

**WebSphere Application Server V4.0.1 for zOS and OS/390**

# **PolicyIVP Construction Lab**

This document can be found on the web at:  
[www.ibm.com/support/techdocs](http://www.ibm.com/support/techdocs)  
Search for document number WP100277 under the category  
of "White Papers"

Date: *Tuesday, May 21, 2002*

**IBM Washington Systems Center**

Donald C. Bagwell  
301-240-3016  
[dbagwell@us.ibm.com](mailto:dbagwell@us.ibm.com)

The author gratefully acknowledges those in the Washington Systems Center who assisted in the writing of this document. Mike Cox is the original author of the process by which the PolicyIVP application could be taken apart, rebuilt and deployed. I shamelessly stole his original work 😊. John Hutchinson and Bob Teichman provided numerable suggestions for improvement, which I have incorporated into this document.

<b>A Little Background</b>	1
<b>What's Required on your Workstation</b>	1
<b>What's Required on your zOS Server</b>	1
<b>An Important Note about the Bitmap Illustrations Shown Here</b>	1
<b>Overview of Lab</b>	1
<b>Phase 1: Get ZIP file from Website and Unzip to PC</b>	2
Overview	2
Create PC Directory and FTP files	2
<b>Phase 2: Generate EAR using AAT</b>	3
Overview	3
Start AAT and Create the Application	3
Import the Utility Files	3
Import the EJB JAR Files	5
Modify the Properties of the EJBs	6
<i>BMP Entity Bean</i>	6
<i>CMP Entity Bean</i>	7
<i>Session Bean</i>	7
Import the Web Application and Modify its Properties	8
Validate, Deploy and Export the Application	10
<i>Validate</i>	10
<i>Deploy</i>	11
<i>Export</i>	11
Review	11
<b>Phase 3: Create Application Server</b>	12
Overview	12
Start SMS EUI and Add Conversation	12
Add Server	13
Add Server Instance	14
Add J2EE Resource and Resource Instance for DB2	15
Validate, Commit and Activate the Conversation	16
Perform non-GUI Tasks	17
<i>Define WLM Application Environment</i>	17
<i>Run RACF Batch Job</i>	18
<i>Grant Server Region ID Access to PolicyIVP Database Table</i>	19
<i>Create JCL Start Procedures</i>	20
<i>Start Server and Verify Registration</i>	22
Review	23
<b>Phase 4: Deploy Application</b>	24
Overview	24
Deploy Your Version of PolicyIVP Application	24
<i>Add New Conversation</i>	24
<i>Import the Application</i>	24
<i>BMP Bean: resolve resource reference and set JNDI name</i>	26
<i>CMP Bean: resolve resource reference and set JNDI name</i>	27
<i>Session Bean: set JNDI name</i>	27
<i>WebApp: set JNDI name</i>	27
<i>Transfer the EAR File to the WAS Server</i>	27
<i>Validate, Commit and Activate the Conversation</i>	28
Review	28
<b>Phase 5: Test Application</b>	29
Modify "ejbivp.sh" shell script	29
Run the shell script	30
Review	30

**WP100277 - PolicyIVP Construction Lab**

<b>Reference: Configuring the WebApp to Serve as the Client</b> .....	31
<b>Reference: Where the JAR Files Came From</b> .....	31
Overview .....	31
Import Repository .....	31
Export JAR Files .....	32
<b>Reference: Where the WAR File Came From</b> .....	34
Download PolicyIVP.ear file .....	34
Extract PolicyWebApp.war file .....	34
<b>Change History of Document</b> .....	34

## A Little Background

This white paper was born from a lab exercise created for the "Wildfire" education for WebSphere V4.0.1 for zOS and OS/390. This paper is essentially the lab exercise, modified somewhat to make it less specific to the class itself.

This white paper has three companion files, which were included on the same page as this paper ([www.ibm.com/support/techdocs](http://www.ibm.com/support/techdocs), search for White Paper WP100277).

- WP100277\_intro.pdf** -- a short paper that explains what this EJB application called "PolicyIVP" really is. Understanding the application will help you make sense of this document as you do things while deploying the application.
- WP100277\_A.zip** -- in the Wildfire class we simply supply the students the program files in the form of files in the system HFS. For this paper those five files are supplied as a separate ZIP file.
- WP100277\_B.zip** -- this ZIP consists of various JCL jobs needed to create the new J2EE application server. Read the file `index.txt` inside the ZIP file for a description of what each file is for. The instructions in this white paper will tell you when to go get a file from this ZIP.

## What's Required on your Workstation

In order to run through this exercise, your workstation requires:

- WindowsNT or Windows2000 operating system
- At least 256MB of memory (more is better)
- WebSphere V4.0.1 Application Assembly Tool (AAT) Version 023 or higher
- WebSphere V4.0.1 Systems Management Administration Tool Version 4.01.006 or higher
- A 3270 emulator

## What's Required on your zOS Server

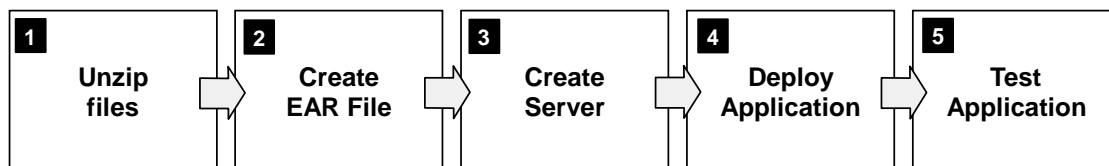
WebSphere Application Server V4.0.1 for zOS and OS/390 must be installed and bootstrapped. The job `BBOICD`, created by the ISPF customization dialogs, must have been run (the JCL job `BBOICD` is supplied in the `WP100277_B.zip` file)

## An Important Note about the Bitmap Illustrations Shown Here

This paper was written using the Application Assembly Tool (AAT) version 023 and the Systems Management End User Interface (SMEUI) version 4.01.006. Those are not the most recent versions of those two tools. Newer versions may (and probably will) have a slightly different look. *Therefore, you can not assume what you see on your system is exactly what is presented here.* But the concepts will be very much the same.

## Overview of Lab

This lab has five phases, and the flow of looks like this:

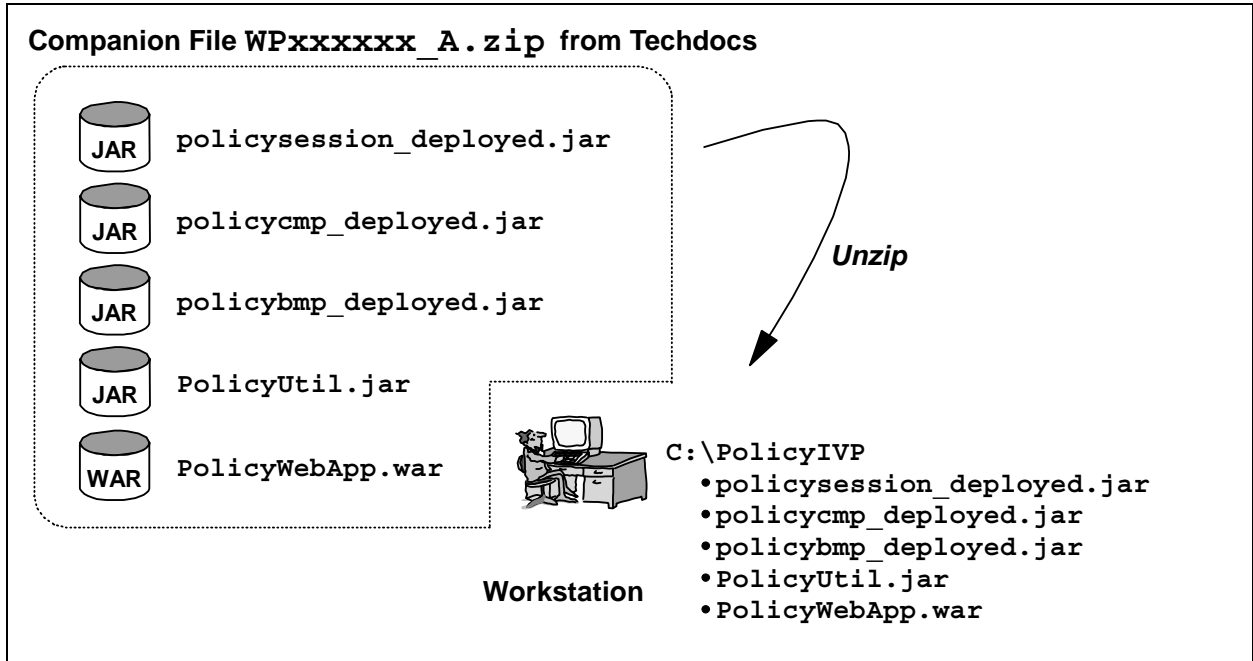


Boxes with the three question marks ("???) provide an explanation for what's going on. They do not contain "to do" activities, but do offer some background on why you're doing what you're doing.

## Phase 1: Get ZIP file from Website and Unzip to PC

??? The purpose of this phase is to get the JAR files and WAR file provided in the companion ZIP file onto your PC ready for use. Receiving a bundle of JAR/WAR files is similar to how an application developer might deliver an application to the "assembler."

### Overview



??? Where did these files come from? See "Reference: Where the JAR Files Came From" on page 31 and "Reference: Where the WAR File Came From" on page 34.

What are these files? They are:

- polycsession\_deployed.jar -- contains session bean after being deployed and exported from VisualAge for Java
- polycmp\_deployed.jar -- contains the BMP bean
- policybmp\_deployed.jar -- contains the CMP bean
- PolicyUtil.jar -- contains necessary utility files for the application
- PolicyWebApp.war -- contains the servlet that acts as the client to the session bean

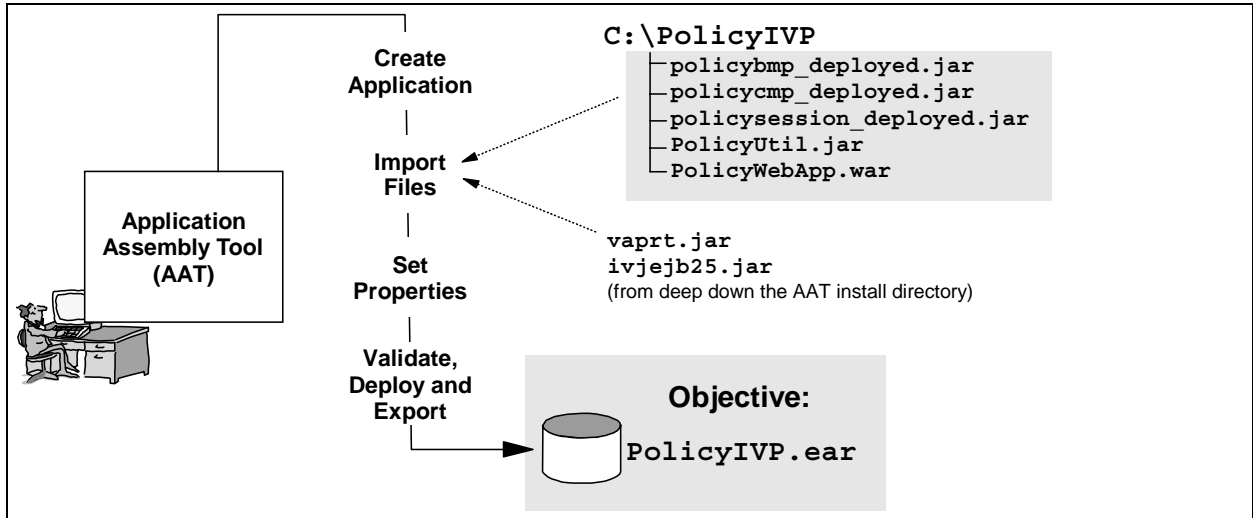
### Create PC Directory and FTP files

- Create a directory on your PC called C:\PolicyIVP
- Get the WP100277\_A.zip file from the Techdocs website.
- Unzip the contents of the file into the C:\PolicyIVP directory on your workstation.

## Phase 2: Generate EAR using AAT

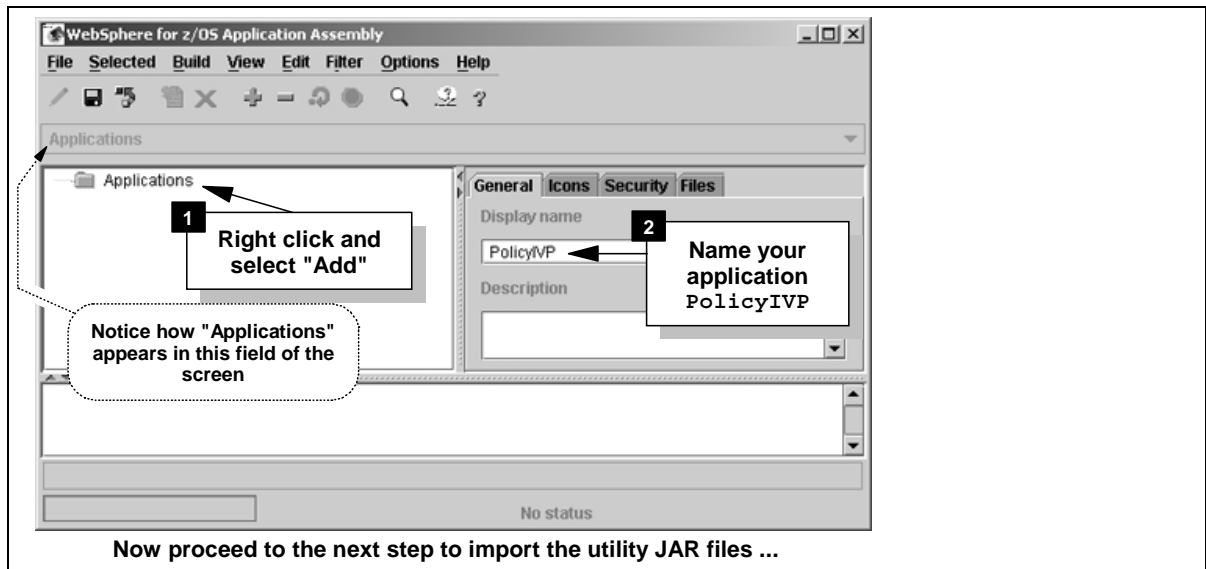
??? EAR files are another form of zipped archive file, and it is in EAR files that J2EE applications are packaged for deployment into the WAS 4.01 runtime. The tool used to generate the EAR file is the "Application Assembly Tool" (AAT).

### Overview



### Start AAT and Create the Application

- Select *Start* ⇒ *Programs* ⇒ *IBM WebSphere for zOS* ⇒ *Application Assembly*
- When AAT comes up, add an application called "PolicyIVP":

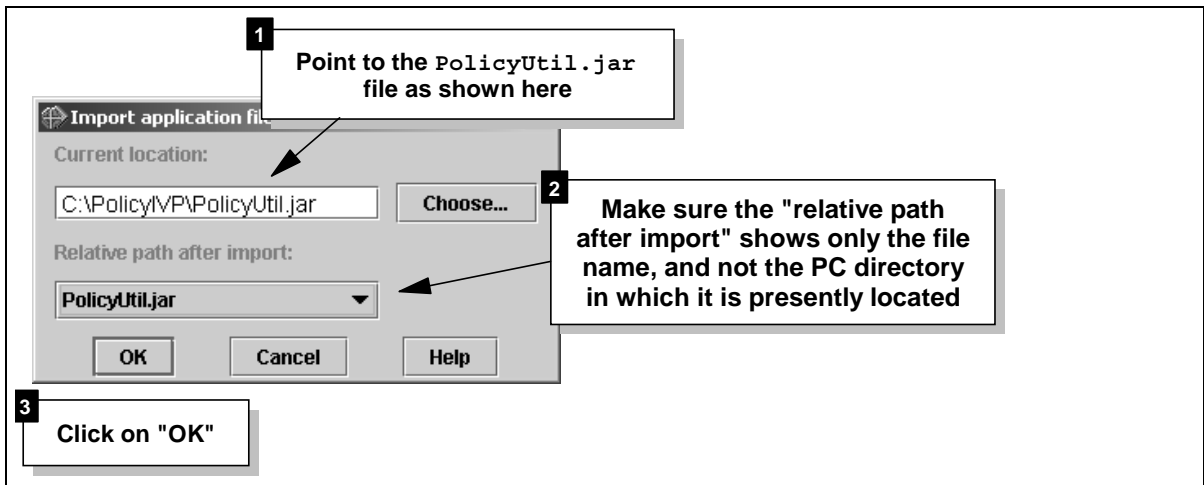


### Import the Utility Files

??? The application needs certain Java utility class files to run. One file was exported from VAJ (PolicyUtil.jar). The other JAR files you'll get from the directory in which the AAT tool itself is installed (vaprt.jar and ivjejb35.jar).

- Click on the "Files" tab immediately above where you entered the PolicyIVP application name, and then click on the button marked "Import". You'll get a small pop-up panel that looks like what follows. Perform the tasks shown in the following picture:

## WP100277 - PolicyIVP Construction Lab



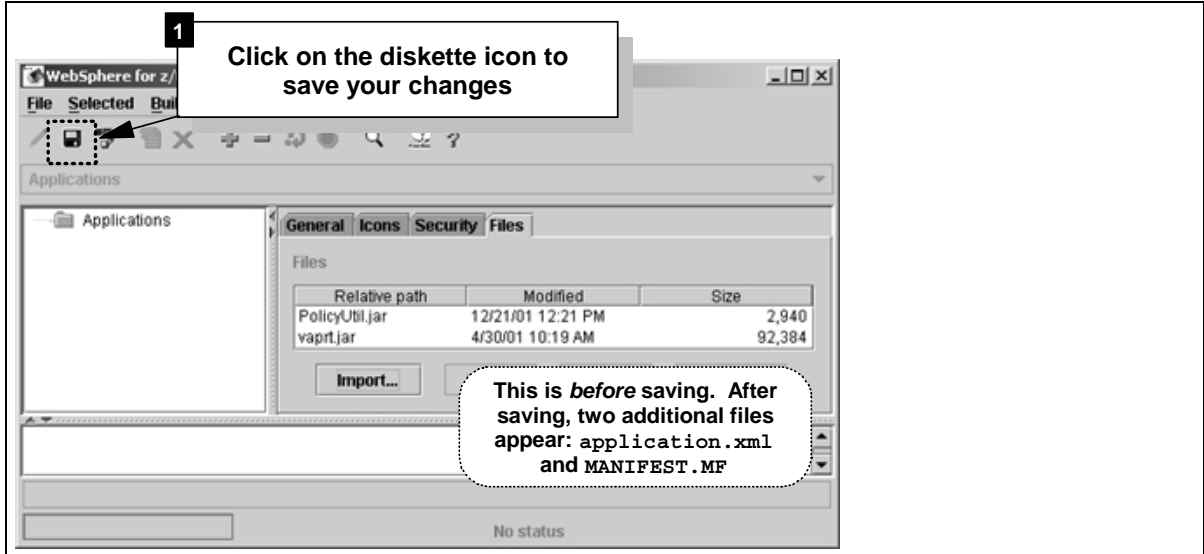
- Click on the "Import" button again, and this time bring in the following file:

C:\Program Files\IBM\WebSphere for zOS Application Assembly\config\CommonJars\vaj35\vaprt.jar

**Hint:** You may find it easier to click on the "Choose" button and then navigate the tree structure to get that file, rather than typing out the whole name.

Make sure the "Relative path after import" shows just the file name `vaprt.jar`, and not that long directory name.

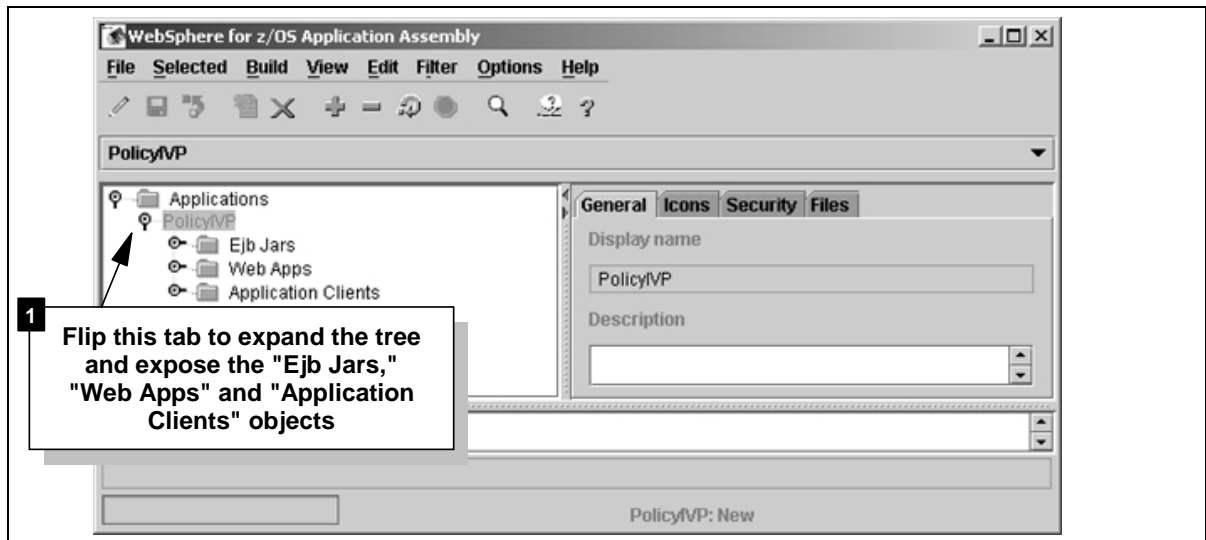
- Click on the "Import" button one last time and bring in the file `ivjejb35.jar`, which is in the same deep directory where the `vaprt.jar` file was.
- Your AAT panel should now look like this. Save the changes as shown in the picture:





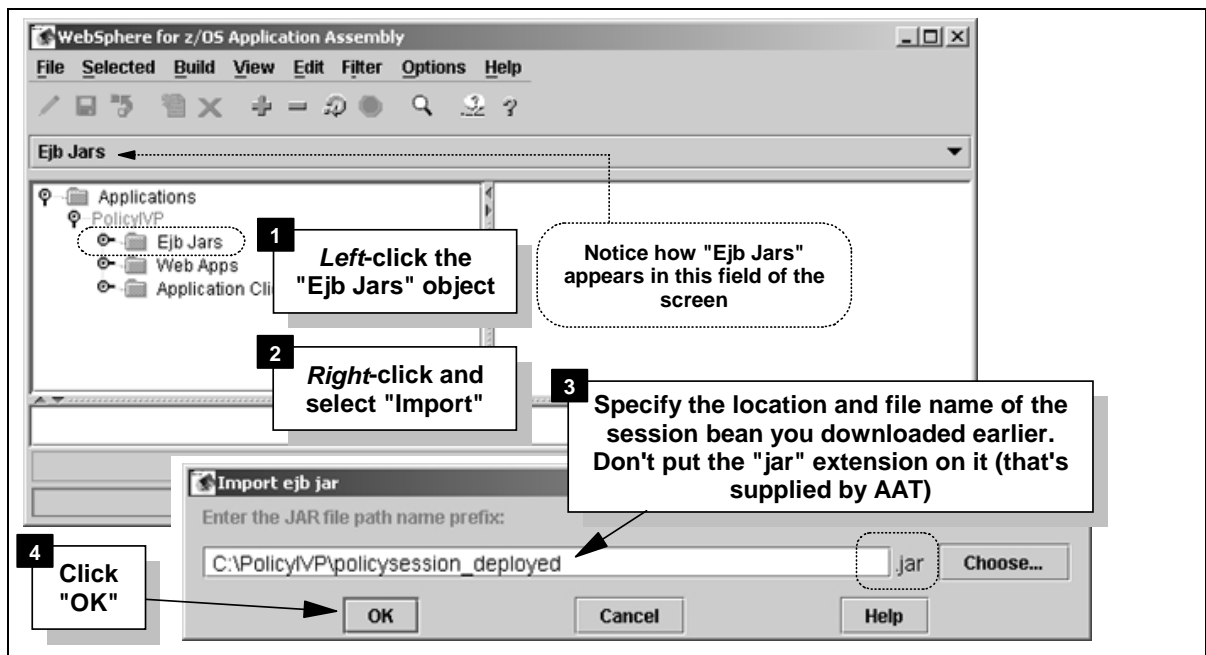
## WP100277 - PolicyIVP Construction Lab

- You should see the PolicyIVP application now appear below "Applications". Expand the tree below "PolicyIVP" by clicking on the tab next to the name:



### Import the EJB JAR Files

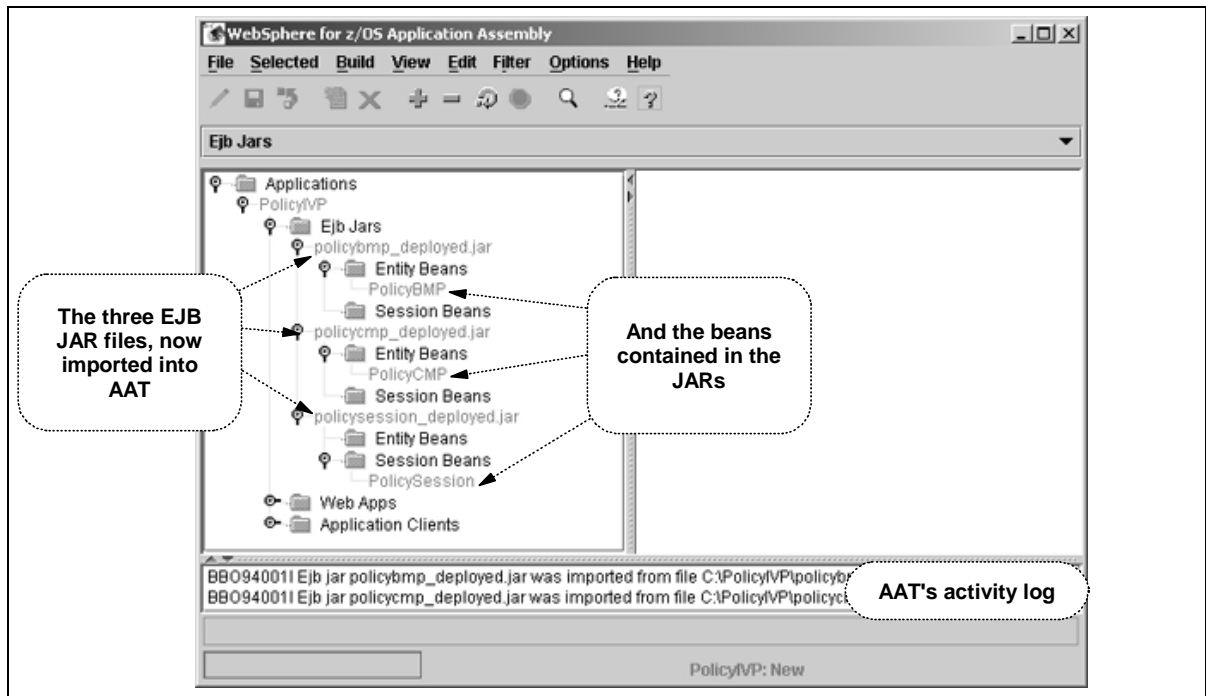
- Do the following to import the *session bean*:



- Repeat that process twice more (selecting "Ejb Jars" each time) and bring in the two entity beans:
  - The CMP entity bean (`polycymp_deployed.jar`)
  - The BMP entity bean (`policybmp_deployed.jar`)

When you're done, the AAT panel should look like this after you expand out all the tabs on the panel:

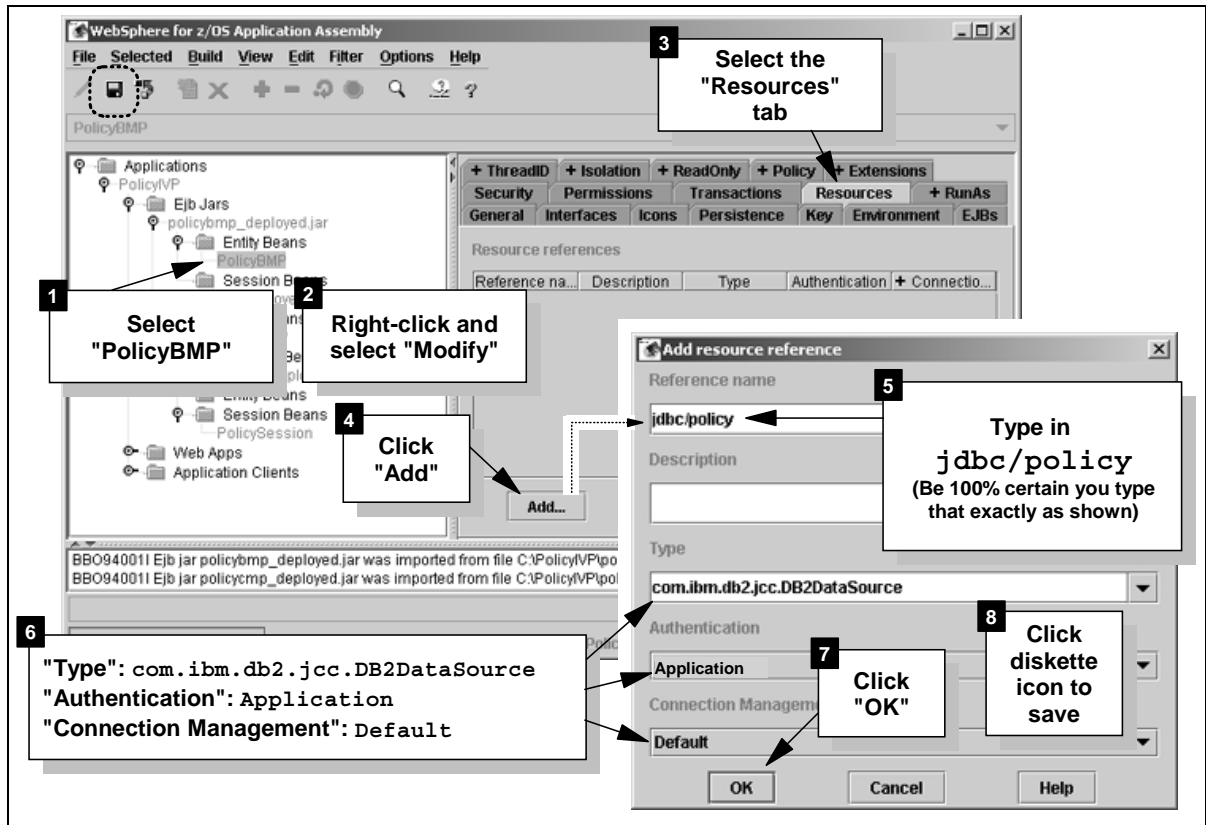
## WP100277 - PolicyIVP Construction Lab



### Modify the Properties of the EJBs

#### BMP Entity Bean

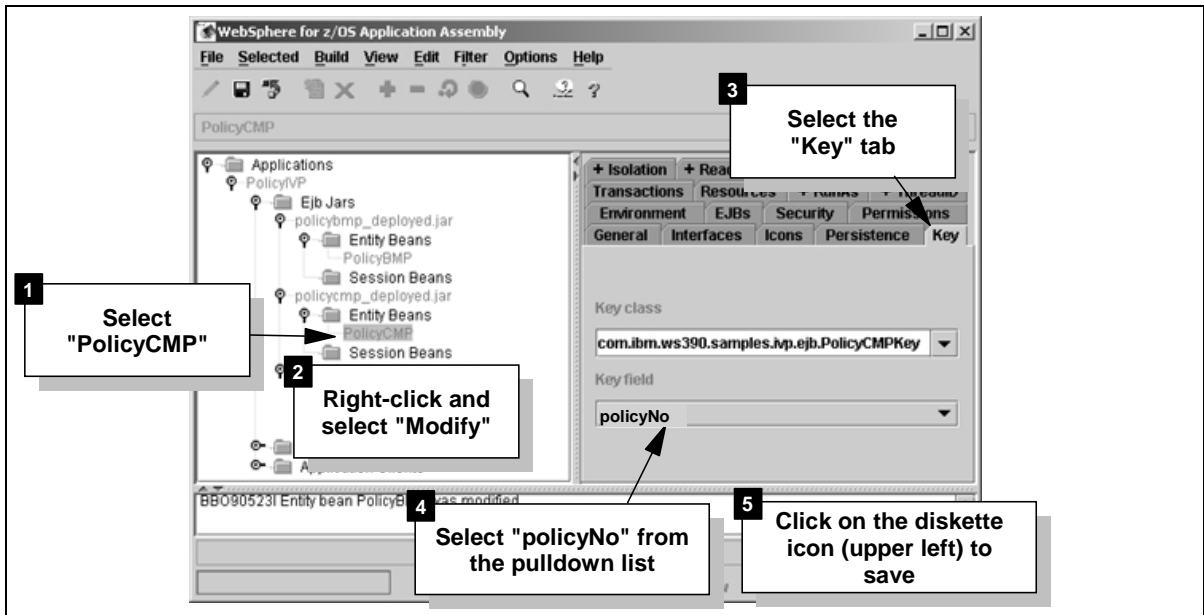
- Do the following:



??? The BMP entity bean uses `java:comp` to find its data source (in this case, DB2), which it references by using `java:comp/env/jdbc/policy`.

### CMP Entity Bean

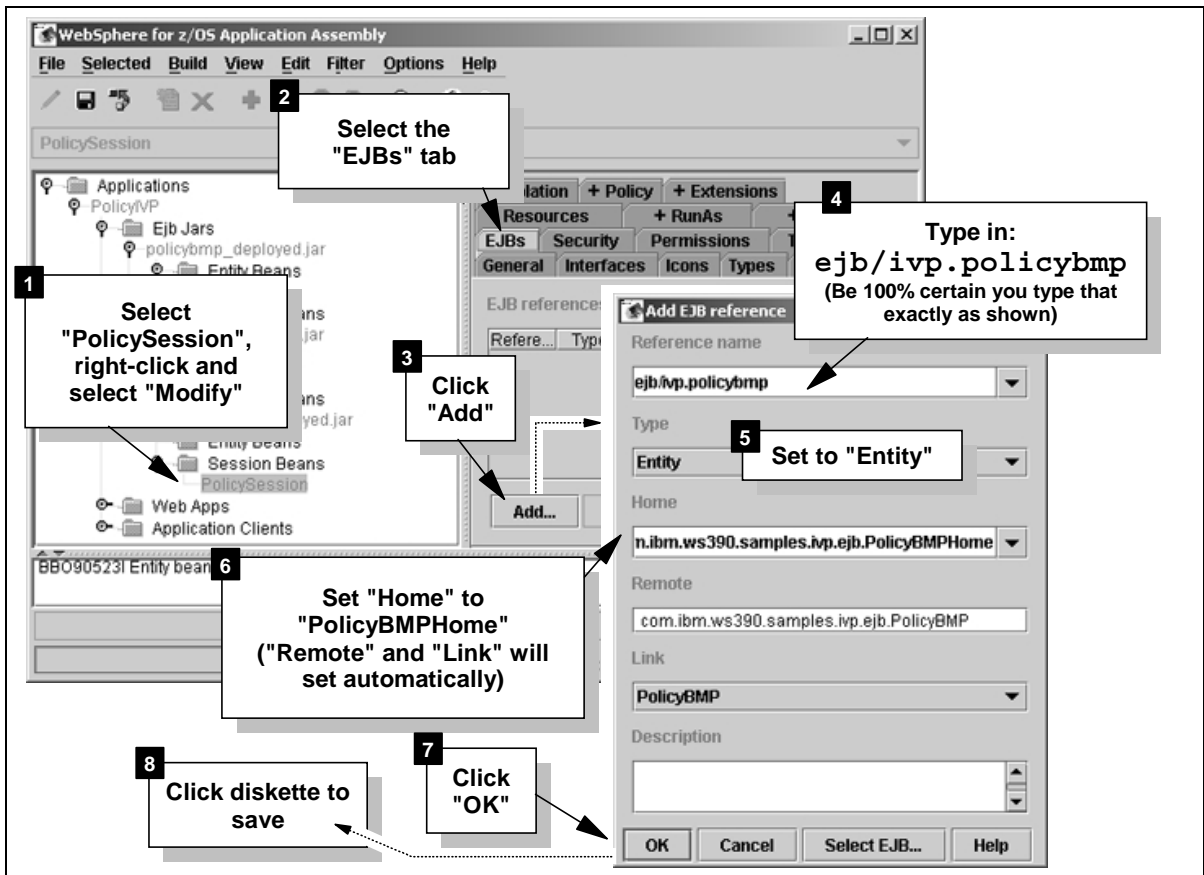
- Do the following:



??? The CMP entity bean needs to have its primary key field set to function properly. The tasks shown above had you choose the database column "policyNo" as the primary key.

### Session Bean

- Do the following to set the reference to the BMP entity bean:



## WP100277 - PolicyIVP Construction Lab

??? The session bean needs to know how to get to the two entity beans when its programming calls for it to store data. To do that, you tie the "java:comp" lookup string used in the application to the "home interface" of the entity bean. Later, when you deploy the application, the JNDI name of the entity bean will be resolved and the session bean will then be able to lookup and find the entity bean.

You just set the information for how to reach the BMP bean. Now you have to do the same thing for the CMP bean.

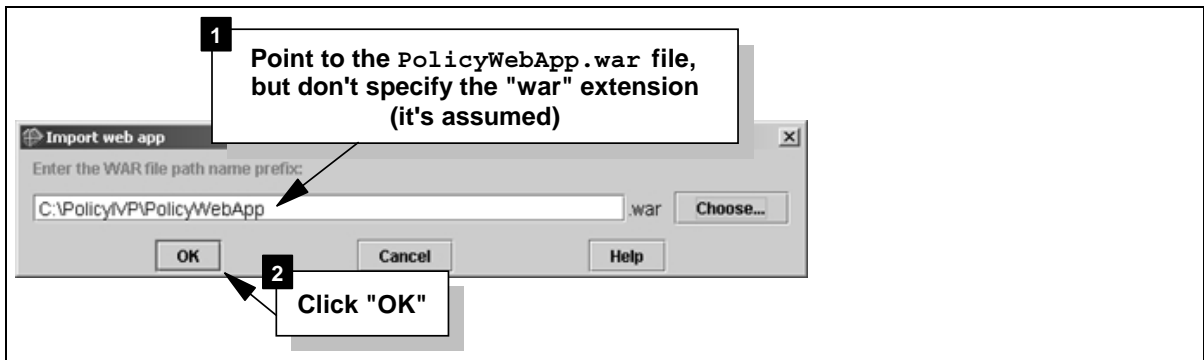
- ❑ **Repeat** the *previous eight steps*, only this time provide information for the CMP bean. Be certain to provide the "Reference Name" as `ejb/ivp.policycmp`. The naming convention is nearly identical, except for "CMP" vs. "BMP".

### ✓ **Validation Activity**

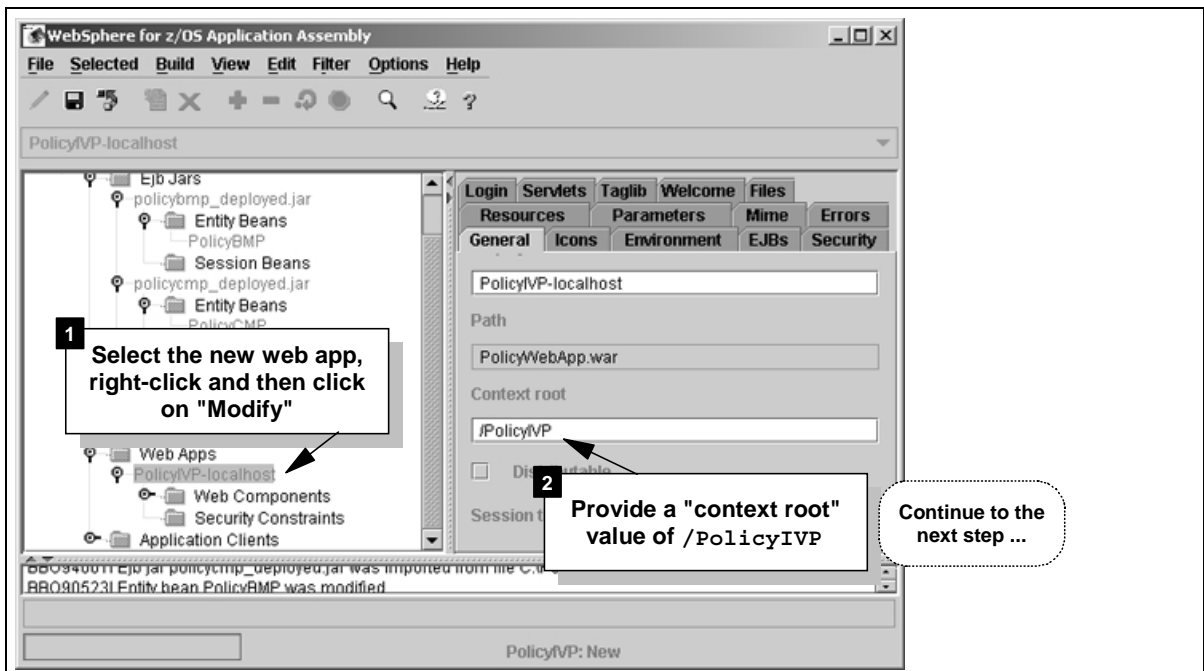
- ❑ Did you create an EJB reference from the Session bean to *both* the BMP bean *and* the CMP bean?

### **Import the Web Application and Modify its Properties**

- ❑ Select the "Web Apps" folder in AAT, right-click and select "Import":



- ❑ Once the WAR file is imported, do the following:



## WP100277 - PolicyIVP Construction Lab

??? The "context root" value is comparable to the "root URI" value from WAS 3.5 Standard Edition. It is used to equate an inbound URL with an application. In a later lab you will use a URL of *something* like `http://www.your_host.com/PolicyIVP/...` to drive this application. WAS will use this context root to know that URL is asking for *this* application.

□ And then do the following:

1 Select the "EJBs" tab

2 Click "Add"

3 Reference name: `ejb/ivp.policysession`

Type: `ejb/ivp.policysession`

4 Set to "Session"

5 Set "Home" to `PolicySessionHome`. Others values set automatically

6 Click "OK," then click on diskette icon to save

You should still be in your modify session on PolicyIVP-localhost

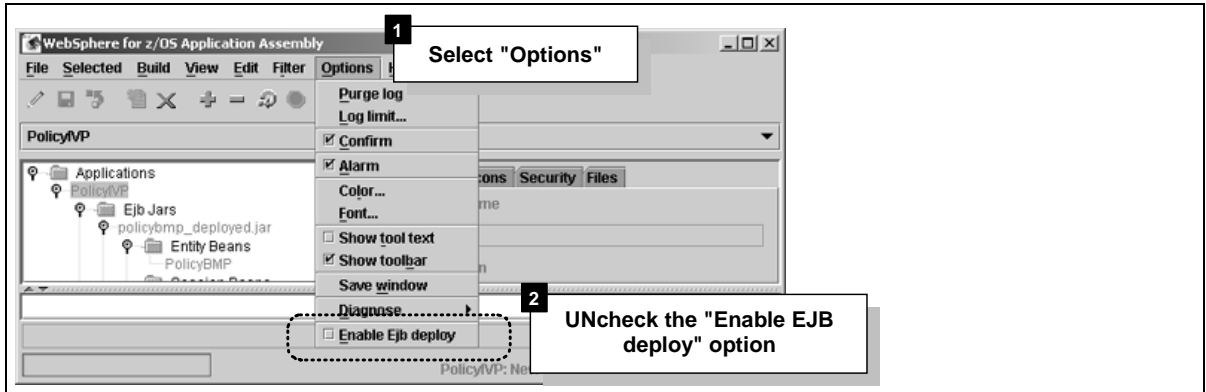
??? The servlet will be the client in the "client/server" relationship, and it needs to know about the session bean to which it'll attach. This process provided the reference name, and then linked that name to the Home and Remote interfaces of the PolicySession bean. This process is identical in purpose to what you did earlier to connect the session bean with the two entity beans.

**Validate, Deploy and Export the Application**

??? The goal here is to produce an EAR file that can then be deployed to the WAS 4.01 runtime. Before the AAT tool will allow you to export the EAR, you must first "validate" the application (AAT checks syntax, etc.), then "deploy" it (AAT updates some files and marks application ready for export).

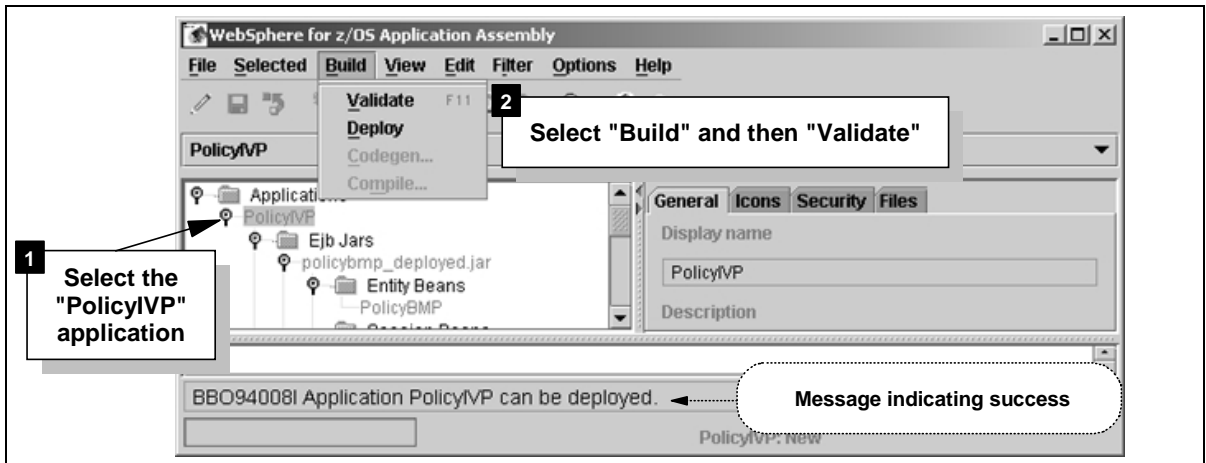
**Validate**

- ❑ Disable the "EJB Deploy" option:



??? You need to instruct AAT to *not* generate the stubs, ties and persister classes. The application developer did that when the JAR files were created. (That's why the JAR file names have "\_deployed" as part of their name.) You *could* have AAT regenerate the code, but it would take extra time and is not necessary.

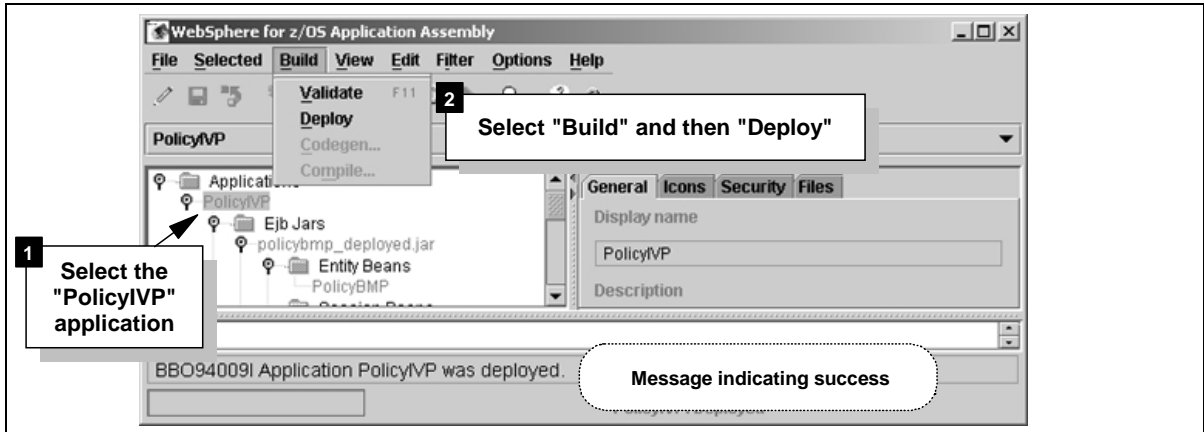
- ❑ Now do the following:



??? Validating an application involves the AAT performing a quick check on syntax and other settings. Once validated, the application is eligible for deployment, which comes next.

**Deploy**

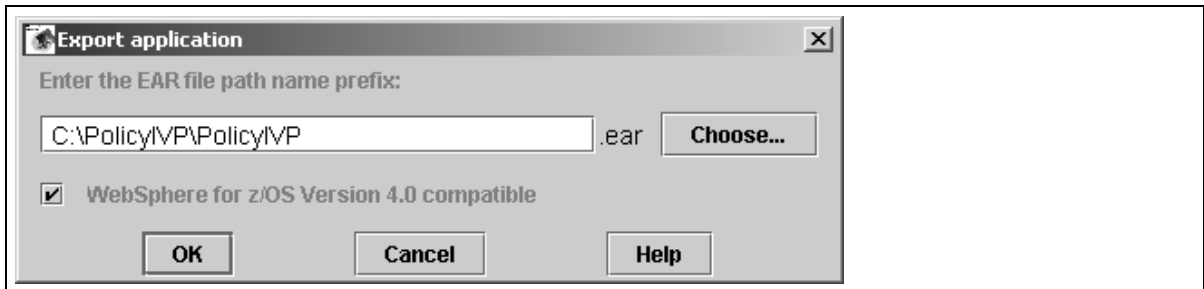
- Do the following:



??? Deploying is the final step prior to exporting the application to an EAR file format. When you "deploy" an application in AAT, the tool does the final checking to insure all the pieces are there to construct an EAR file.

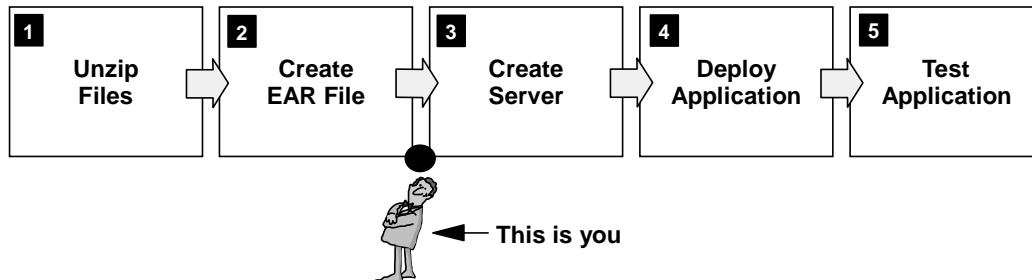
**Export**

- Select the "PolicyIVP" application again, right-click on it and then select "Export"
- Point to your C:\PolicyIVP directory and give the EAR file you are about to create a name of PolicyIVP (you don't need to specify the "ear" extension; that's automatic):



- Click on "OK" and the EAR file will be created.
- Close the AAT tool.
- (Optional) Use a ZIP tool to review the contents of the EAR file.

**Review**

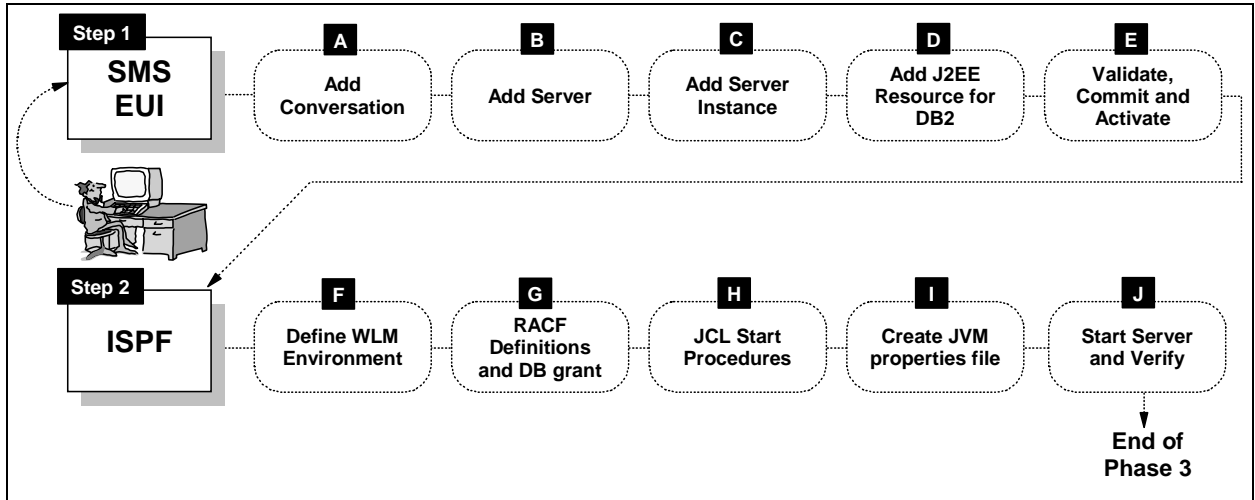


??? You are done with the AAT. With an EAR file on your C: drive, you're now ready to use the WebSphere 4.0.1 Systems Management interface tool to install the new application into the J2EE server environment.

## Phase 3: Create Application Server

??? You may already have a J2EE application server created on your WebSphere system. In this phase you will create a *new* application server -- APSRV3 -- and in Phase 4 deploy the EAR file you created into that server.

### Overview



### Start SMS EUI and Add Conversation

- Start the SMS EUI program and connect to the server using the WASADM1 userid with password WASADM1 (or whatever userid and password is defined for your system).
- Do the following:

**1** Right-click the "Conversations" folder

**2** Click "Add"

**3** Name the conversation and provide a brief description

**4** Click on the diskette icon (highlighted above) and be patient while the conversation is saved. You will then see the following:

Your new conversation added to the list. It's not yet active, but it's now known to the server

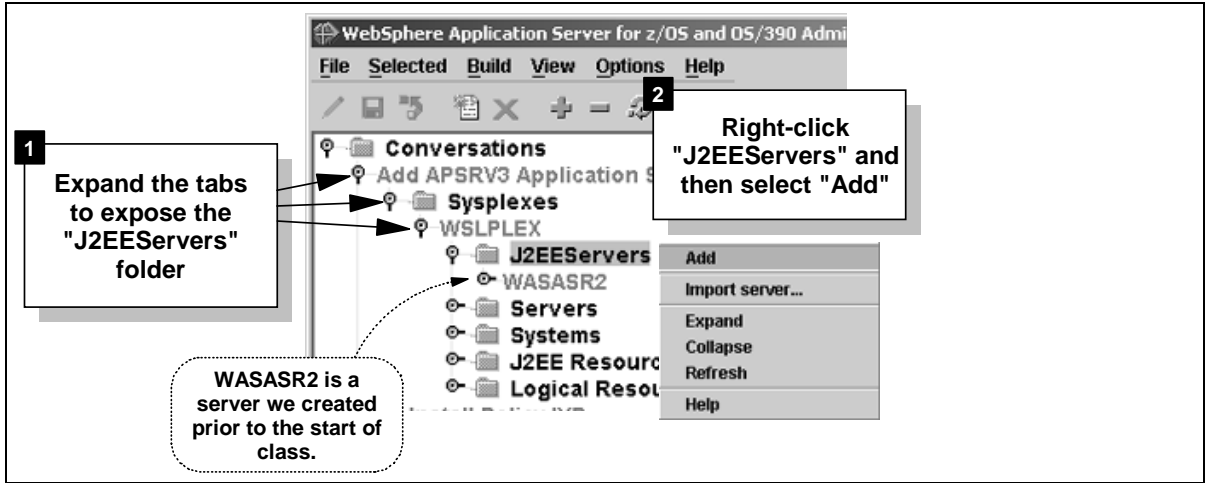
Success message

BBON0515I Conversation Add APSRV3 Application Server was added.



**Add Server**

- Start this process by doing the following:



- The right-hand side of the SMS EUI panel will now be a scroll-pane with many input fields. Fill in each field with the information provided here:

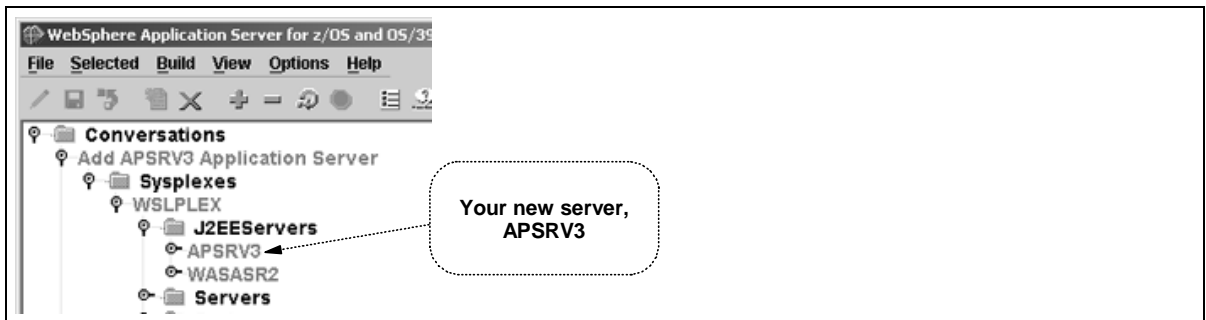
??? It's important that you use the values as provided here. If you change a value here, you will need to change the jobs provided in WP100277\_B.zip as well as the step in this lab where the WLM application environment is established. There's a linkage between all the parts.

Field	Value
Server Name	APSRV3
Server Description	(whatever you wish to add)
Control Region Identity	APSRV3C
Server Region Identify	APSRV3S
Server Region Stack Size	0 (the number zero)
Production J2EE Server	(check)
Debugger Allowed	(accept default)
Object Level Trace Hostname	(accept default)
Object Level Trace Port	(accept default)
Isolation Policy	select "Multiple transactions per server"
Replication Policy	select "One per server"
Local Identity	APSRV3D
Remote Identity	APSRV3I
Register Transaction Factory	(accept default)
Allow Server Region Recycling	(accept default)
Server Recycling Interval	(accept default)
Logstream Name	(accept default)
Control Region Proc Name	APSRV3C
Enable Setting OS Thread Identity to RUNAS	(accept default)
Allow Non-Authenticated Clients	(check)
Userid Password Allowed	(check)
Userid Passticket Allowed	(check)
DCE Allowed	(accept default)
DCE Quality of Protection	(accept default)
DCE Keytab File	(accept default)
SSL Type 1 Allowed	(accept default)
SSL Client Certificates Allowed	(accept default)
Kerberos Allowed	(accept default)
Send Asserted Identities Allowed	(accept default)
Accept Asserted Identifies Allowed	(accept default)

## WP100277 - PolicyIVP Construction Lab

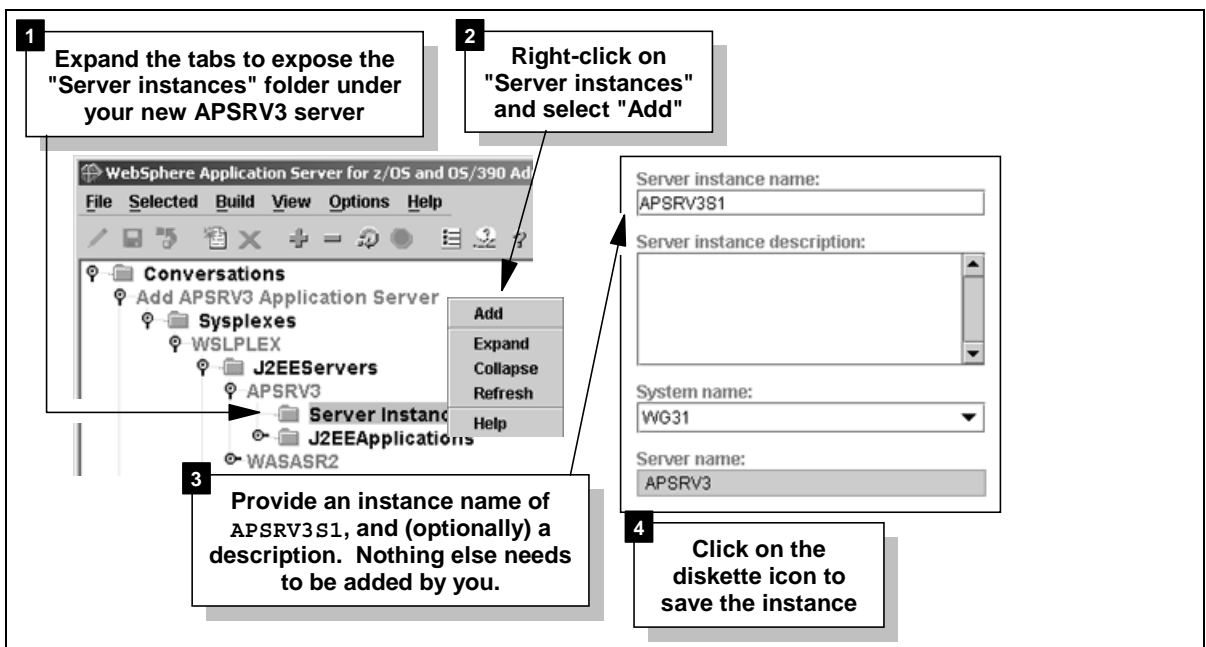
Field	Value
SSL Use Confidentiality Only .....	<i>(accept default)</i>
SSL RACF Keyring .....	<i>(accept default)</i>
SSL V2 Timeout .....	<i>(accept default)</i>
SSL V3 Timeout .....	<i>(accept default)</i>
<i>Security Preference List:</i>	
Password Order .....	1
Passticket Order .....	2
(All other entries in Preference List): .....	<i>(leave blank)</i>
Write Server Activity SMF Records .....	<i>(accept default)</i>
Write Container Activity SMF Records .....	<i>(accept default)</i>
Write Server Interval SMF Records .....	<i>(accept default)</i>
Write Container Interval SMF Records .....	<i>(accept default)</i>
Interval Length .....	<i>(accept default)</i>
Environment Variable List	<i>(leave blank ... values from sysplex level set during bootstrap used for this server)</i>

- ❑ Click on the *diskette icon* to save your changes. Be patient while the server saves all the changes. It may take a minute or so. When it is complete, you should see your server in the "tree:"



### Add Server Instance

- ❑ Do the following:



??? Why a name of APSRV3S1? The name could be anything you like, but it should be related to your server name, which was APSRV3. We are using a naming convention for servers of APxxxx, where xxxx is a four-character string of your choosing ("SRV3" for this lab). The format for the instance name is APxxxxSy, where xxxx is the same four-character string as used for the server, and "y" is a number starting at 1 for the first instance, 2 for the second, etc.

**Add J2EE Resource and Resource Instance for DB2**

- Add the "resource" first:

**1** Expand the tabs to expose the "J2EE Resources" folder under the WSLPLEX folder

**2** Right-click and select "Add"

**3** Name your resource anything you'd like (no blank spaces)

**4** Select DB2datasource from the list

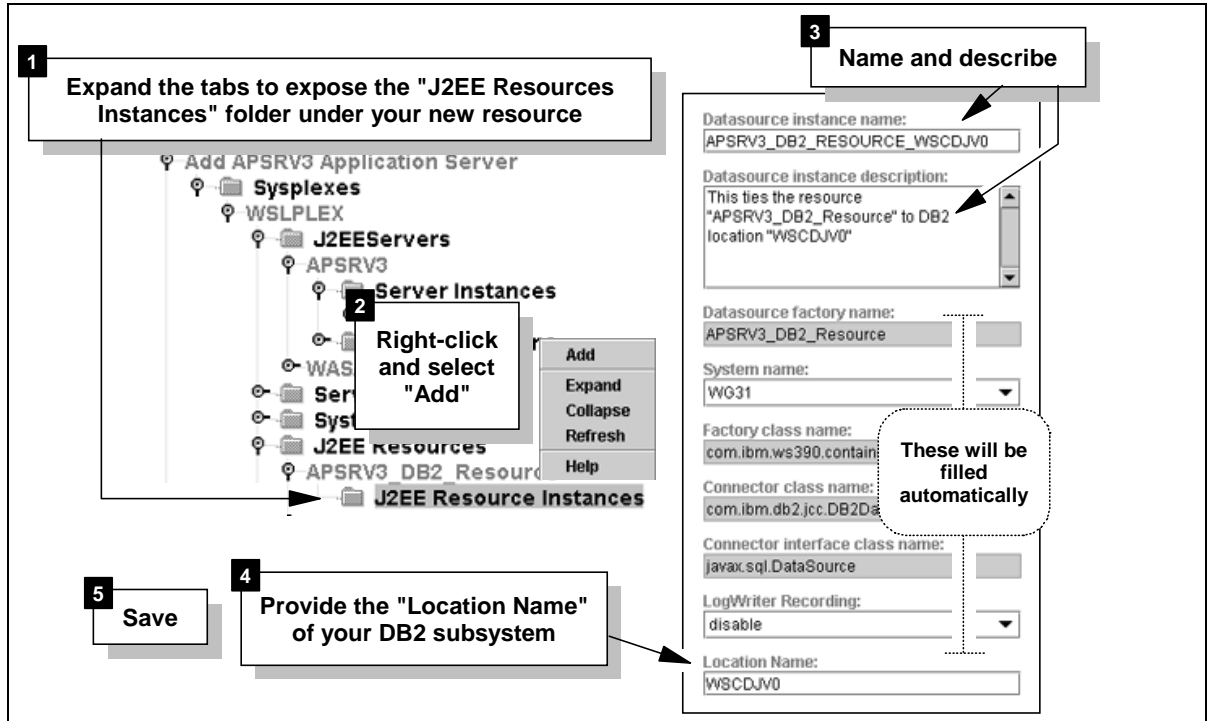
**5** Click diskette icon to save

Other resources may already be defined. That's okay; you'll define your own for this lab

These will be filled in for you based on your selection from the "J2EE Resource type" list below

??? Depending on who has done what with your copy of WebSphere prior to you doing this lab, there may already be a J2EE Resource defined for your DB2 subsystem. Creating another pointer to the same DB2 subsystem won't hurt anything, and it's good practice.

- Now add the "resource instance":



??? WSCDJV0 is the "group attach name" for the DB2 subsystem DJV0 on the class system. Use whatever "group attach name" you have defined for your copy of DB2.

**Validate, Commit and Activate the Conversation**

- Highlight the conversation name in which you're presently working and select *Build* ⇒ *Validate* from the menu bar. If all is proper with the conversation, you'll see:

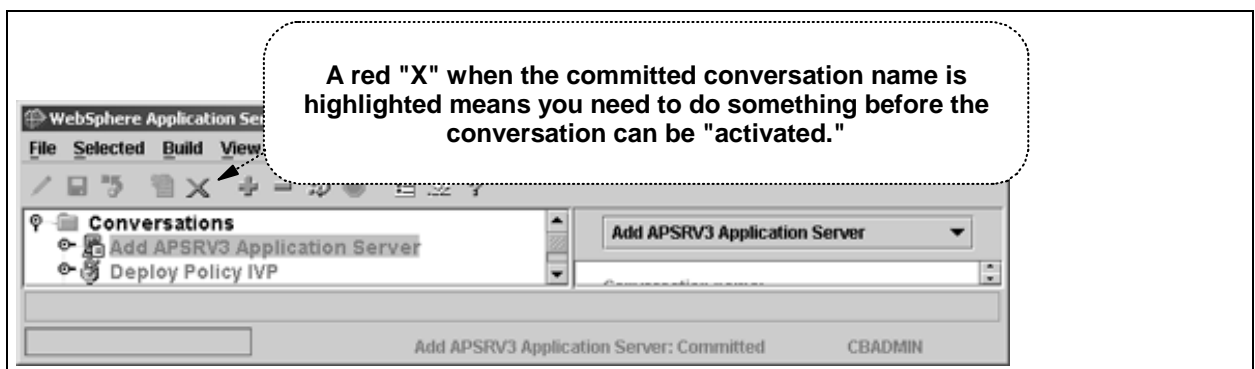
BBON0442I Conversation Add APSRV3 Application Server is valid

- Keep the conversation name highlighted and select *Build* ⇒ *Commit* from the menu bar.

??? Once a conversation has been committed, no further changes can be made to it. The SMS EUI will ask "Are you sure?" to make certain it's okay to commit the changes you entered.

BBON0444I Conversation Add APSRV3 Application Server was committed

At this point your SMS EUI panel should look something like this:



## WP100277 - PolicyIVP Construction Lab

- ❑ Keep the conversation name highlighted and select *Build* ⇒ *Complete* ⇒ *All Tasks* from the menu bar. Answer "Yes" to the question "Are you sure...?"

??? The tool wants to validate that you've done all the non-GUI tasks, such as defining the WLM application environment, creating the JCL start procedures and defining the RACF definitions. *You haven't yet done those activities* (that's the next step), but you'll tell an un-truth to the tool and say you have.

- ❑ Your conversation name should have a check mark next to it, indicating it's ready to be activated. Select *Build* ⇒ *Activate* from the menu bar. Answer "Yes" to the question "Are you sure...?"

??? A great deal of system activity occurs on the server when a conversation is activated. This process takes time. Be patient.

Once activated, the conversation should have a "lock with a lightning bolt" icon next to it:



### Perform non-GUI Tasks

??? Though the conversation has been activated, much more needs to be done. In this section you'll go to ISPF and do some system work to define the WLM application environment, the RACF definitions and the JCL start procedures.

### Define WLM Application Environment

??? WLM will start additional *server regions* as the workload requires.

- ❑ Log onto TSO using an ID that has authority to access WLM.
- ❑ Enter `WLM` at the ISPF main panel command line, and then press "Enter" to get past the initial screen.
- ❑ When prompted, select Option 2, "Extract Definition from WLM couple data set."
- ❑ When the definitions have been extracted, select Option 9, "Application Environments."
- ❑ Place a "2" (for "Copy") next to one of the other WebSphere definitions and hit "Enter"
- ❑ Now create your new environment:

WP100277 - PolicyIVP Construction Lab

Application-Environment Notes Options Help

-----  
Copy an Application Enviro

Command ==> \_\_\_\_\_

Application Environment . . . APSRV3 Required

Description . . . . . Application Server APSRV3

Subsystem Type . . . . . CB

Procedure Name . . . . . APSRV3S

Start Parameters . . . . . IWMSNM=&IWMSNM

Limit on starting server address spaces for a subsystem instance:

1 1. No limit

2 2. Single address space per system

3 3. Single address space per sysplex

F1=Help F2=Split F3=Exit F4=Return F7=Up F8=Down  
F9=Swap F10=Menu Bar F12=Cancel

**1** Provide a definition name of APSRV3 and then provide a description

**2** Provide a proc name of APSRV3S (note the "S" on the end of that)

**3** Press F3 to save and exit

The other fields should have the same values copied over from WASASR2

- Press F3 *again* to get to the "Definition Menu" panel. Select the "Utilities" pulldown and then the "Validate" selection.
- Select the "Utilities" pulldown again and select "Install"
- Select the "Utilities" pulldown once again and select "Activate"
- When asked *which* policy to activate, and put a slash next to the appropriate value for your system and then press "Enter."
- Press F3 to exit.
- Go to the SDSF LOG and verify your WLM application environment is there. Issue the following command:

```
/D WLM,APPLENV=*
```

You should see your WLM environment "available":

```
IWM029I 14.52.48 WLM DISPLAY 977
APPLICATION ENVIRONMENT NAME STATE STATE DATA
APSRV3 AVAILABLE
WASASR2 AVAILABLE
CBINTFRP AVAILABLE
CBNAMING AVAILABLE
CBSYSMGT AVAILABLE
```

If the APSRV3 application environment is not there, go back into WLM and repeat the "Install" and "Activate" steps.

**Run RACF Batch Job**

??? We are supplying you with a simple RACF job and script. There is a way to generate this script using the ISPF Customization dialogs, but this is a simpler way to achieve the same end result. In this step you will create two data sets, store the supplied JCL and referenced RACF script, and then submit the JCL job.

- Create a small Fixed Block 80 PDS and give it a name of (hlq).CNTL.
- Create a small Variable Block 255 PDS and give it a name of (hlq).DATA

## WP100277 - PolicyIVP Construction Lab

- From the file WP100277\_B.zip, unzip the file RACFJCL and place it in the FB 80 data set as member RACFJCL.
- From the same WP100277\_B.zip, unzip the file APPLRAC and place it in the VB 255 data set as member APPLRAC.
- Edit hlq.CNTL (RACFJCL) and make sure the JOB card is okay for your system, and make sure the SYSEXEC statement points to hlq.DATA (the VB 255 data set) properly.
- Now submit the job hlq.CNTL (RACFJCL), which calls the RACF CLIST hlq.DATA (APPLRAC). Check the output to make sure all the commands ran without failure.

### Grant Server Region ID Access to PolicyIVP Database Table

??? When the PolicyBMP or PolicyCMP beans try to access the DB2 table (BBO.POLICYDO), it needs to have INSERT, UPDATE, DELETE and SELECT authority on the table to do its job. If you don't grant this authority, the access fails with a SQL -551.

- From the file WP100277\_B.zip, unzip the file BBOIGRT and place it in the hlq.CNTL dataset.

??? By the way, this member BBOIGRT is one that is created when you run the ISPF Customization Dialogs. If that's been done on your system -- and it must have been, otherwise you wouldn't have a bootstrapped WebSphere and you wouldn't have gotten this far -- then you have that member on your system already. If you know where the output data set is from the installation of WebSphere, then go get that copy of BBOIGRT and use it.

- Edit the member and reduce it just this portion (remove the MOFW half at the bottom). Then *update the userids* on the "TO" portion of the GRANT statement to be that of your server region's ID (APSRV3S):

## WP100277 - PolicyIVP Construction Lab

```
***** ***** Top of Data *****
000001 //BBOIGRT JOB 1,MSGLEVEL=1,USER=SYSADM1,PASSWORD=SYSADM1
000002 // SET DB2='DSN710'
000003 //* See instructions at the bottom of this file.
000004 //*=====
000005 //* The GRANTS in the BBOIGSV step are the GRANT statements needed
000006 //* for a J2EE application server. The GRANT needs to be made for
000007 //* the identity associated with BBOASR2 server. The second GRANT
000008 //* would only be needed if your J2EE application server has session
000009 //* beans deployed in that server.
000010 //*=====
000011 //BBOIGJS EXEC PGM=IKJEFT01,DYNAMNBR=20
000012 //*STEPLIB DD DSN=&DB2..SDSNLOAD,DISP=SHR
000013 //DBRMLIB DD DSN=&DB2..SDSNDBRM,DISP=SHR
000014 //SYSTSPRT DD SYSOUT=*
000015 //SYSUDUMP DD SYSOUT=*
000016 //SYSPRINT DD SYSOUT=*
000017 //SYSTSIN DD *
000018 DSN SYSTEM(DJV1)
000019 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA71) -
000020 LIB('WSCDJV1.DJV1.RUNLIB.LOAD')
000021 END
000022 //SYSIN DD *
000023 GRANT INSERT,SELECT,DELETE,UPDATE
000024 ON BBO.POLICYDO
000025 TO APSRV3S;
000026
000027 GRANT INSERT,SELECT,DELETE,UPDATE
000028 ON BBO.STATEFUL_BEANS
000029 TO APSRV3S;
000030 /*
***** ***** Bottom of Data *****
```

??? BBO.POLICYDO is the simple table defined for use by the PolicyIVP application. The table BBO.STATEFUL\_BEANS is a WAS 4.01 system table used by stateful session beans. The PolicyIVP session bean is *not* a stateful bean, so in truth granting APSRV3S access to the table wasn't strictly required. It didn't hurt, though.

- Submit the job and make sure it ran okay.

??? You'll probably have to check with your DB2 administrator to make sure the ID under which this runs has proper authority. The JCL provides a userid and password on the JOB statement, but your system most likely doesn't have a userid of SYSADM1 with a password of SYSADM1.

### Create JCL Start Procedures

- From the WP100277\_B.zip file, place the files APSRV3C and APSRV3S into one of your proclibs.

??? You named the *control region* procedure in the SMS tool when you created the server, and you named the *server region* procedure in WLM when you defined the application environment. The names you provide these procs does matter: it must match the other definitions.

**Note:** The ISPF Customization Dialogs provide the same sample JCL start procs. If you know where the output data set for the dialogs is kept on your system, copy BBOASR2 and BBOASR2S from that data set over to your proclib. The files APSRV3C and APSRV3S in the WP100277\_B.zip file are simply copies of BBOASR2 and BBOASR2S.



- Now edit and modify the new APSRV3C procedure (the control region proc):

**1** Control Region Proc Name

```

//APSRV3C PROC SRVNAME=,
//      PARMS=' '
//  SET RELPATH='controlinfo/envfile'
//  SET CBCONFIG='/WebSphere390/WAS401'
//APSRV3C EXEC PGM=BBOCTL,REGION=0M,
// PARM='/ -ORBsrvname &SRVNAME &PARMS'
//BBOENV DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&SRVNAME/current.env'
//CEEDUMP DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSOUT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSPRINT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//
(comment removed to save space)

```

**2** Clear SRVNAME= value so it appears as shown here

**3** Point to config directory

??? What's a "config directory?" At initial configuration time, the person installing WebSphere provided an HFS directory under which all the WebSphere configuration data (server and application) would be stored. Whatever value that person supplied at installation time needs to be supplied here as well.

??? Why clear the SRVNAME= field? The value of this symbolic is the server *instance* name to be used. This will be passed in on the start command. With this cleared out, if you fail to pass in a SRVNAME= value on the start command a JCL error will result. You could hard-code your instance name in the JCL if you wish. But by leaving it blank you are forcing yourself to consciously provide a valid instance name on the start command.

- Edit and modify the new APSRV3S procedure (the server region proc):

**1** Your Server Region Proc Name

```

//APSRV3S PROC IWSSNM=,PARMS='-ORBsrvname '
//  SET CBCONFIG='/WebSphere390/WAS401'
//  SET RELPATH='controlinfo/envfile'
//APSRV3S EXEC PGM=BBOSR,REGION=0M,TIME=1
// PARM='/ &PARMS &IWSSNM'
//STEPLIB DD DISP=SHR,DSN=WAS401.WAS.SBBOULIB
//BBOENV DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&IWSSNM/current.env'
//CEEDUMP DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//
(comment removed to save space)

```

**2** Clear IWSSNM= field so it appears as shown here

**3** Point to config directory

??? Why clear the IWSSNM= field? The value of this symbolic is the server *instance* name to be used. This will be passed in by WLM when WLM starts the server region. You could hard-code your server instance name in the JCL if you wish. The value passed in by WLM will simply override it.

**Start Server and Verify Registration**

- At the SDSF LOG screen, issue the command:

```
/S APSRV3C.APSRV3S1,SRVNAME=APSRV3S1
```

??? Why such a long start command? The pieces of this command are as follows:

**/S APSRV3C.APSRV3S1,SRVNAME=APSRV3S1**

- The procedure member name
- The procedure identifier
- Symbolic variable name
- Symbolic value, which is the server instance to be started

You supply a procedure identifier equal to the server instance name because that's what WebSphere uses when it stops and restarts servers automatically (which it does when you do things like deploy another application into a server, or change a key setting for a server). If you were to start the server with *just* the APSRV3C name, the stop command issued by WebSphere would fail. The symbolic SRVNAME= is being passed in because the JCL you just created depends on you passing in this variable in order to properly start the server.

- Browse through the log looking for the following indications of success in the order presented:

```
BBOU0020I INITIALIZATION COMPLETE FOR CB SERIES CONTROL REGION 622
```

??? Indicates the control region is started.

```
+BBOU0694I NAMING REGISTRATION STARTED FOR SERVER APSRV3
```

??? The process of registering the new server into the naming space has started

```
+BBOU0004I CB SERIES SERVER APSRV3S1 IS STARTING
```

??? WLM has started the first server region instance.

```
+BBOU0021I INITIALIZATION COMPLETE FOR CB SERIES SERVER APSRV3S1
```

??? Server region is successfully up, but registration continues.

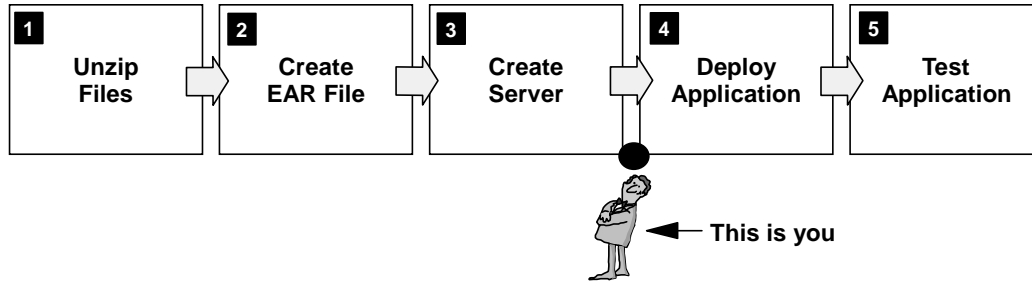
```
+BBOU0698I REGISTERING SERVER APSRV3
```

??? Another phase of registering the server has started.

```
+BBOU0695I NAMING REGISTRATION COMPLETED FOR SERVER APSRV3
```

??? Congratulations. The new server is started.

**Review**




??? The server and server instance has been created, but no applications have been deployed to the server. That's the next phase.

## Phase 4: Deploy Application

??? With the application server created, you are now ready to deploy the EAR file. This involves the use of the SMS EUI tool.

### Overview



- Create conversation
- Delete PolicyIVP from WASASR2 server
- Validate, commit and activate
- Create *another* conversation
- Install PolicyIVP into APSRV3
- Validate, commit and activate

### Deploy Your Version of PolicyIVP Application

#### Add New Conversation

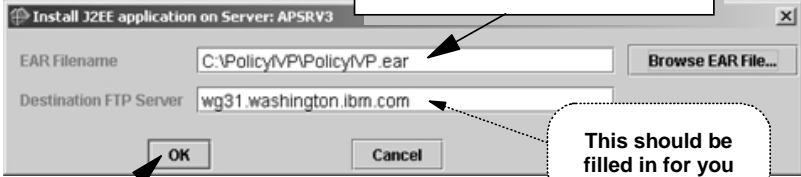
- You should be familiar with this process by now. If you want a reminder, see "Start SMS EUI and Add Conversation" on page 12.

#### Import the Application

- Expand the tree and locate the APSRV3 tab under the "J2EE Servers" folder. Right-click on APSRV3 and select "Install J2EE application..."
- In the popup that results, supply the location and name of the EAR file you generated back in Phase 3 of this lab:

1


**Supply the directory and name of the EAR file you created. Use the "Browse" button if you wish.**



**This should be filled in for you**

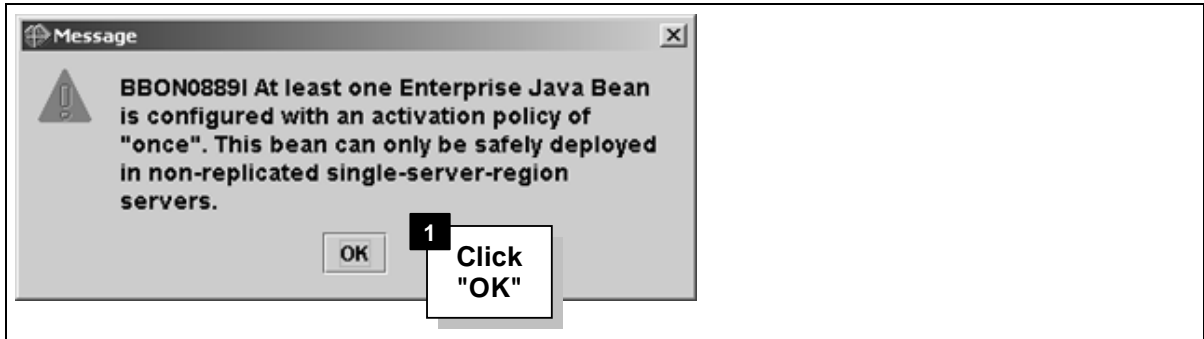
2

**Click on "OK" and wait while EAR is loaded**



## WP100277 - PolicyIVP Construction Lab

- You'll receive the following message:



??? This pops up because one of the XML deployment descriptors in the EAR file defines the activation policy for this EJB as "once." The file `websphere390xdd.xml` has a tag of:

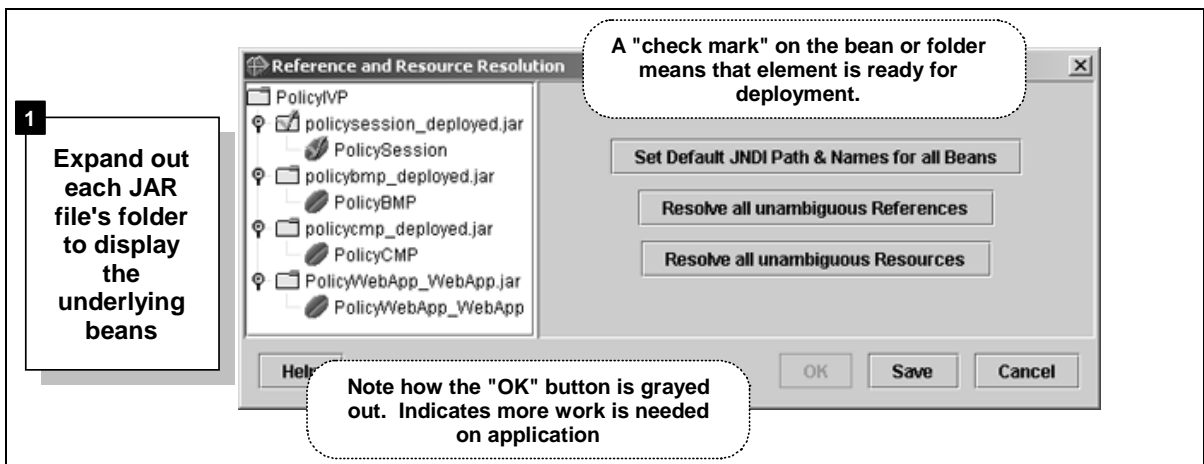
```
<activation>once</activation>
```

That means you shouldn't have copies of this running in different regions of the same server. Different servers is okay, but recall that WAS 4 allows multiple *server regions* of a given server to be started by WLM based on the workload its sees. You may very well have defined your WLM application environment with "support a single server region only" (which would prevent multiple regions from starting), but the SMEUI tool doesn't know that. So it issues this message.

This message also applies to multiple instances of the same server (also known as "replicated servers). So for applications defined as `<activation>once</activation>`, you should not deploy into servers defined with multiple instances.

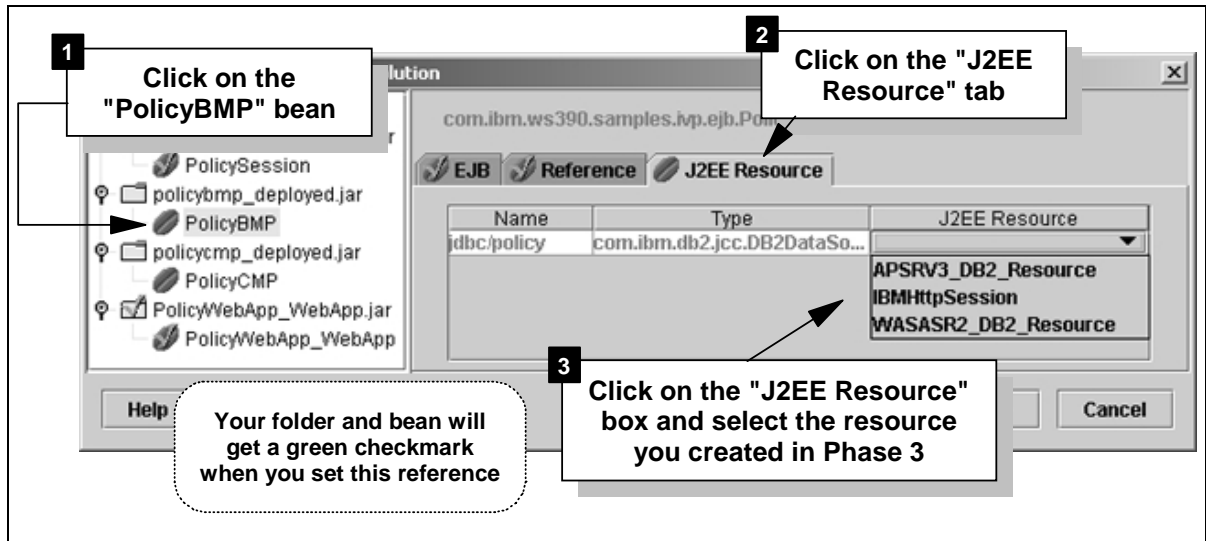
For the purposes of this PolicyIVP application, don't worry about this message. You won't be driving enough workload against this to make a difference. Just be aware of why this message pops up.

- You should now have a "Reference and Resource Resolution" panel. Toggle the switches next to each folder to expose the "beans":

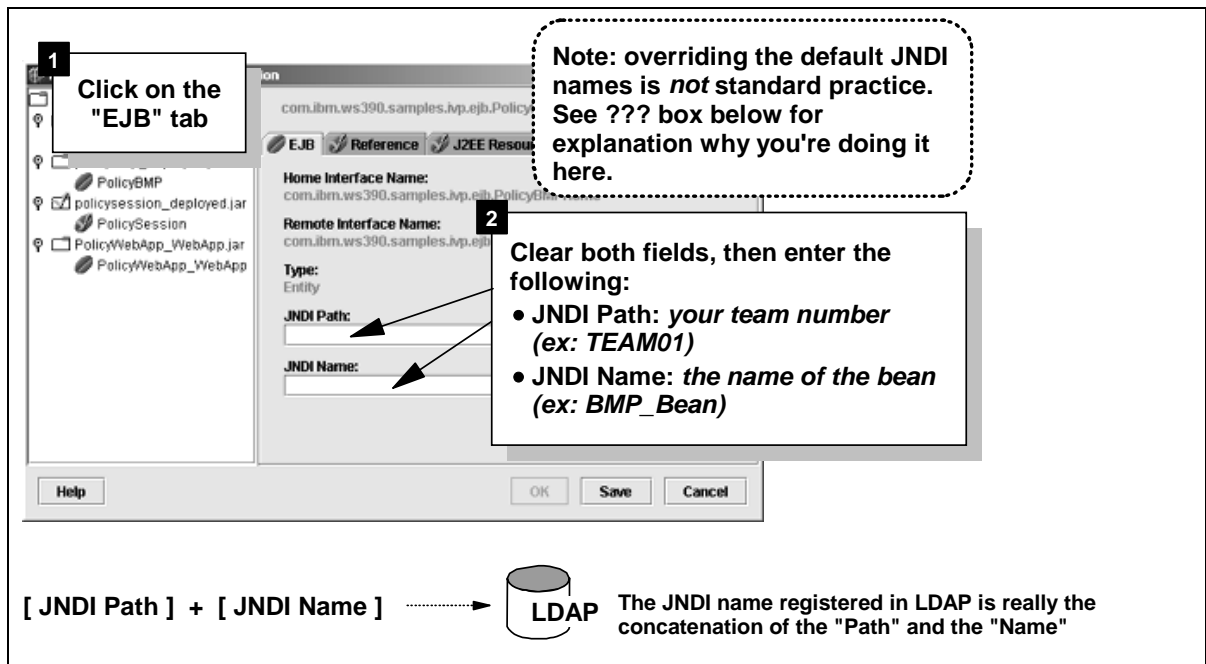


**BMP Bean: resolve resource reference and set JNDI name**

- Tie the BMP bean's symbolic data resource reference ("jdbc/policy") to the J2EE data "Resource" you created earlier:



- Now set the JNDI name for the BMP bean:



??? Normally you would click on the "Set Default JNDI Name and Path" button (but don't do that here). The SMS EUI tool would then fill in the values with a unique name based on the Sysplex and server name along with the class file name of the bean's home interface. For this lab we wanted to illustrate that the JNDI name can be anything you want<sup>note</sup>, as long as it's unique in the JNDI namespace. *Key lesson: JNDI name and path is not the HFS location of the class file!*

**Note:** "... JNDI name can be anything you want" is true so long as the application uses symbolic lookups. If the application uses hard-coded JNDI lookups, then you *must* set the JNDI names of the beans *exactly equal* to whatever is hard-coded in the application. PolicyIVP uses "java:comp" symbolic lookups, so you can apply any unique JNDI name you wish.

## WP100277 - PolicyIVP Construction Lab

- Verify that the BMP bean has a green check mark on it's folder and bean in the "tree" structure, as well as a green check on each of the tabs. That means the BMP bean is ready to go.

### CMP Bean: resolve resource reference and set JNDI name

- Click on the CMP bean and resolve the "J2EE Resource" for that bean.
- Set the JNDI name for the CMP bean using the same type of naming convention you used for the BMP bean (TEAM01 and CMP\_Bean).
- Verify that all the elements for the CMP bean have green check marks.

### Session Bean: set JNDI name

??? The session bean does not do any direct access of a data resource, so you do not need to set the "J2EE Resource" for this bean. You do need to provide a JNDI name for it, however.

- Set the JNDI name for the session bean using the same type of naming convention you used for the BMP and CMP beans (TEAM01 and Session\_Bean)

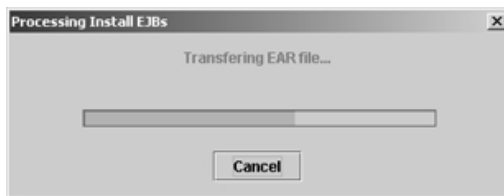
### WebApp: set JNDI name

- Set the JNDI name for the webapp using the same naming convention (TEAM01 and WebApp).

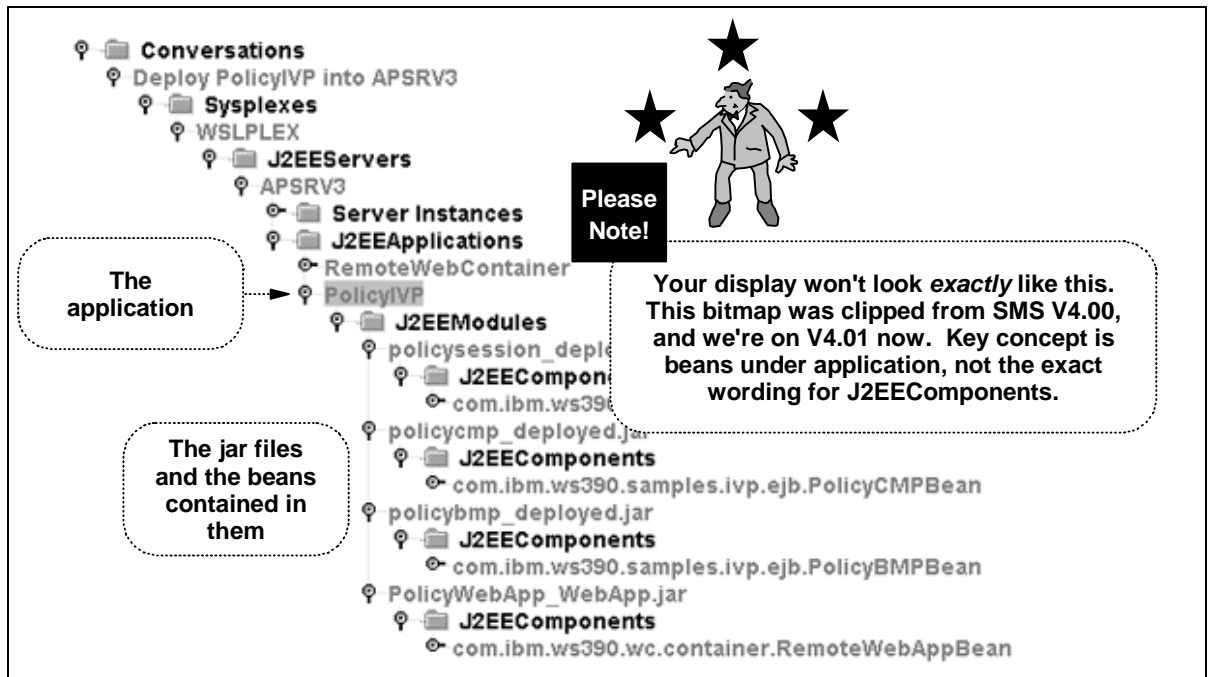
### Transfer the EAR File to the WAS Server

??? At this point all of your beans should have the "check mark" on them, indicating you're ready to go.

- Click on the "OK" button. You'll see a progress panel that looks like this:



And if you expand the tree, you'll see the beans on the server:

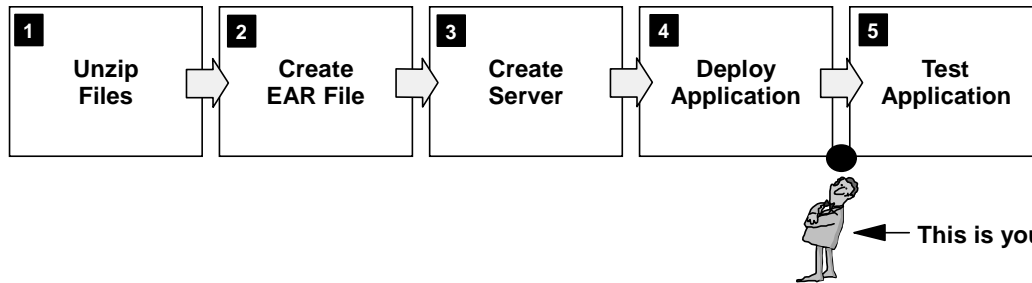


**Validate, Commit and Activate the Conversation**

- ☐ Same process as before. (See "Validate, Commit and Activate the Conversation" on page 16 for a reminder of how to do this).

??? It'll take a minute or more to activate. The process involves shutting down the APSRV3 server and restarting it, in the process going through the effort of registering all the new stuff you added to the server. Be patient.

**Review**



??? The next step is to validate the deployment by running a client against the EJBs.



## Phase 5: Test Application

??? If all was done properly, the deployed application should now respond to contact by a client. The client you will use is the "fat client" you used earlier in this class to validate the "PolicyIVP" application.

### Modify "ejbivp.sh" shell script

??? WebSphere comes with a shell script that is used to drive a Java program that will act as the client to the session bean. That shell script -- `ejbivp.sh` -- is located in the:

```
/usr/lpp/WebSphere401/samples/PolicyIVP/ejb
```

directory. You may either edit the existing copy (back it up first, please), or copy it to another directory before modifying it.

**Note:** the `/usr/lpp/WebSphere401` portion of that may be different on your system, depending on where WebSphere is installed on your system. The default is `/usr/lpp/WebSphere`.

- Edit `ejbivp.sh` and make sure the `LIBPATH` and `CLASSPATH` entries all point to the right "install root" directory. By default they'll all start with `/usr/lpp/WebSphere`. If that's where WebSphere is installed on your system, then you're all set. If WebSphere is installed somewhere else, then you'll need to do a global change to point to the right directories.

- Add the following two lines to the file immediately after the `export CLASSPATH` line:

```
export RESOLVE_IPNAME=[the IP name of your WebSphere runtime]
export RESOLVE_PORT=[the port on which the WebSphere SM server is listening]
```

??? This provides the Java client knowledge of how to get to the SM Server. You can leave these two out and it'll *probably* work, provided that the WebSphere runtime is on the same TCP stack as your OMVS environment, and the SMS is listening on the default port 900.

- Now update the two `java ...` lines at the bottom of the file and provide the JNDI name of the session bean. If you followed the directions, it should have a name of `/TEAM01/Session_Bean`:

```

:
# Run the testcase for 'bmp'
java -DSESSION_NAME= com.ibm...TestClient bmp
# Run the testcase for 'cmp'
java -DSESSION_NAME= com.ibm...TestClient cmp
    
```

This portion is the Java class file for the "fat client"

1 Overwrite what's there initially with `/TEAM01/Session_Bean`

This is a parameter to request either "bmp" or "cmp" persistence

#### ??? Two Important Notes:

1. The use of Java "fat clients" is not typical in the "real world." This business of editing shell scripts and typing out JNDI names is a lab exercise and not something you'll do outside of testing like this. Normally a servlet will be the client to the session bean. See "Reference: Configuring the WebApp to Serve as the Client" on page 31 for information on how to do that.
2. In the "real world" you'll also take the "default JNDI name." We had you override it with a made-up name here just to illustrate the point that JNDI names are just strings of characters, and *not* HFS directory pointers.

**Run the shell script**

- ❑ Go into OMVS (or via a Telnet terminal), change to the directory in which the `ejbivp.sh` file resides, and issue the command:

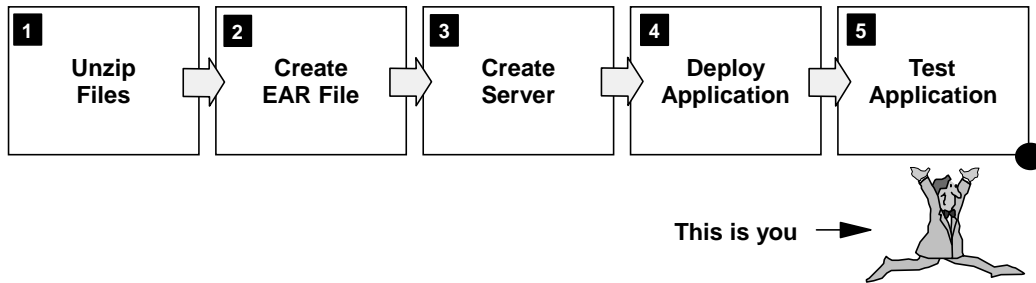
```
./ejbivp.sh
```

- ❑ Look for the following as an indication of success:

```
***** bmp bean will be run!
Look up policy session home
Obtaining polycysession bean jndi name...
The polycysession bean jndi name is: (JNDI name you gave)
Lookup policy session home
homeName: (JNDI name you gave)
Narrow policy session home
Driving policy session bean
bmp IVP has completed successfully Good sign!

***** cmp bean will be run!
Look up policy session home
Obtaining polycysession bean jndi name...
The polycysession bean jndi name is: (JNDI name you gave)
Lookup policy session home
homeName: (JNDI name you gave)
Narrow policy session home
Driving policy session bean
cmp IVP has completed successfully Good sign!
```

**Review**



??? Congratulations!

**STOP HERE! End of Lab**  
(What follows is reference material, not lab)

## Reference: Configuring the WebApp to Serve as the Client

This exercise had you package up and deploy a web application in the form of the WAR file called `PolicyWebApp.war`. That web application is now installed on the server. It's ready to act as the client to the session bean as soon as you do the steps necessary to enable the `webcontainer.conf` file, bind the application to the virtual host.

The steps necessary to do all of that are outlined in the white paper [WP100238](#), which is located on the [www.ibm.com/techdocs/support](http://www.ibm.com/techdocs/support) website, just as this white paper was.

Some things you'll need to know to get that servlet working:

- The "context root" is `/PolicyIVP`
- The initial HTML page will be `/PolicyIVP/cebit.html`

## Reference: Where the JAR Files Came From

### Overview

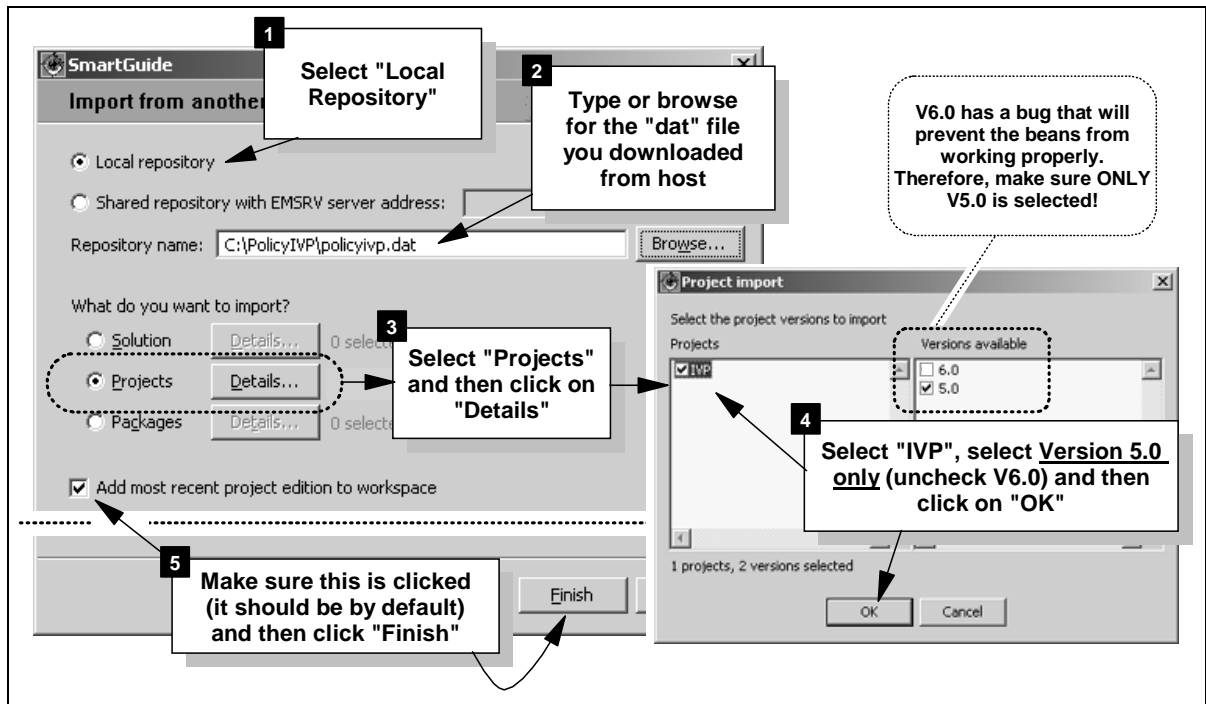
WebSphere for zOS and OS/390 ships with a VisualAge for Java "repository" file (called a "dat" file) which contains the PolicyIVP application. If you have VAJ 3.5.3 on your workstation, you may "import" the DAT file and then export the individual EJBs as separate JAR files.

### Import Repository

VisualAge for Java maintains its work in a single large database-like file called a "repository." That repository has a file extension of "dat." One feature of VAJ is the ability to *export* a project out of VAJ into another "dat" file, which can then be *imported* into another copy of VAJ. This allows projects to be sent to other developers. That's what you're doing here: importing a "dat" file into *your* copy of VAJ.

- Create a directory on your PC called `C:\PolicyIVP`
- FTP in *binary mode* the VisualAge for Java repository "dat" file:
 

```
From: /usr/lpp/WebSphere401/samples/PolicyIVP/ejb/policyivp.dat
To: C:\PolicyIVP\policyivp.dat
```
- Start VisualAge for Java. You may need to select "Go to the Workbench" on a pop-up panel to complete the initialization of VAJ.
- When VAJ is up, select *File* ⇒ *Import* ⇒ *Repository* ⇒ *Next* to bring up the "Import from another Repository" panel.
- Now you need to tell it what "dat" file to import, and a little about the what from inside the repository you wish to select. Do the following:



It'll take a few moments to bring everything into VAJ. Be patient.

- ❑ After the "dat" file has been successfully imported into VAJ, you no longer need the file on your C: drive. *Delete* the file C:\PolicyIVP\PolicyIVP.dat from your workstation.

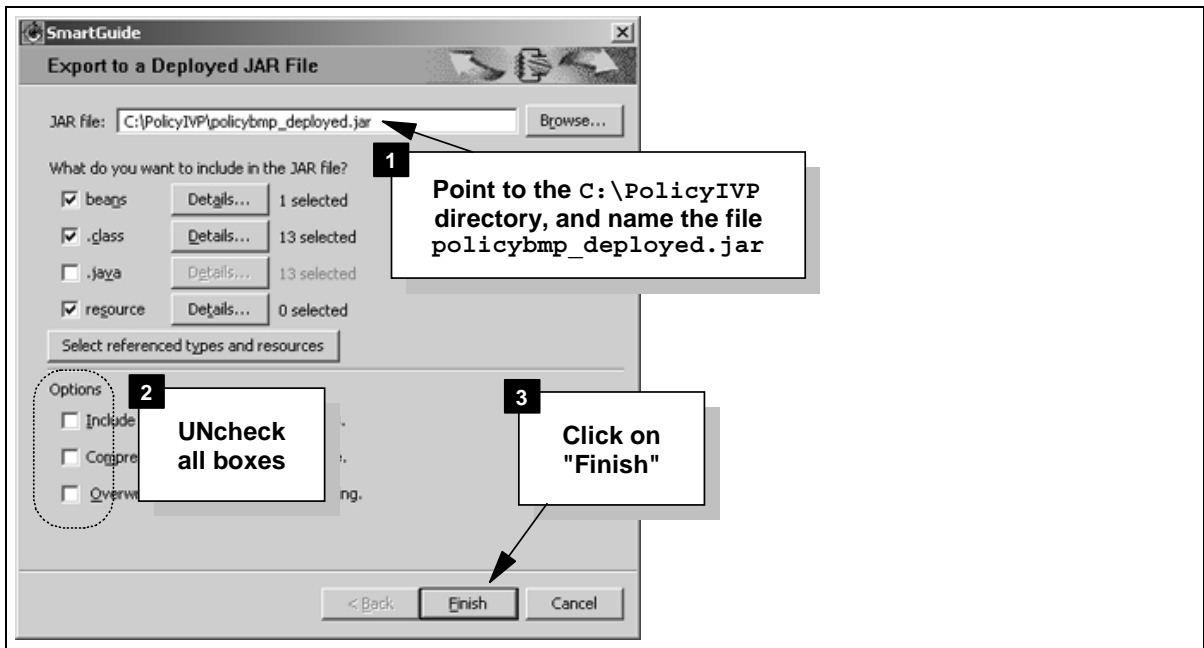
**Export JAR Files**

??? The various EJBs, clients and utilities are in VAJ, but need to be exported to be used on the server. We'll export to JAR files, then use the Application Assembly Tool (AAT) to construct the EAR file used during deployment.

- ❑ Select the "Policy" EJB group under the EJB tab, click the *right* mouse button, then select "Generate Deployed Code." Be patient while it re-generates the code.

??? This results in the stubs, ties and persistor code being re-generated. If the copy of VAJ you're using is newer than the one on which the developer used, these class files will be refreshed with the newer code from *your* copy of VAJ.

- ❑ Select the **PolicyBMP** bean, *right-click* and then *Export* ⇒ *Deployed JAR*. This results in the following screen being displayed:



??? The JAR file you're putting out to your C: drive contains not only the Java class files produced by the developer, but also the stubs, ties and persistor code generated by VAJ.

- Repeat the same process, this time for the **PolicyCMP** bean, exporting it to:  
C:\PolicyIVP\polycymp\_deployed.jar
- Repeat the process once more, this time for the **PolicySession** bean, exporting it to:  
C:\PolicyIVP\polycysession\_deployed.jar
- Now go back to the "Projects" tab of VAJ, locate the "IVP" project and select the following package:

com.ibm.ws390.samples.ivp.utilities

Then right-click the package, select *Export* ⇒ *Jar File* ⇒ *Next*. Then point to the C:\PolicyIVP directory and name the output file PolicyUtil.jar.

??? You should now have four files in the C:\PolicyIVP directory: polycysession\_deployed.jar, policybmp\_deployed.jar, polycymp\_deployed.jar and PolicyUtil.jar. What's left is the PolicyWebApp.war file, which is discussed next.

## Reference: Where the WAR File Came From

??? The WAR file is packed in the PolicyIVP.ear file that ships with WAS 4.01. To get at the file requires only that you unzip the WAR file from the EAR and make a small modification to it.

### Download PolicyIVP.ear file

- ❑ Establish an FTP session to the system on which WAS is installed, and download *in binary format* the following file:

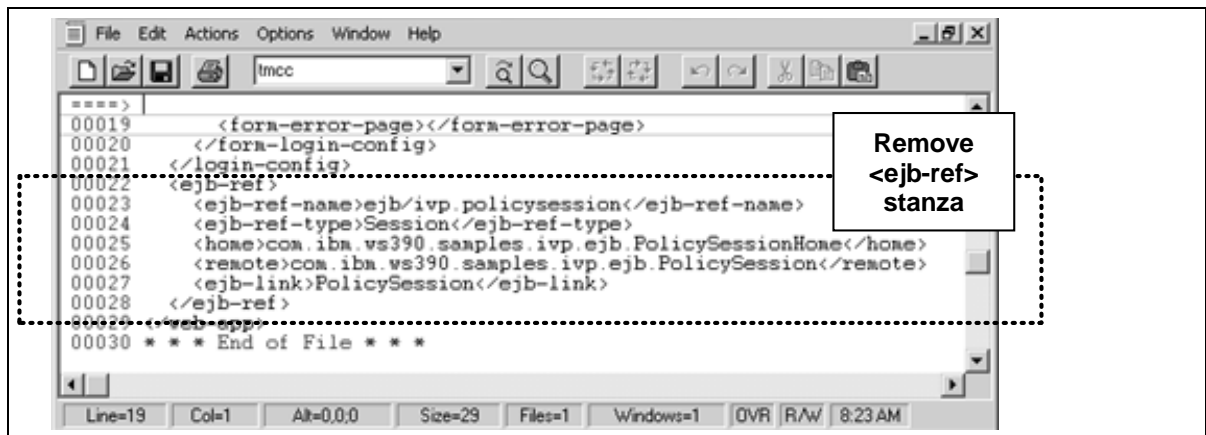
```
/usr/lpp/WebSphere401/samples/PolicyIVP/ejb/PolicyIVP.ear
```

### Extract PolicyWebApp.war file

- ❑ Using any unzip tool (WinZIP®, PKZIP®), extract from the EAR file the file called PolicyIVP.war.
- ❑ Rename the WAR file to PolicyWebApp.war

??? There's no technical reason why this is required. It is done here simply so the WAR file can have a name other than "PolicyIVP," which is the same as the EAR file. PolicyWebApp.war is a more descriptive name better indicating the contents of the file.

- ❑ (**Optional**) Edit the web.xml file which is inside the PolicyWebApp.war file and remove the <ejb-ref> stanza:



??? The EAR file shipped with WAS has already been run through the AAT tool, and the <ejb-ref> stanza in the web.xml file was placed there by AAT to provide the Webapp's reference to the Session bean. *You may leave this <ejb-ref> stanza in place.* Leaving it in place will simply mean that the WebApp's EJB reference will already be there when you run the AAT tool. For this lab, we removed the <ejb-ref> stanza.

## Change History of Document

- May 21, 2002 -- Original Document

End of Document