, if the Administrative Console does not properly show the changes:
  - ▲ Click the **Refresh** button
  - ▲ Exit Administrative Console and start it again



- The XML configuration is stored in the configuration database.
- The S/390 version currently does not have an administrative console, but since XML configuration is a Java program, it should work across all platforms.

## Document Type Definition (DTD)



- DTD specifies the grammatical structure of an XML document by containing a list of tags that define how the elements of the document relate to one another within the document's tree structure. This allows XML parsers to understand and interpret the document's contents.



- Starting with 3.5, DTD is published in WebSphere. The DTD is in the **xmlconfig.dtd** file that is located in the <server\_root>/bin directory.

## DTD Form



- A DTD comes in the form of a simple text file that can be either:
  - ▶ Stored in a separate file, OR
  - ▶ Embedded within the XML file
  
- XML documents that reference a DTD will contain the `<!DOCTYPE>` declaration which either contains the DTD declarations, or specifies the location of an external DTD. (`<!DOCTYPE LibraryCatalogue SYSTEM "library.dtd">` for example)



- For more information, please see the IBM DTD tutorial at <http://www-4.ibm.com/software/developer/library/buildappl/writetd.html>.

## DTD Catalogs



- To make it easier to use industry-standard DTDs and other grammars, local copies are installed with the Application Server. Some of the grammars in the library are:
  - ▲ Channel Definition Format (CDF)
  - ▲ Mathematics Markup Language (MathML)
  - ▲ Wireless Markup Language (WML)
  
- Over 100 DTD catalogs are in:  
<server\_root>/web/xml/grammar/dtd  
subdirectories



- CDF was developed by Microsoft. CDF enables subscription to Web-based channels (a Web site or a portion of a Web site). The subscription can be implemented to have the Web site automatically send the subscriber updates to the channel (using push technology) or transmit the updates at the request of the subscriber (using pull technology). In either case, the subscriber must be able to link to a CDF file on the channel site. The CDF file is an XML document that conforms to the CDF DTD.
- MathML was developed by the W3C (World Wide Web Consortium). MathML is a grammar for documents that contain math formulas, notations, and other data.
- WML is a grammar for creating documents so that they can be efficiently transmitted and displayed on narrowband devices such as cellular phones, personal digital assistants (PDAs), and pagers.

## What This Lecture Covered



- This lecture covered the following:
  - ▲ What is the XML Configuration Manager Tool
  - ▲ Fundamental Features
  - ▲ Command syntax, XML grammar, and XML file notes
  - ▲ How to export
  - ▲ How to create a new Applications Server
  - ▲ How to make minor change



## IBM WEBSHERE WORKSHOP - LAB EXERCISE

# XML Configuration Manager Tool Lab

### What This Exercise is About

In the following lab you will use the XML Configuration Manager Tool (XMLConfig) to import and export configuration data to and from the WebSphere Application Server administrative repository.

### User Requirement

You should have WebSphere Application Server installed and set up with the Big3 Application.

### What You Should Be Able to Do

After completing this lab, you should have a basic understanding of the XMLConfig program. You should be able to perform common tasks such as exporting or importing changes.

### Introduction

The XML Configuration Management Tool (XMLConfig) allows you to import and export configuration data to and from the WebSphere Application Server administrative repository. This tool offers another way to perform WebSphere Administration tasks other than by using the GUI interface.

The XMLConfig program uses XML documents to specify the configuration data. The XML documents must be coded to the WebSphere Application Server Configuration Markup Language (WASCML) syntax. A print out of WASCML syntax has been provided for you in *Appendix F*.

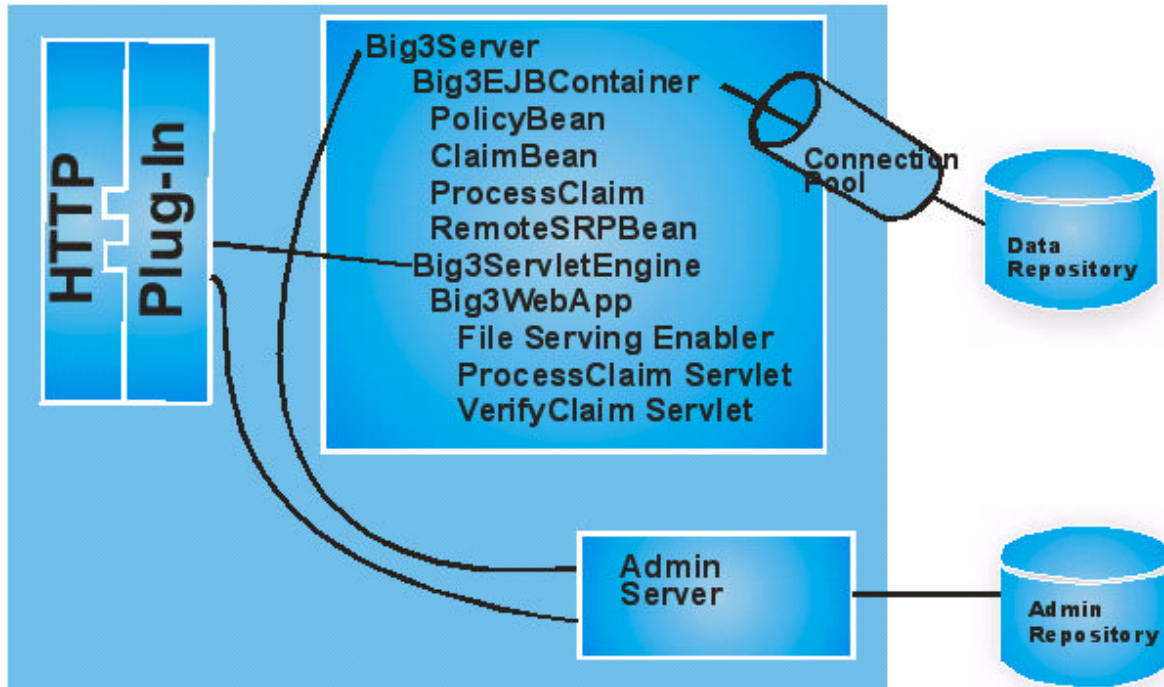
The XMLConfig tool provides three fundamental features:

- **Full export**
- **Partial export**
- **Import**

This lab will consist of the following parts:

- Part One: Export All Objects in the WebSphere Administrative Domain
- Part Two: Starting and Stopping the WebSphere Application Server by Using XML
- Part Three: Creating a New Application Server, Web Applications, and Servlets
- Part Four: Adding Servlets to an Existing Web Application
- Part Five: Partial Export of an Application Server
- Part Six: Configuring Big3 by Using XML Config
- Part Seven: Testing the Application Server

The following diagram shows the configuration you will achieve by following the steps in this lab:



### Exercise Instructions

Parts One through Five of this lab are to be completed on the AIX lab computer. Parts Six and Seven are to be completed on the Solaris machine. Estimated time: 1 hour 15 minutes

The following lab assumes that the WebSphere Application Server has been installed and that you have the Big3 EJB Application setup. It also assumes that the Application Server and WebSphere Administrative Console are running. We will use the WebSphere Administrative Console to view changes that are made with the XMLConfig program.

To indicate the WebSphere Application Server installation directory, **<as\_root>** will be used. Replace this with the full path where you installed WebSphere Application Server. On the AIX machine this is **/usr/WebSphere/AppServer**.

Note: Throughout the lab, replace **<node\_name>** with the node name that contains the administrative server to which you are connected. Use the WebSphere Administrative Console and select the **Topology** view to determine the node name of your AIX lab computer.

### **Part One: Export All Objects in the WebSphere Administrative Domain**

In this part of the lab, we will use the XMLConfig tool to export all the objects that are in the WebSphere Administrative Domain. This is a great way to take a “snap shot” of the current configuration, and could be used to quickly rebuild the WebSphere Application Server.

Note: If you get a login prompt, security is still on, and the system appears to be waiting for the thread to die. **Remember to restart your administrative server.**

\_\_1. In a terminal window, type **cd <as\_root>/bin**.

\_\_2. Type **.XMLConfig.sh -adminNodeName <node\_name> -export export.xml**.

The XMLConfig program will connect to the WebSphere Administrative Server, “walk the tree” of all the objects in the administrative repository, and generate an XML file that is called **export.xml**. Since no path was specified, the file will be put in the current directory (the **bin** directory).

\_\_3. You can view the generated text file with any text editor. Launch your editor and view the **export.xml** file. Verify that **<?xml version="1.0"?>** is the first line of the XML file.

**Note:** To start the text editor in AIX, click the pencil/paper icon or type **dtpad <filename>** in a terminal window.

Like HTML tags, there are start and end tags for each section of the generated XML file. For example, locate the **<virtual-host>** section and note the **</virtual-host>** tag that ends the section.

Note that the action for each section is “update”. An “update” on a nonexistent object is an implicit create. Since we did a full export of all objects, this XML file can be used to re-create all objects on a fresh install of WebSphere Application Server. Simply change the node name


and you can quickly re-create the resources. Since the resources would not exist on a fresh install, the “update” action would change to “create”.

**Version 3.5 eliminated the following limitations of XMLConfig** - There was no way to XML-base install the database driver on a node. You had to manually edit the XML file and add the **<install-info>** tag.

## Part Two: Starting and Stopping the WebSphere Application Server by Using XML

Common tasks such as starting and stopping the WebSphere Application Server had to be done with the WebSphere Administrative Console. These tasks can now be done by using the XMLConfig program. This part of the lab will show you how easily this can be done.

- \_\_1. Confirm that the **Big3Server** is running by using the **Topology** view of the WebSphere

Administrative Console and looking for the “running” icon . Select the **Big3Server** and the General Tab will display the current state as either running or stopped.

- \_\_2. Start a text editor and create a new file. Add the following lines:

```
<?xml version="1.0"?>
<!DOCTYPE websphere-sa-config SYSTEM "$server_root$$dsep$bin$dsep$xmlconfig.dtd">

<websphere-sa-config>
  <node name="<node_name>" action="locate">
    <application-server name="Big3Server" action="stop">
      </application-server>
    </node>
  </websphere-sa-config>
```

- \_\_3. Save the file as **stop.xml** (It is acceptable to save it in the **<as\_root>/bin** directory) and exit the text editor.

- \_\_4. In a terminal window, type the following:

```
./XMLConfig.sh -adminNodeName <node_name> -import stop.xml
```

The XMLConfig program will read the **stop.xml** file, connect to the WebSphere Application Server, and perform the stop action.

**Current limitations of XMLConfig** - If you go back to the WebSphere Administrative Console, it will say that the server is running. However, the console messages will say that the server has stopped. This is because the topology tree does not match the stopped state. You must select the **Big3Server** in the topology view and activate the **Refresh** button -



, or restart the WebSphere Administrative Console to have any changes by XMLConfig displayed. This issue has been identified and a fix is planned.

- \_\_5. Edit the **stop.xml** and change the action from **stop** to **start**. Save the file as **start.xml**.
- \_\_6. In a terminal window, type the following:  
**./XMLConfig.sh -adminNodeName <node\_name> -import start.xml**
- \_\_7. Click **Refresh** on the WebSphere Administrative Console and note that the server has started.

### Part Three: Creating a New Application Server

In this part of the lab, we will create an XML document that will define a New Application Server that will be named myServer.

- \_\_1. Copy **/usr/xml\_lab/createAS.xml** file to **<as\_root>/bin** by typing the following commands:

```
cd <as_root>/bin
cp /usr/xml_lab/createAS.xml .
```

**Note the (space period) at the end of the copy command**, if they are not there the command will fail. The "." tells AIX to place the files that are being copied in the current directory.

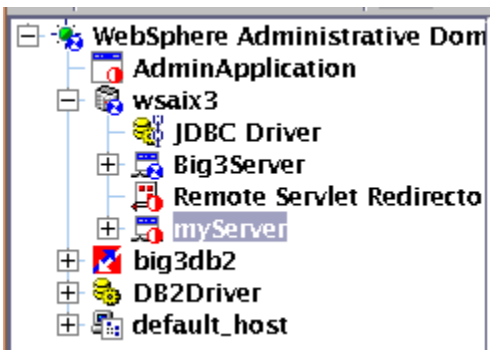
- \_\_2. Start your text editor and open the **createAS.xml** document. Note the XML tags content - **<application-server>**, **<container-name>**, and **<servlet-engine>**. Compare them with the information in **Appendix F: XML Package Summary**.
- \_\_3. Change the node name to your machine's node name, and save the **createAS.xml** file.

```
<websphere-sa-config>
  <node name="wsaix#" action="locate">
    <application-server name="myServer" action="update">
      <executable>java</executable>
```

\_\_4. Import the **createAS.xml** by typing the following:

```
./XMLConfig.sh -adminNodeName <node_name> -import createAS.xml.
```

\_\_5. Select the node in the topology view and activate the **Refresh** button, or stop and restart the WebSphere Administrative Console to verify that a new application server named "myserver" was created.



In this step we essentially bypassed all the steps that you would have done in the Administrative Console to create a new application server called **myServer**.

#### **Part Four: Adding Servlets to an existing Web Application**

In this part of the lab, we will demonstrate the adding of servlets to the Web application that was created in the previous step.

\_\_1. Copy **/usr/xml\_lab/createServlets.xml** to **<as\_root>/bin**.

\_\_2. Start your text editor and open the **createServlets.xml** document.

Notice the action attribute for tags: *node*, *application-server*, *servlet-engine*, and *web-application* is set to **locate**.

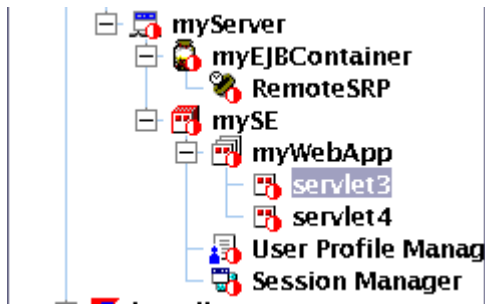
Observe the action attribute for the *servlet* tag is set to **update**. Since the servlet does not exist, it is implicitly created.

\_\_3. Change the node name to your machine's node name, then save and close the **createServlets.xml** document.

\_\_4. Now create the two servlets by typing the following command:

```
./XMLConfig.sh -adminNodeName <node_name> -import createServlets.xml.
```

\_\_5. Select *myWebApp* (under **myServer** --> **mySE**) in the **Topology** view and activate the **Refresh** button, or stop and restart the WebSphere Administrative Console and verify that *servlet3* and *servlet4* were added.



It is a good idea to browse through the Administrative Console.

### **Part Five: Partial Export of Big3 Application Server**

In this part of the lab, we will use the partial export feature of XMLConfig to export just the Big3Server Application Server. The Application Server object and all its children are exported.

To do a partial export you need an XML file that specifies the resources that you want to export. For this lab, the partial export XML file is called **partial\_exportAS.xml**, and it is used as an input parameter in the XMLConfig export.

\_\_1. Copy the `/usr/xml_lab/partial_exportAS.xml` to `<as_root>/bin`.

\_\_2. Open the **partial\_exportAS.xml** file by using a text editor. Note that the *jdbc-driver name* is **OracleDriver**, the *data-source name* is **big3oracle**, and the *application-server name* is **Big3Server**. Notice the *action* attribute is **export**.

**Remember:** In the deployment lab exercise, you changed the container properties from DB2 to Oracle.

\_\_3. Change the **node name** to your machine-specific node name. Notice that the *action* attribute is **locate**, then save and close the **createServlets.xml** document.

- \_\_4. Now do the partial export by typing the following command\*:

```
./XMLConfig.sh -adminNodeName <node_name> -export big3ASCfg.xml -partial  
partial_exportAS.xml
```

**\*NOTE:** The command should be typed continuously on one command line. This is a single command, not two. It is advisable to review the process log and look for Java exceptions or other errors.

- \_\_5. Open the **big3ASCfg.xml** file just created by using a text editor. Browse the file to see the coding.

### **Part Six: Configuring Big3 by Using XMLConfig**

The next step is to import the **big3ASCfg.xml** that was created in the last part to a Solaris machine. First, move the Big3 files in the correct places. The files that make up the Big 3 application are stored on the Solaris machine in **/usr/Big3**.

- \_\_1. Copy the deployed **Big3 JAR file** to the `<as_root>/deployedEJBs` directory by typing the following commands:

```
cd <as_root>/deployedEJBs  
cp /usr/big3/big3deployed.jar .
```

**Note the (space period) at the end of the copy command**, if they are not there the command will fail. The “.” tells Solaris to place the files that are being copied in the current directory.

- \_\_2. Create directories in the path `<as_root>/hosts/default_host/Big3WebApp/servlets` and `<as_root>/hosts/default_host/Big3WebApp/web` by typing the following commands:

```
cd <as_root>/hosts/default_host  
mkdir Big3WebApp  
cd Big3WebApp  
mkdir web  
mkdir servlets
```

- \_\_3. Copy all of the **HTML** files to the `<as_root>/hosts/default_host/Big3WebApp/web` directory by typing the following commands:

```
cd <as_root>/hosts/default_host/Big3WebApp/web  
cp /usr/big3/html/* .
```

- \_\_4. Copy the **servlet JAR file** to the `<as_root>/hosts/default_host/Big3WebApp/servlets` directory by typing the following commands:

```
cd <as_root>/hosts/default_host/Big3WebApp/servlets
cp /usr/big3/servlets/* .
```

- \_\_5. FTP the **big3ASCfg.xml** file from the AIX computer over to the Solaris computer. Put the file into the `<as_root>/bin` directory. The instructor can assist you with any FTP commands.

- \_\_6. Start the text editor and open the `<as_root>/bin/big3ASCfg.xml` file.

- \_\_7. Change the **node name** to match your `<Sun_machine_name>`.

**NOTE:** You will need to do this in two places.

- \_\_8. Use the search feature of the text editor and replace all the `/usr/WebSphere` instances that are in the path with `/opt/WebSphere`.

- \_\_9. Use the search feature of the text editor and locate `<install-info>` tag within the string "jdbc-driver name". Then make the following changes:

```
<install-info>
  <node-name><machine_name></node-name>
  <jdbc-zipfile-location>/export/home/oracle/jdbc/lib/classes12.zip</jdbc-zipfile-location>
</install-info>
```

- \_\_10. Delete the queue name so that it is as follows: `<queue-name></queue-name>`. This will allow the system to generate a valid queue name.

- \_\_11. Change the `<transport-port>` to `-1`. This command instructs Solaris to find and use an open transport port.

- \_\_12. Save the **big3ASCfg.xml** file.

- \_\_13. Verify that the Administrative Server is running. Select the node, and verify that the current state is running.

- \_\_14. To import the file to WebSphere, use the XMLConfig program and type the following command from the **<as\_root>/bin** directory:

```
./XMLConfig.sh -adminNodeName <machine_name> -import big3ASCfg.xml
```

**NOTE:** It is advisable to review the process log and look for Java exceptions or other errors.

### **Part Seven: Testing the Application**

The last step is to test the application that was imported as the **big3ASCfg.xml** file.

- \_\_1. Start the WebSphere administrative console. If it is already running, then stop and restart it.
- \_\_2. Start the **Big3Server** application server.
- \_\_3. Open a browser and access the **index.html** for Big3 on the Solaris machine.
- \_\_4. Submit a claim to test the application. It should complete successfully.

### **Summary**

This lab showed you how to use the XML Configuration Manager Tool (XMLConfig) to import and export configuration data to and from the WebSphere Application Server administrative repository. We also used the tool to stop and start the WebSphere Application Server, create a new Application Server, add servlets to an existing Web Applications, and perform a partial export of the Big3 Application Server.

### **END OF EXERCISE**