Lab:

PolicyIVP Construction

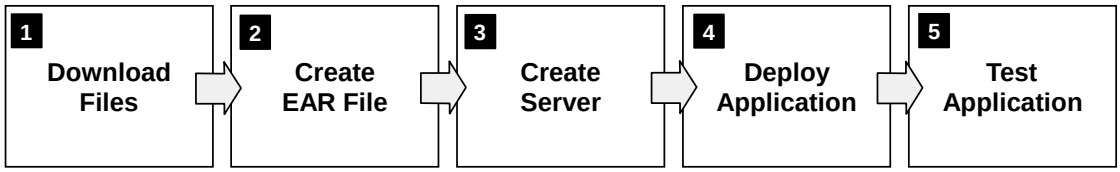
(AAT Version 023)

(This page intentionally left blank)

Lab: Deploying Applications that Access DB2

Overview of Lab

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

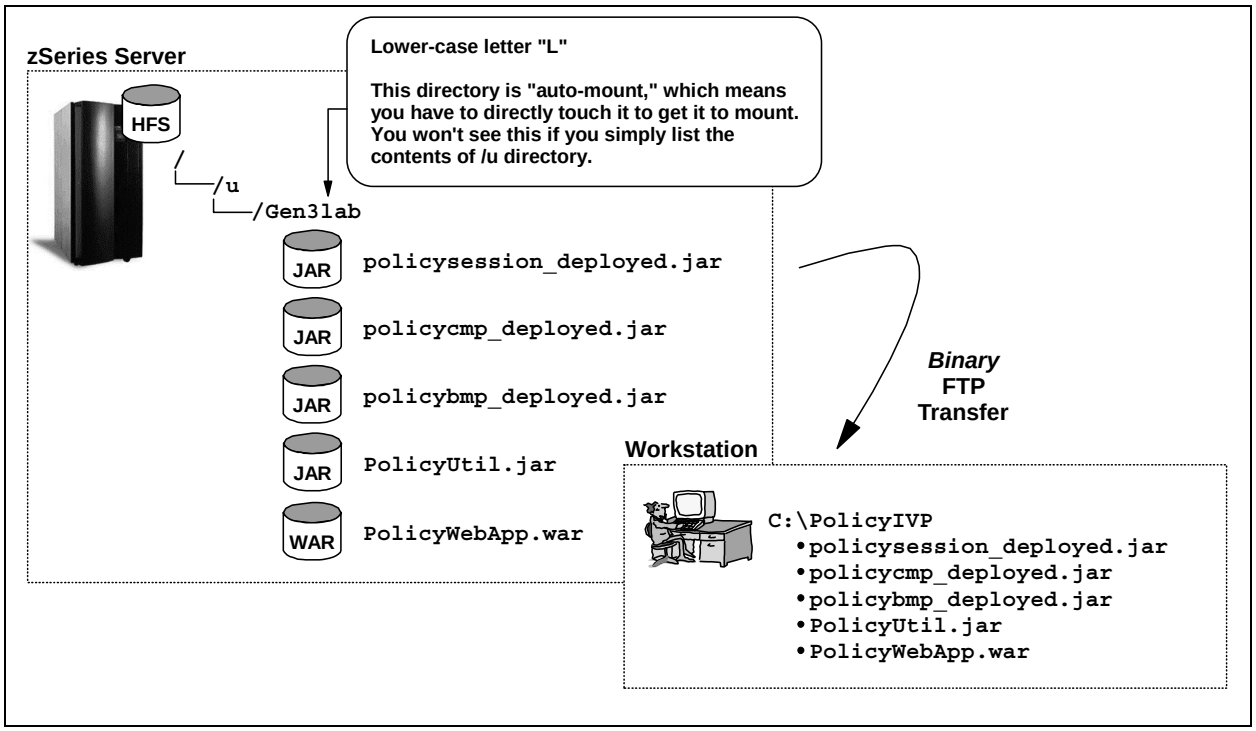


??? Boxes with the three question marks ("???") provide an explanation for what's going on. They do not contain "to do" activities, but do offer some background on why you're doing what you're doing.

Phase 1: Download JAR Files and WAR File from Server

??? The purpose of this phase is to get the JAR files we have placed on the zOS server for you to use. The JAR files contain the EJBs for the PolicyIVP application, and the WAR file contains the webapp. 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 29 and "Reference: Where the WAR File Came From" on page 31.

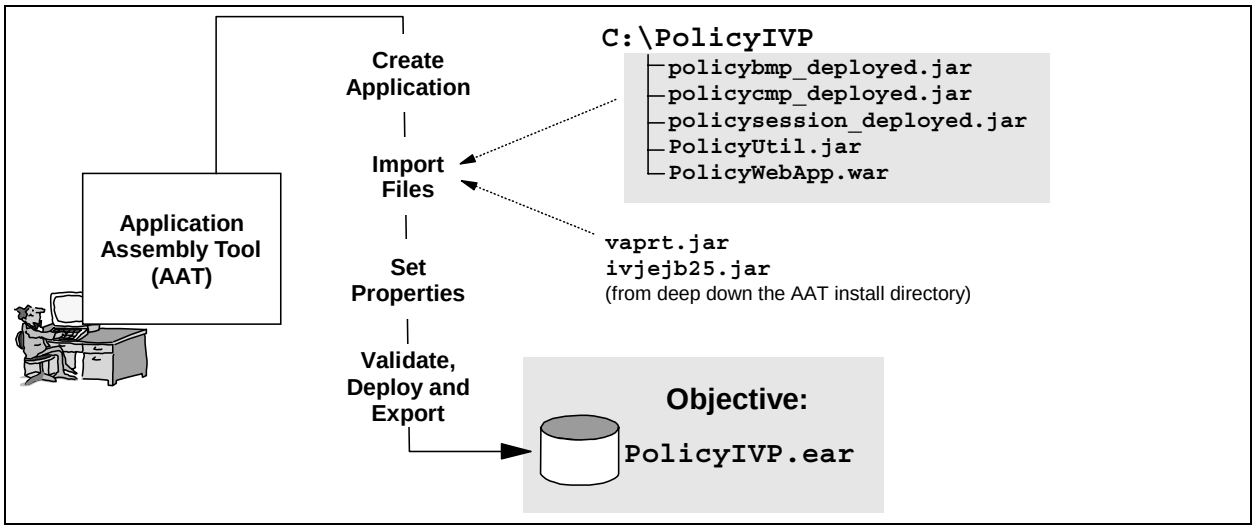
Create PC Directory and FTP files

- ☐ Create a directory on your PC called `C:\PolicyIVP`
- ☐ FTP in *binary mode* from the zSeries server to your PC all the files shown in the picture above (an FTP "get" operation).

Phase 2: Generate EAR using AAT

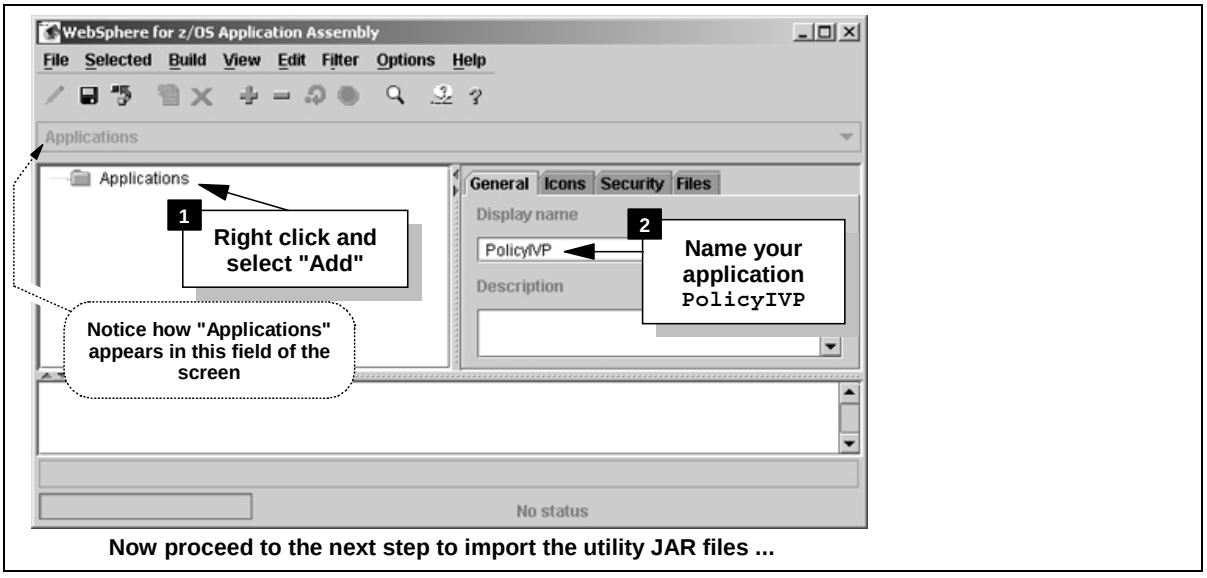
???	EAR files are another form of zipped archive file, and it is in EAR files that J2EE applications are packaged for deployment into the WAS 4.01 runtime. The tool used to generate the EAR file is the "Application Assembly Tool" (AAT).

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":

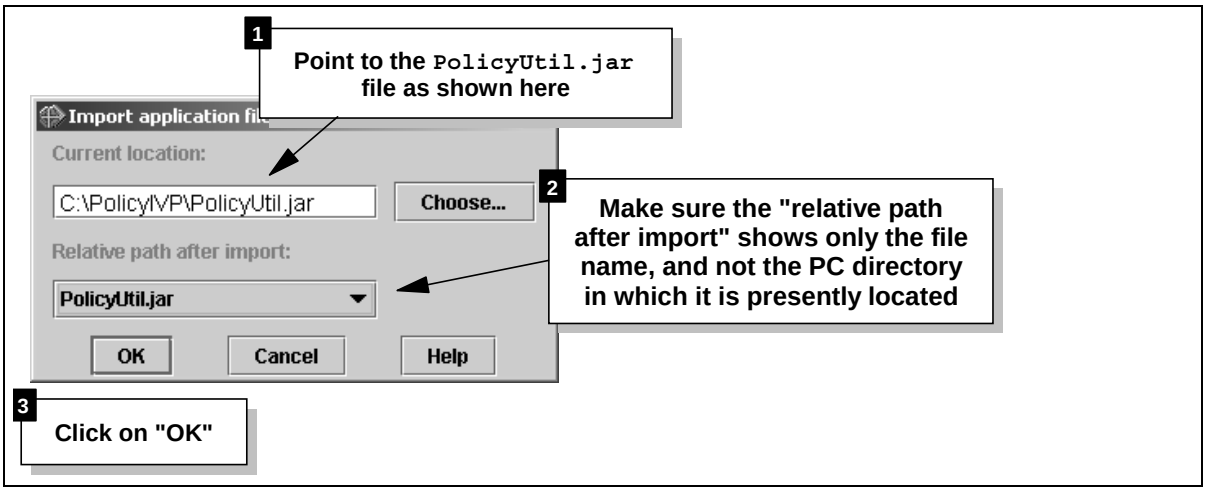


Now proceed to the next step to import the utility JAR files ...

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:



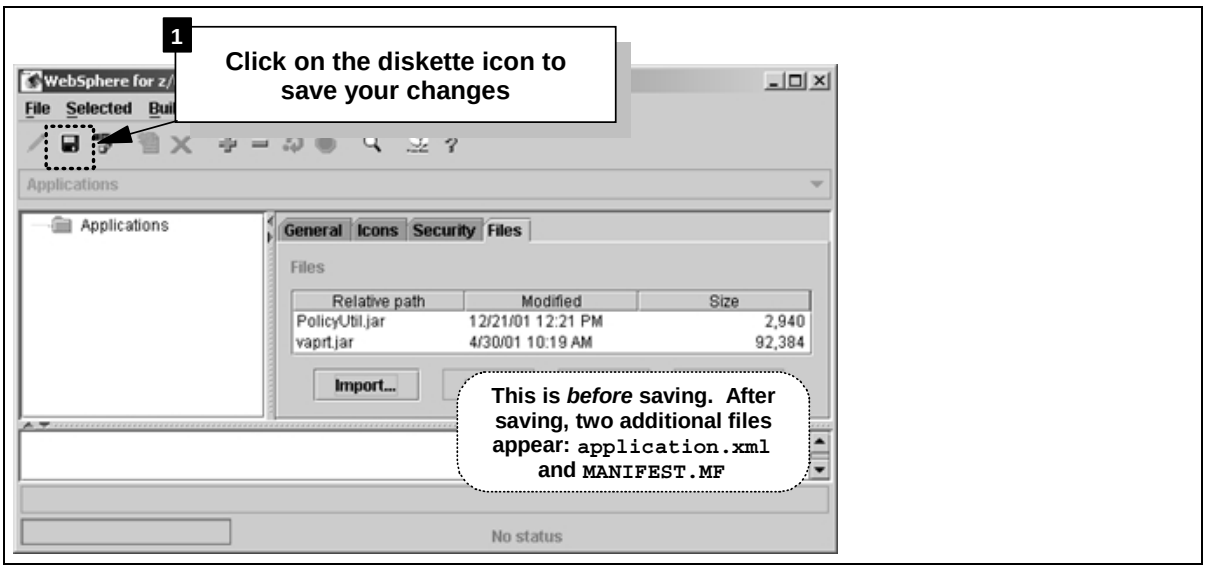
- ☐ Click on the "Import" button again, and this time bring in the following file:

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

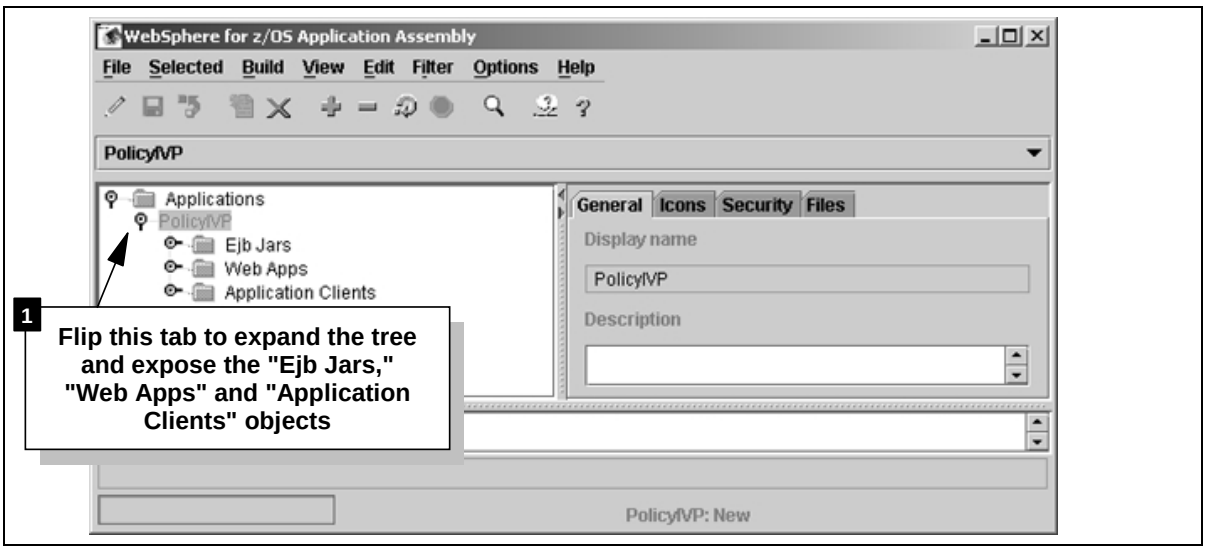
Hint: You may find it easier to click on the "Choose" button and then navigate the tree structure to get that file, rather than typing out the whole name.

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:

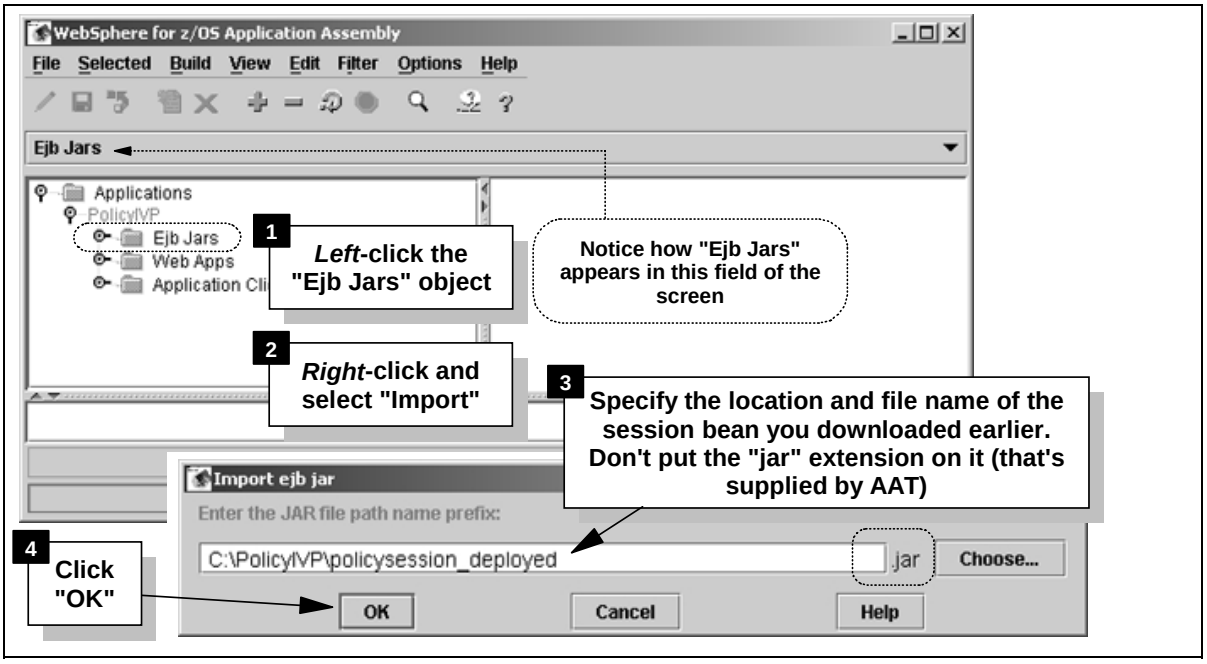


- You should see the PolicyIVP application now appear below "Applications". Expand the tree below "PolicyIVP" by clicking on the tab next to the name:



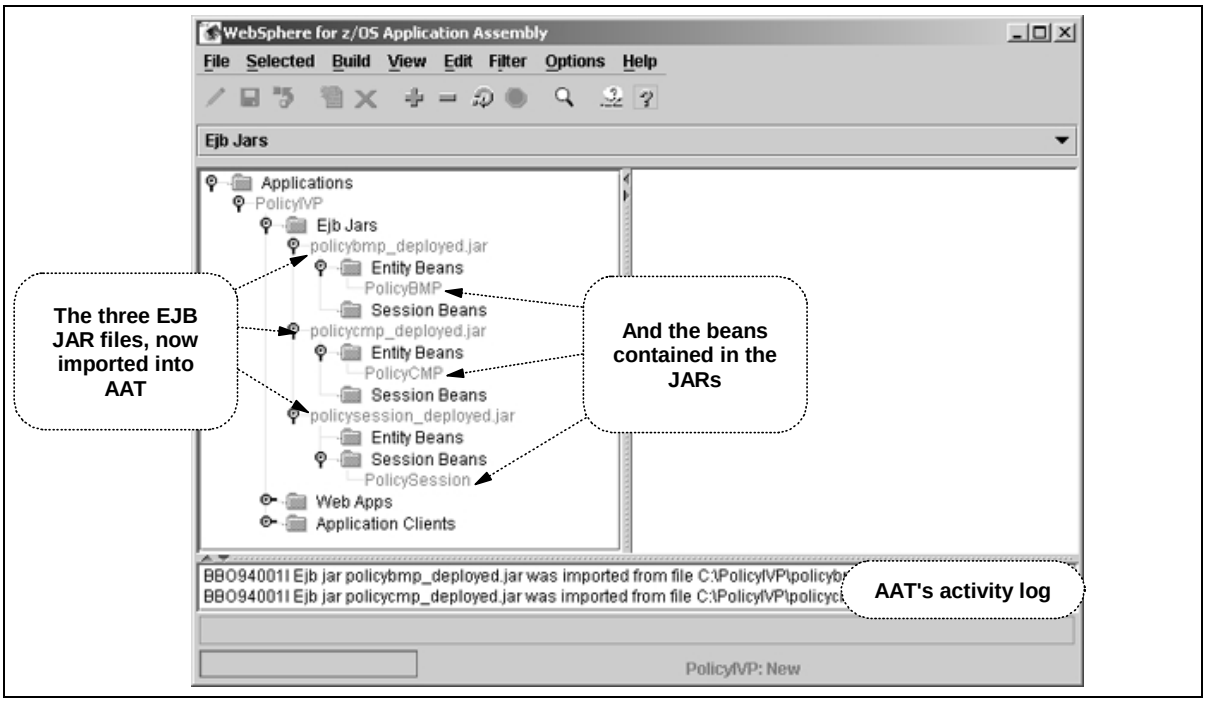
Import the EJB JAR Files

- 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 (`policycmp_deployed.jar`)
  - The BMP entity bean (`policybmp_deployed.jar`)

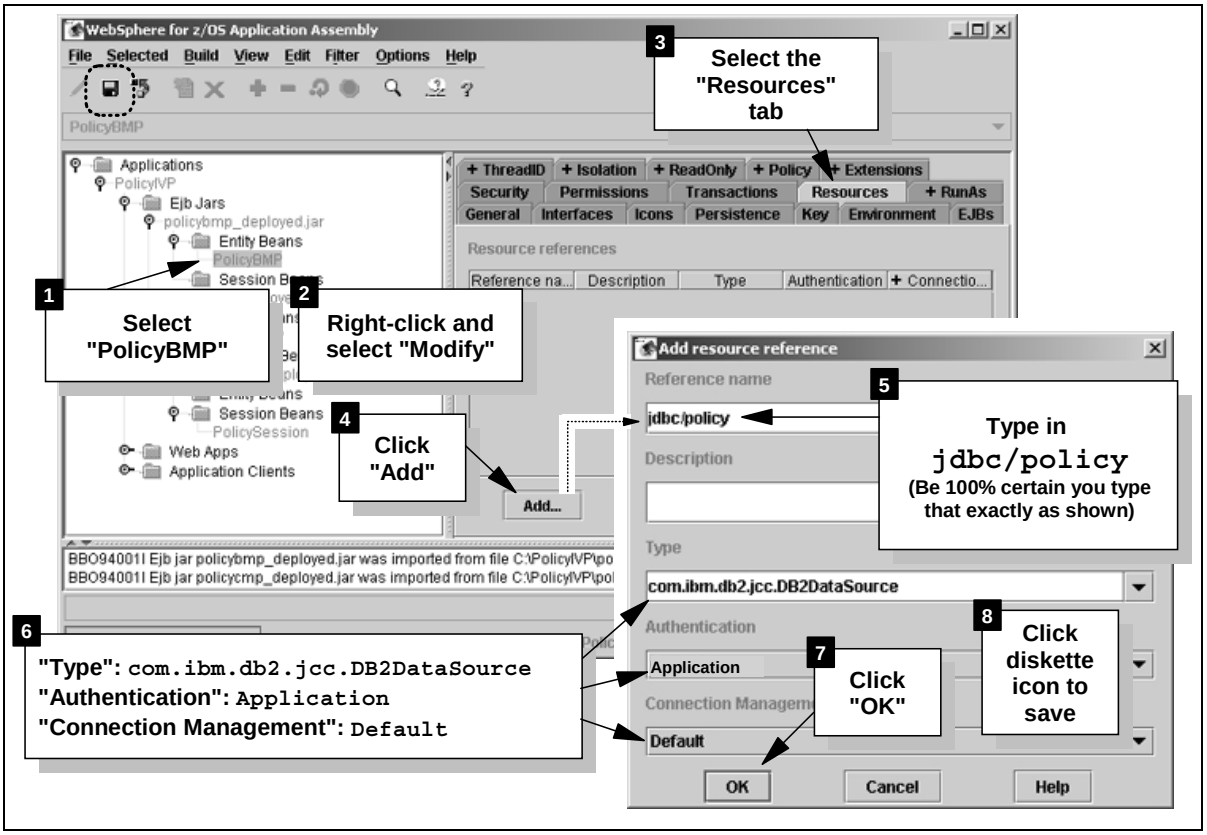
  When you're done, the AAT panel should look like this after you expand out all the tabs on the panel:



Modify the Properties of the EJBs

BMP Entity Bean

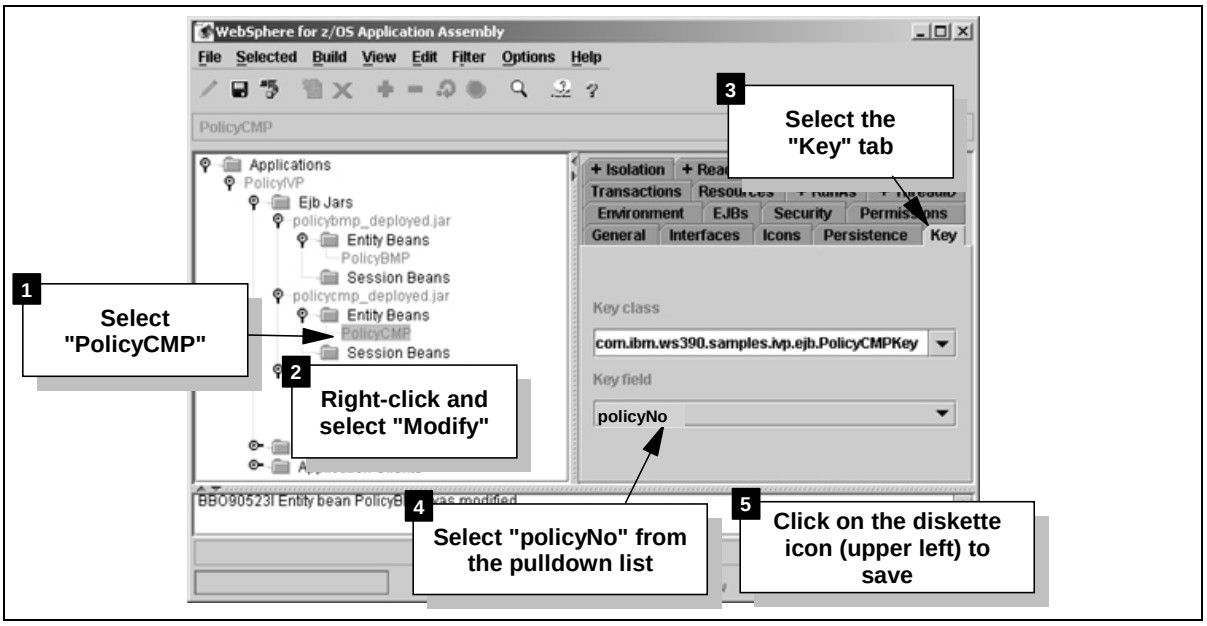
☐ Do the following:



??? 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

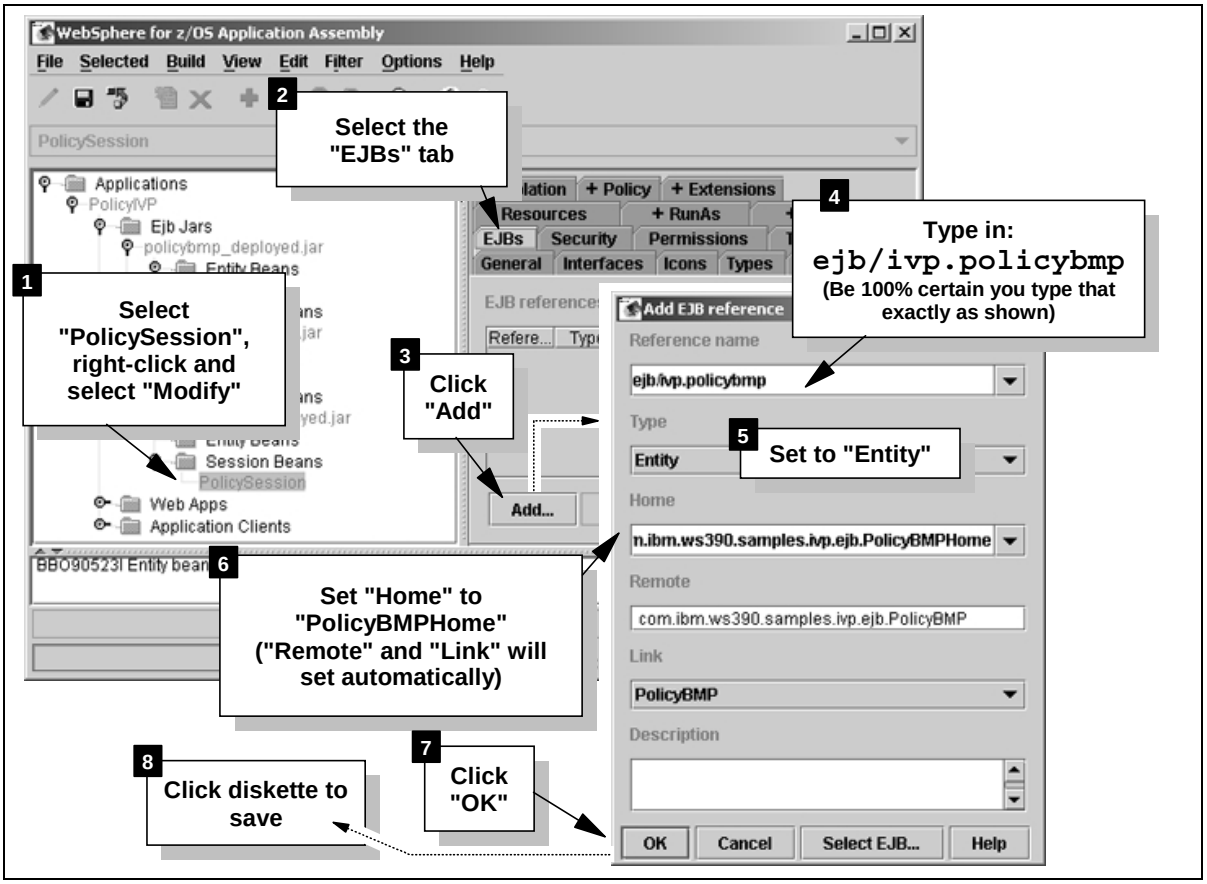
☐ Do the following:



???	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:



???	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.

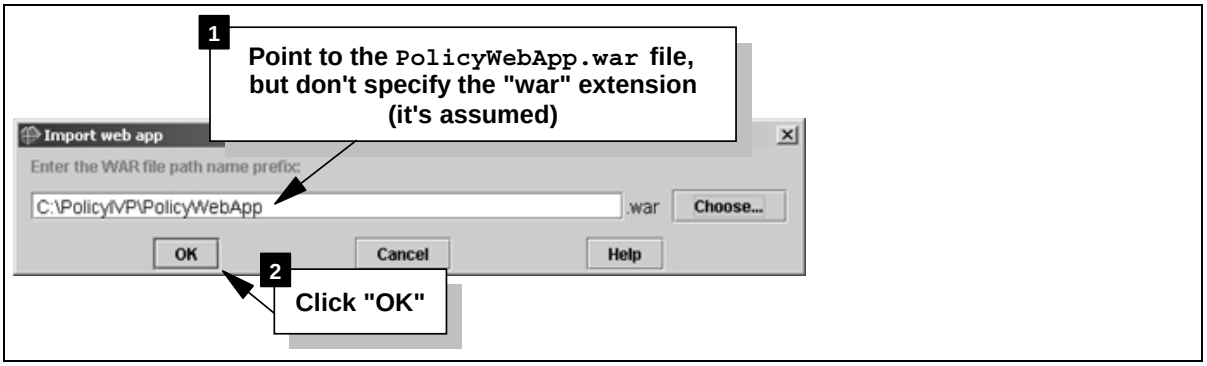
☐ ***Repeat*** the *previous eight steps*, only this time provide information for the CMP bean. Be certain to provide the "Reference Name" as **`ejb/ivp.policycmp`**. The naming convention is nearly identical, except for "CMP" vs. "BMP".

Validation Activity

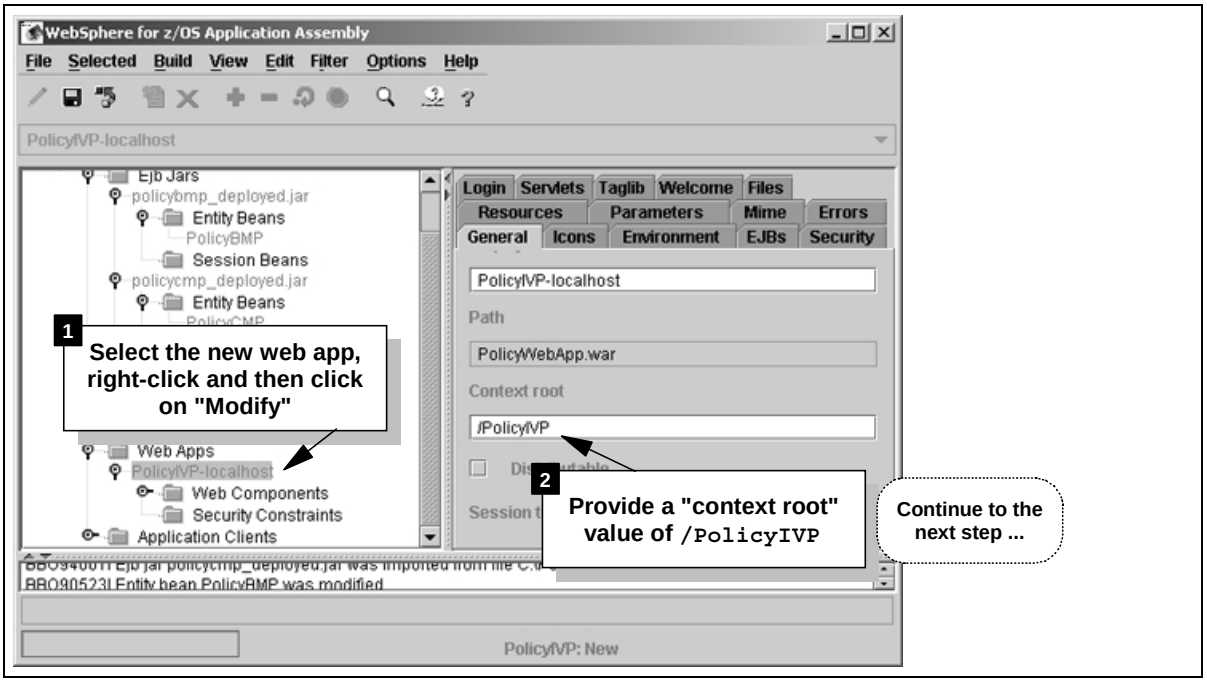
☐ Did you create an EJB reference from the Session bean to *both* the BMP bean *and* the CMP bean?

Import the Web Application and Modify its Properties

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

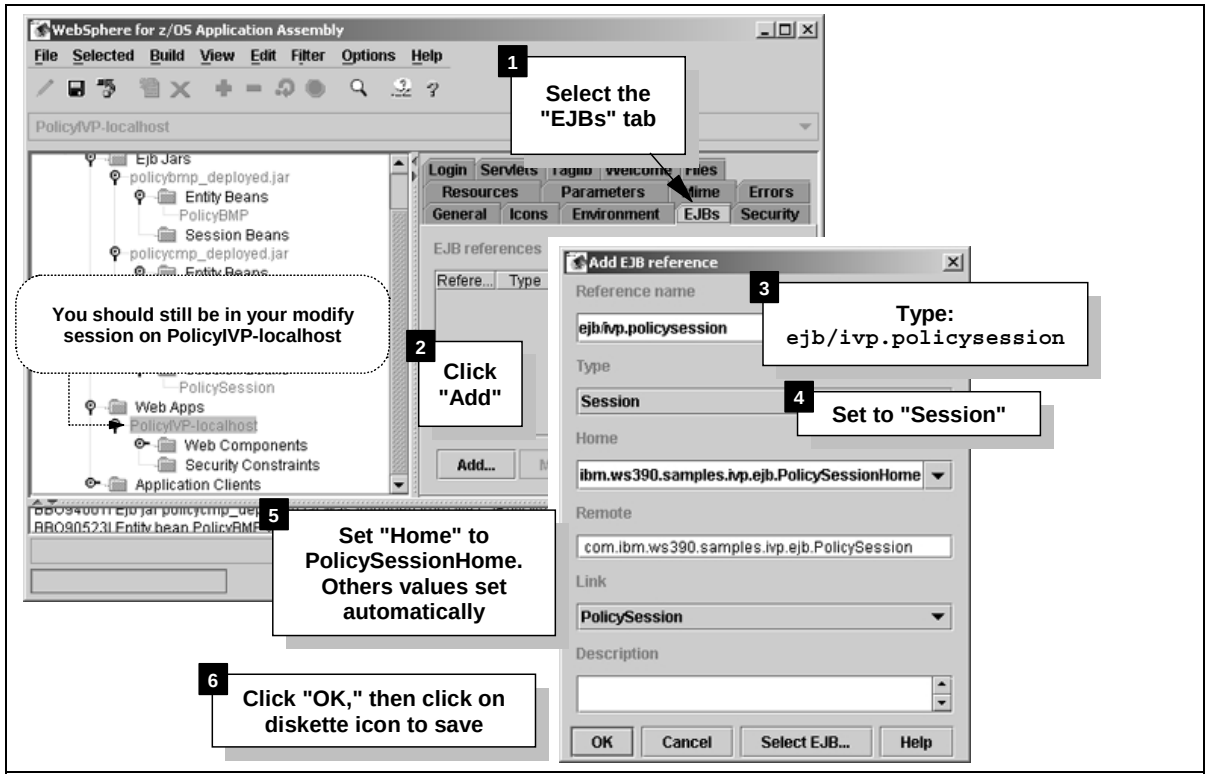


☐ Once the WAR file is imported, do the following:



??? 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:



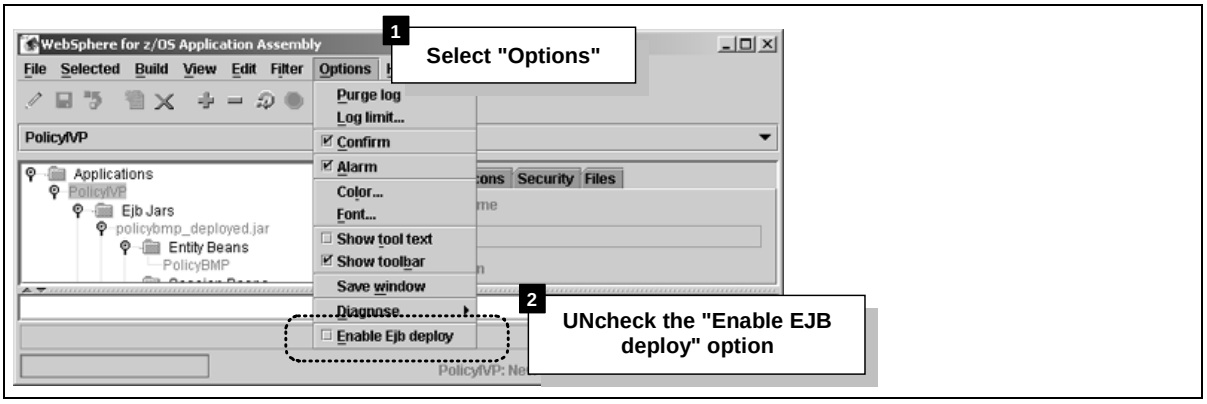
??? The servlet will be the client in the "client/server" relationship, and it needs to know about the session bean to which it'll attach. This process provided the reference name, and then linked that name to the Home and Remote interfaces of the PolicySession bean. This process is identical in purpose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

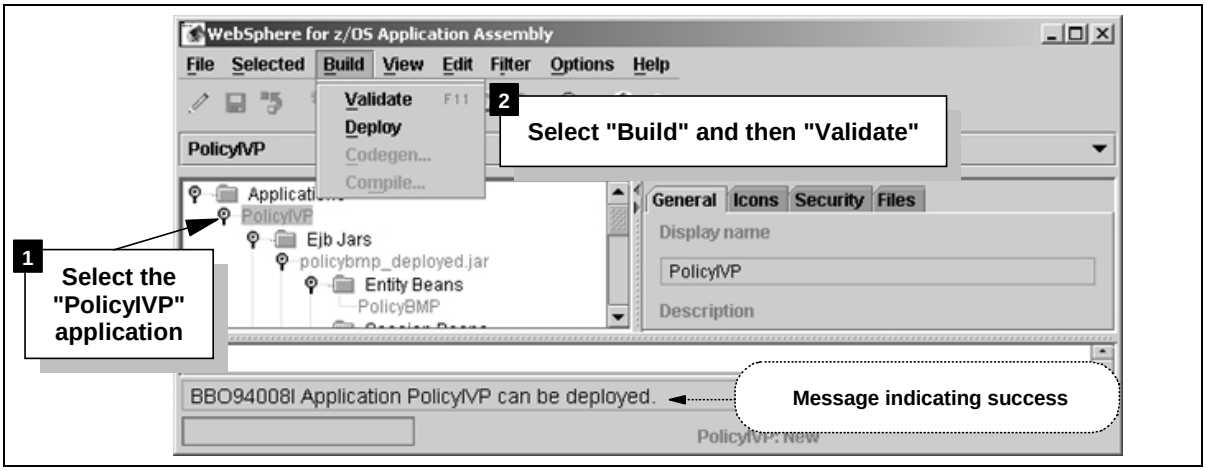
Validate

☐ Disable the "EJB Deploy" option:



??? You need to instruct AAT to *not* generate the stubs, ties and persistor 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.

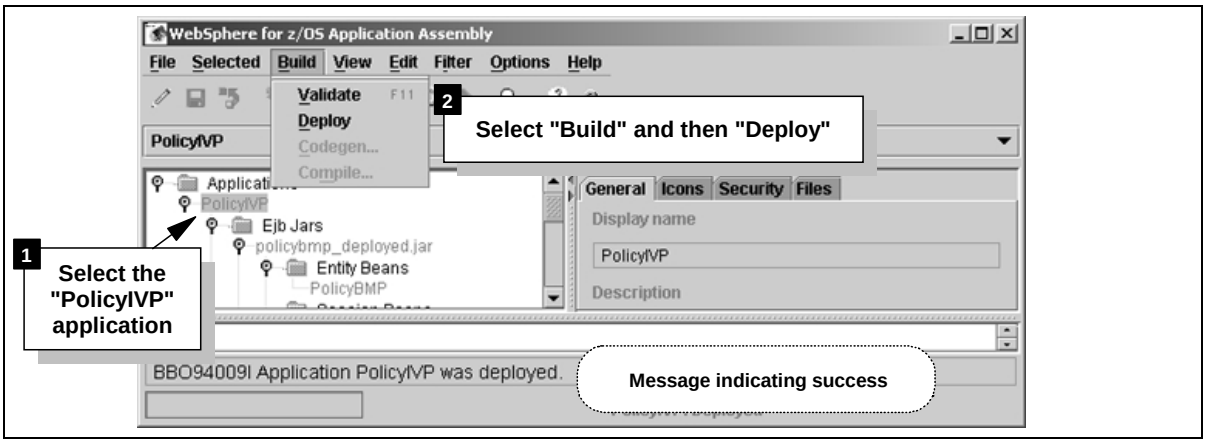
☐ Now do the following:



??? 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

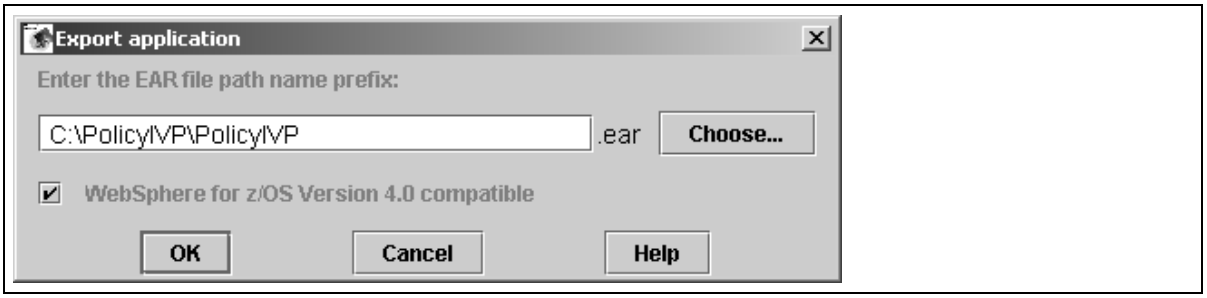
- ☐ Do the following:



???	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.

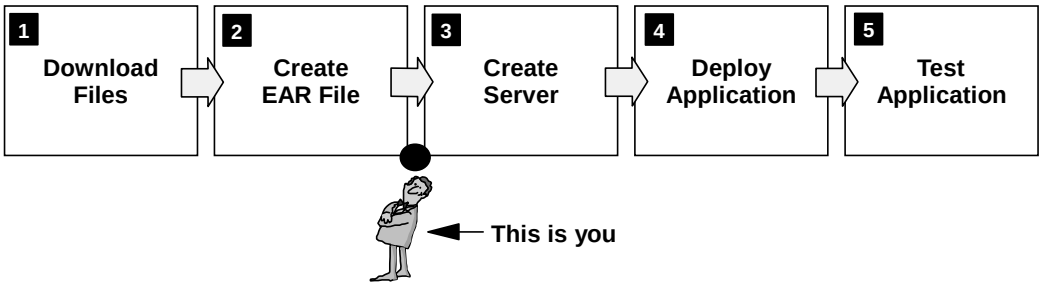
Export

- ☐ 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.
- ☐ Use WinZIP to review the contents of the EAR file.

Review

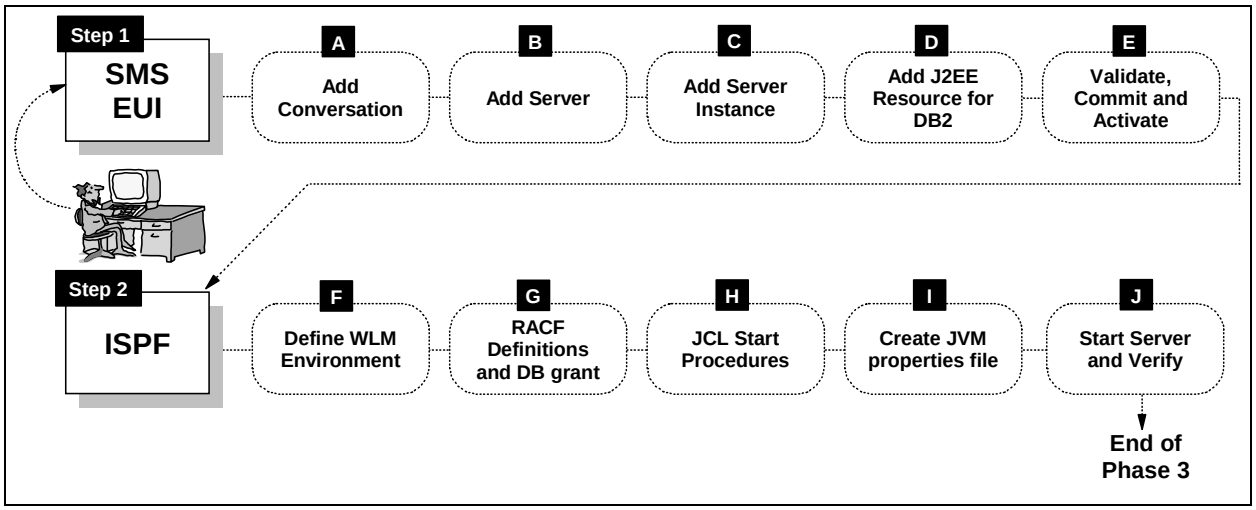


???	You are done with the AAT. With an EAR file on your C: drive, you're now ready to use the WAS 4.01 Systems Management interface tool to install the new application into the J2EE server environment.

Phase 3: Create Application Server

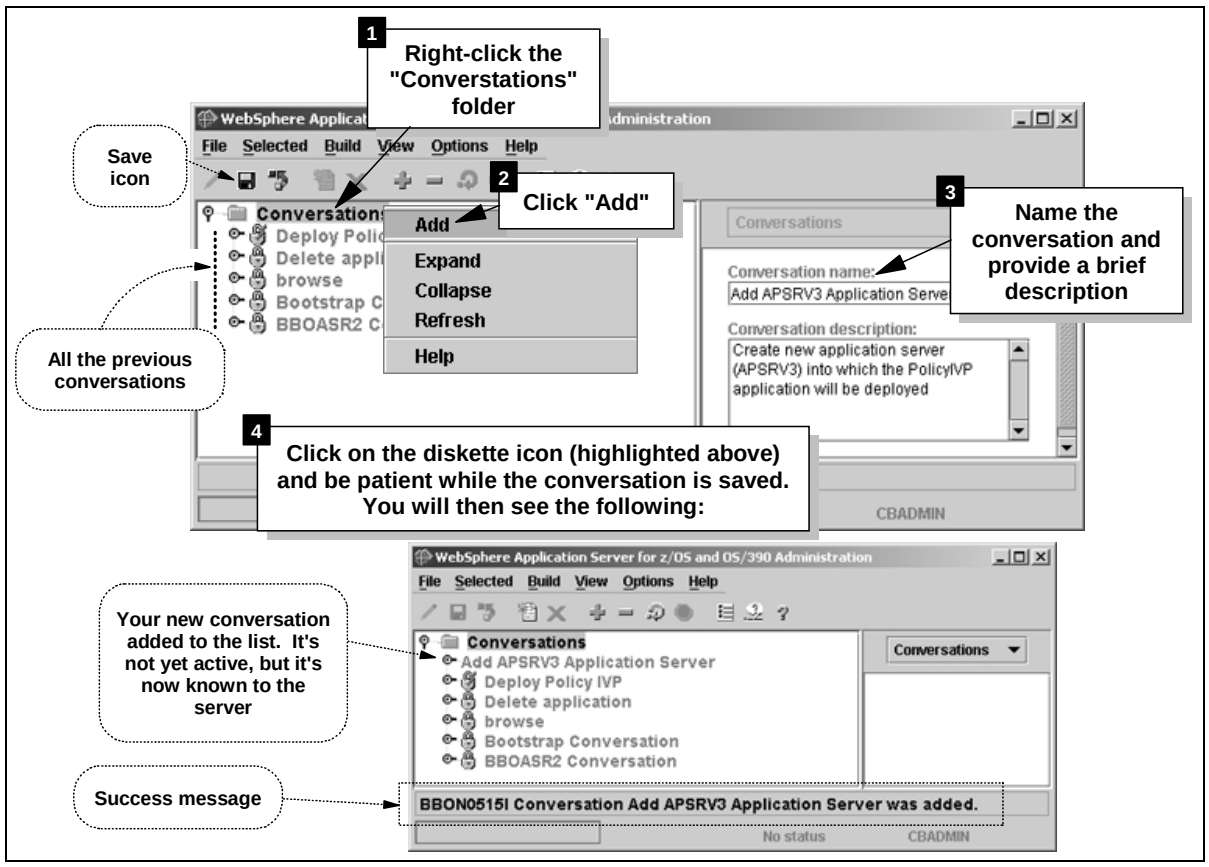
???	In an earlier lab you deployed the already-assembled "PolicyIVP" application into the "WASASR2" application server, which is one we created for you. In this lab you will create a *new* application server -- APSRV3 -- and in Phase 4 deploy the EAR file you created into that server.

Overview



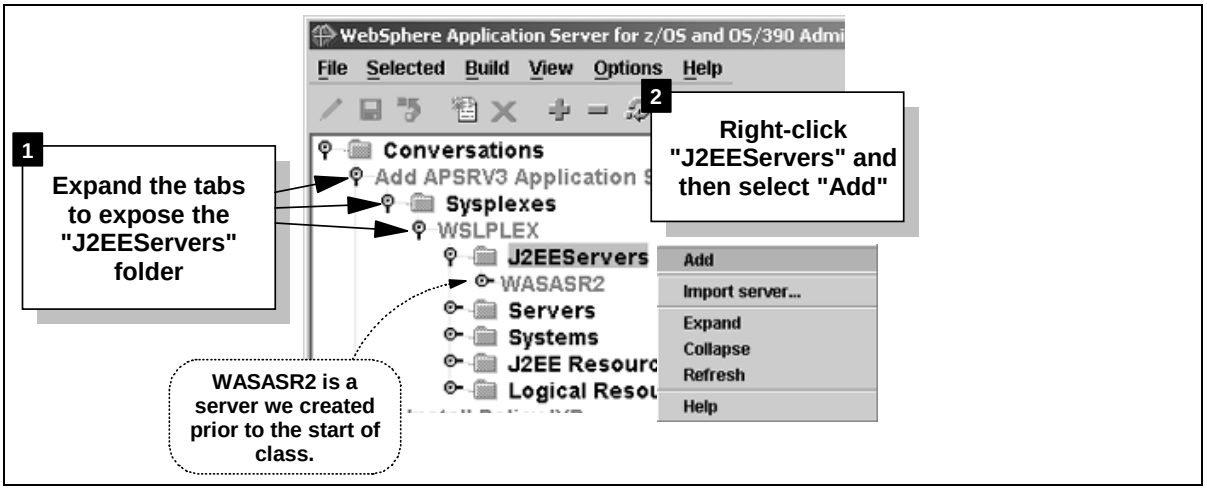
Start SMS EUI and Add Conversation

- ☐ Start the SMS EUI program and connect to the server using the `WASADM1` userid with password `WASADM1`.
- ☐ Do the following:



Add Server

- ☐ Start this process by doing the following:

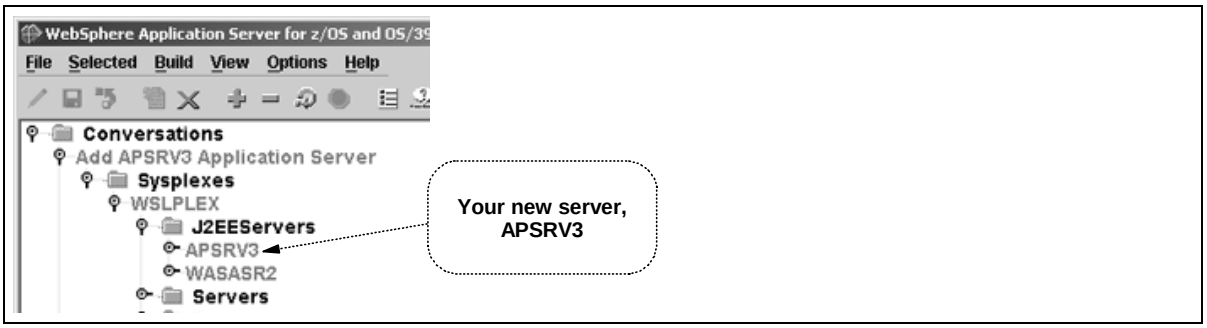


- ☐ The right-hand side of the SMS EUI panel will now be a scroll-pane with many input fields. Fill in each field with the information provided here:

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)*
SSL Use Confidentiality Only	*(accept default*
SSL RACF Keyring	*(accept default)*
SSL V2 Timeout	*(accept default)*
SSL V3 Timeout	*(accept default)*

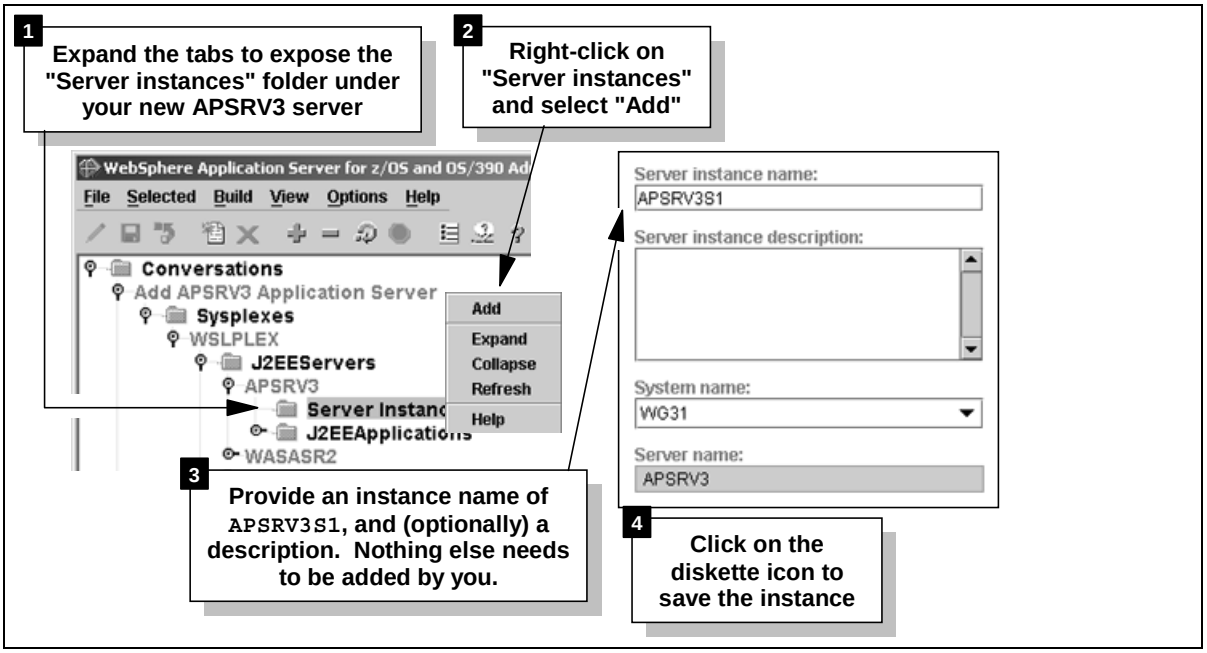
Field	Value
Security Preference List:	
Password Order	1
Passticket Order	2
(All other entries in Preference List):	*(leave blank)*
Write Server Activity SMF Records	*(accept default)*
Write Container Activity SMF Records	*(accept default)*
Write Server Interval SMF Records	*(accept default)*
Write Container Interval SMF Records	*(accept default)*
Interval Length	*(accept default)*
Environment Variable List	*(leave blank ... values from sysplex level set during bootstrap used for this server)*

☐ 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

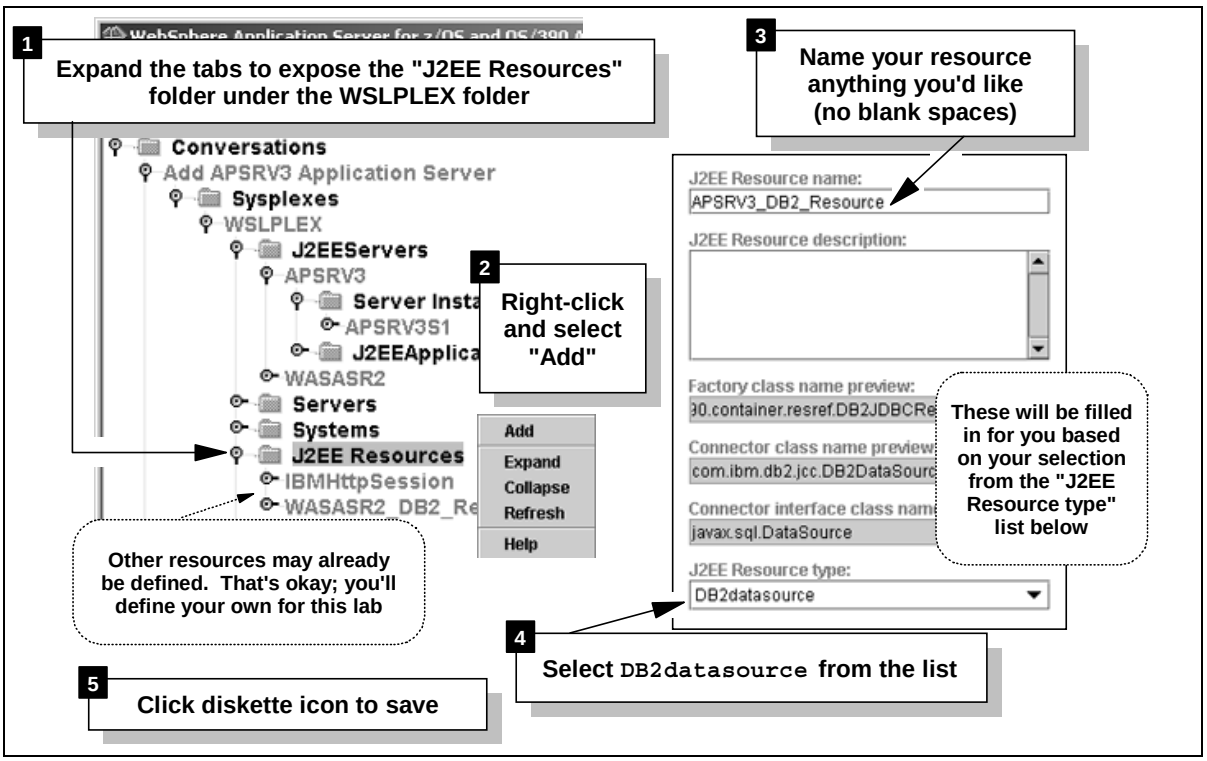
☐ Do the following:



??? 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.

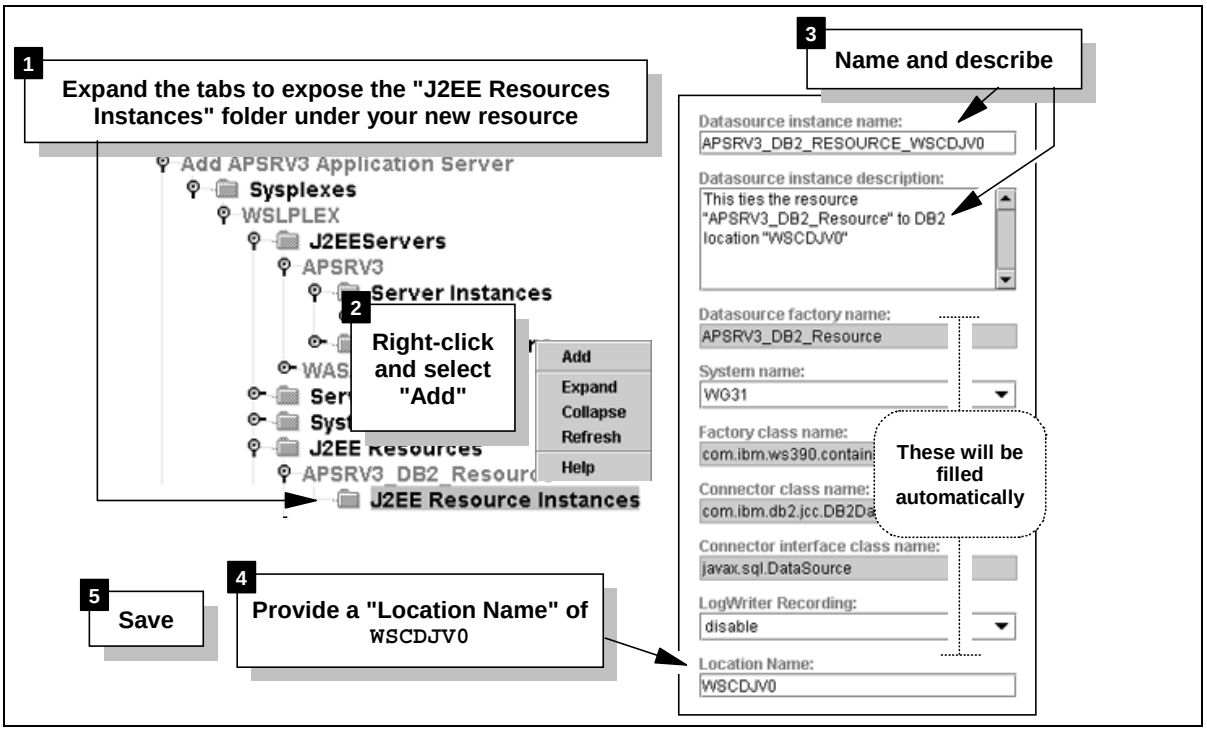
Add J2EE Resource and Resource Instance for DB2

☐ Add the "resource" first:



> ***???*** There will already by a DB2 resource we created prior to class. We're having you do it again just to show you how this is done.

☐ Now add the "resource *instance*":



> ***???*** `WSCDJV0` is the "group attach name" for the DB2 subsystem `DJV0` on your system.

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:

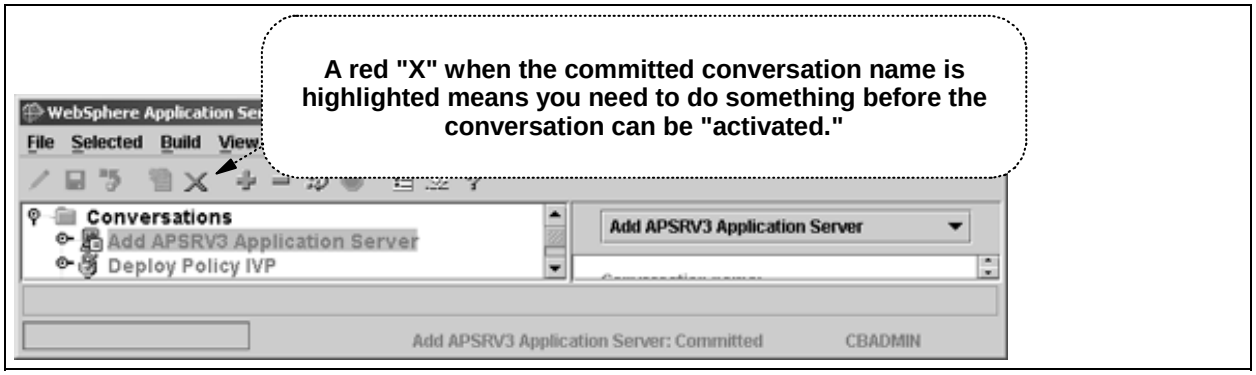
```
BBON0442I Conversation Add APSRV3 Application Server is valid
```

- ☐ 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:



- ☐ 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:

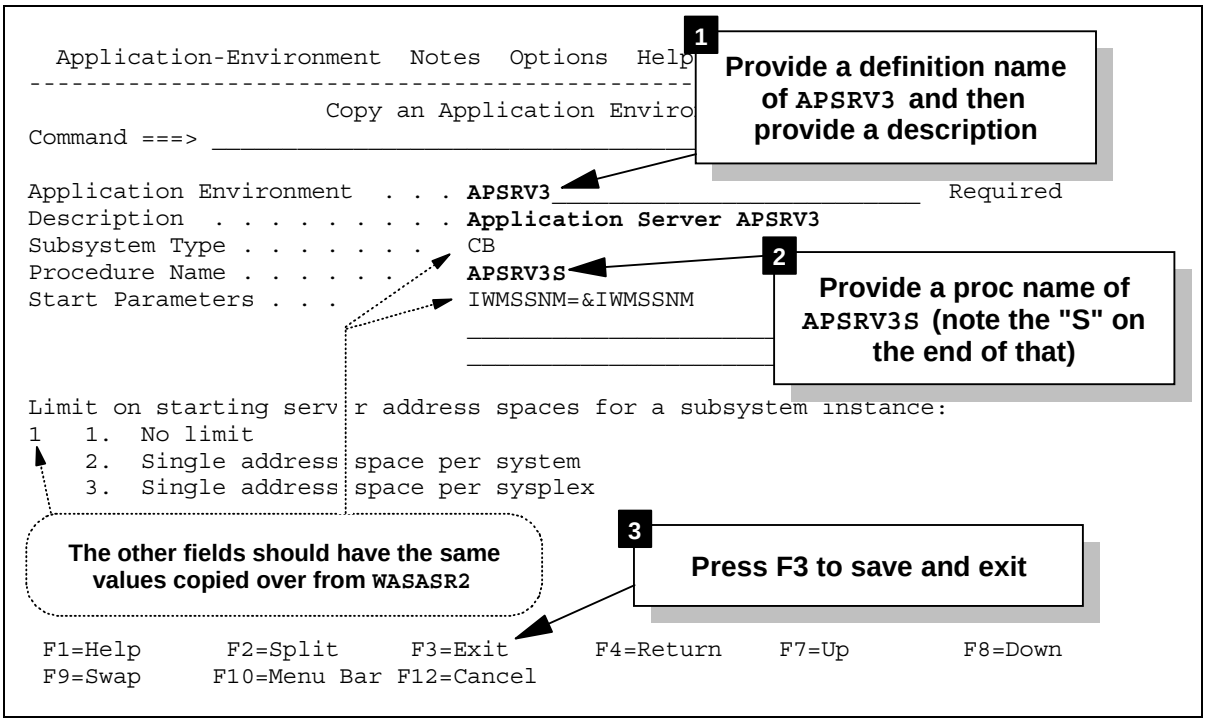
Perform non-GUI Tasks

???	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 RACFdefinitions and the JCL start procedures.

Define WLM Application Environment

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

- ☐ Log onto TSO using userid `USER1`.
- ☐ 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 the definition for `WASASR2` and hit "Enter"
- ☐ Now create your new environment:



- ☐ 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 **`WSCDEF`** and 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

??? Each server you create has its own set of RACF definitions. Rather than have you execute each command by hand, we're supplying you with a CLIST. Feel free to download the CLIST and take it home if you'd like (we'll supply diskettes).

- ☐ Browse the member **$$GEN3.WAS.DATA(APPLRAC)** and see the RACF commands that will be executed. This job is tailored for a server name of APSRV3.
- ☐ Now submit the job **$$GEN3.WAS.CNTL(APPLRAC)**, which calls the RACF CLIST from the previous step. 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.

- ☐ Copy the member TEICHMN.WAS401.CNTL(**BBOIGRT**) over the FB 80 PDS you made earlier in this class: USER1.WAS.CNTL.

  ??? Why TEICHMN.WAS401.CNTL? That was the data set Bob Teichman named when he ran the ISPF customization dialogs to set up the systems for this class. The ISPF customization dialogs generated all the JCL jobs in that data set and updated many of the values in the members with the information Bob supplied on the ISPF panels.

- ☐ 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):

```
****** ***************************** 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 BBBOIGSV 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.

> ***???*** If you wish, you may log onto the SYSADM1 userid and use SPUFI to execute those grants.

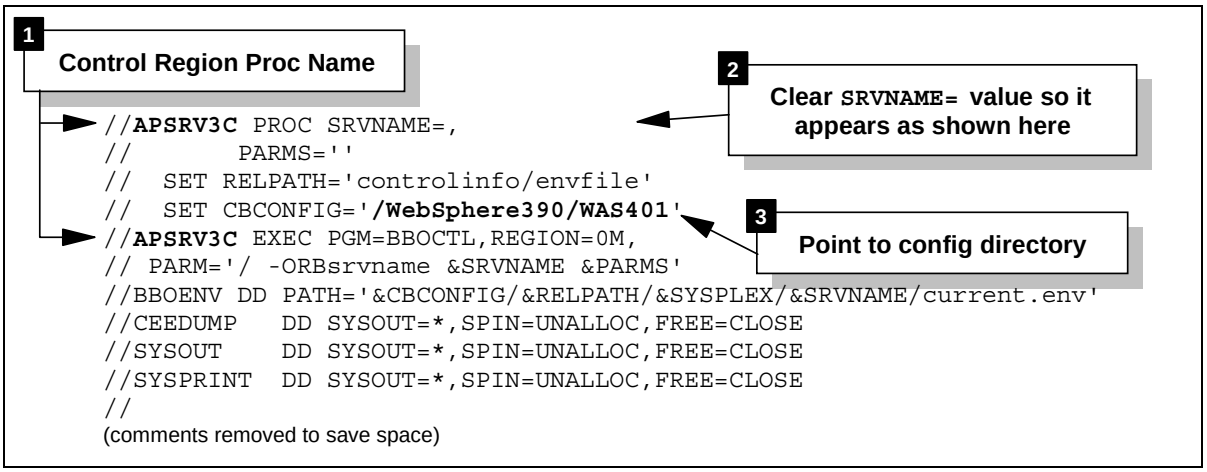
Create JCL Start Procedures

☐ Copy the following supplied sample members over to `SYS1.PROCLIB`, *renaming* them during the copy:

From: **TEICHMN.WAS401.CNTL**	*Copy*	*To*: **SYS1.PROCLIB**
BBOASR2	⇨	APSRV3C
BBOASR2S	⇨	APSRV3S

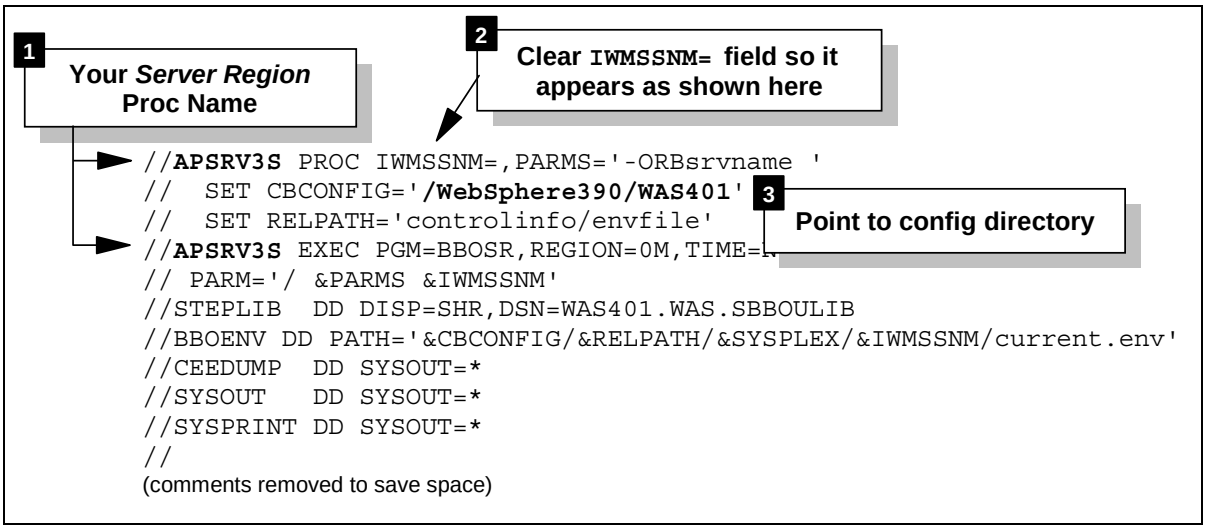
> ***???*** You named the *control region* procedure in the SMS tool when you created the server, and you named the *server region* procedure in WLM when you defined the application enviornment. The names you provide these procs does matter: it must match the other definitions.

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



??? 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):



??? Why clear the `IMWSSNM=` 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.

Create JVM Properties File and Trace Settings File

??? This step involves creating two configuration files by hand. Since not everyone will need custom JVM properties or trace settings, WebSphere has you do this by hand rather than automatically.

☐ Create the following HFS file:

`/WebSphere390/WAS401/controlinfo/envfile/WSLPLEX/APSRV3S1/`**`jvm.properties`**

Permissions **644**

Owner ..... **WASSMSS**

☐ Create the following HFS file:

`/WebSphere390/WAS401/controlinfo/envfile/WSLPLEX/APSRV3S1/`**`trace.settings`**

Permissions **644**

Owner ..... **WASSMSS**

☐ Edit the `jvm.properties` file and add the following (***all on one line in the file***):

```
com.ibm.ws390.trace.settings=
   /WebSphere390/WAS401/controlinfo/envfile/WSLPLEX/APSRV3S1/trace.settings
```

☐ Edit the `trace.settings` file and add the following (*two lines in file*):

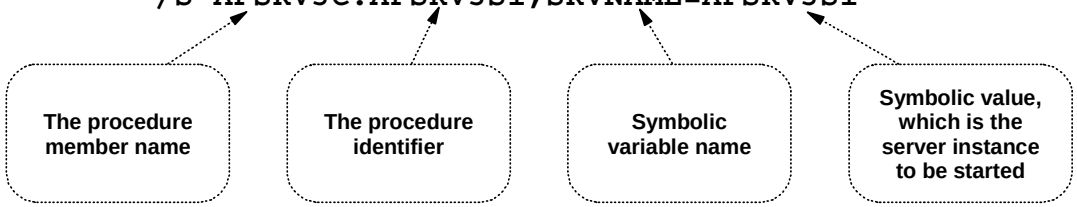
```
#com.ibm.*=all=enabled
com.ibm.*=all=disabled
```

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:



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.

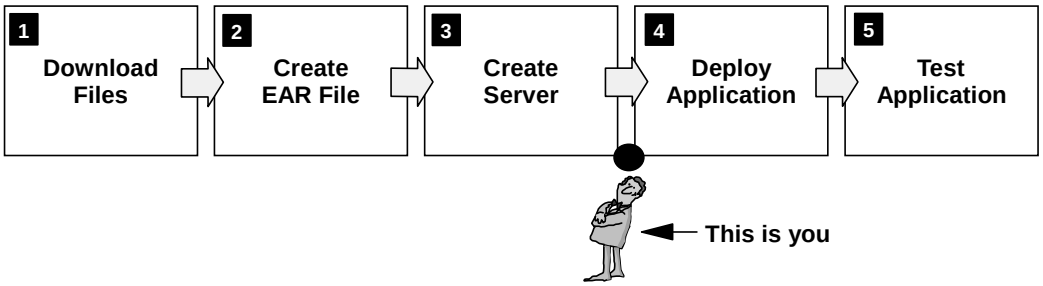
+BBOU0698I REGISTERING SERVER APSRV3

??? 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**

Delete Existing PolicyIVP Application

??? The application name "PolicyIVP" was deployed into WASASR2 in a previous lab, and the application name you just packaged in this lab is *also* named "PolicyIVP." Technically speaking, deleting it from WASASR2 is not required as long as the JNDI names are unique.

- ☐ Start the SMS EUI (or go back to it if it's still started) and add another converstaion. Name the conversation something indicating PolicyIVP is being deleted from WASASR2.

  ??? For a refresher on how to do this, see "Start SMS EUI and Add Conversation" on page 11.

- ☐ Expand the tree and locate the PolicyIVP application under the WASASR2 J2EE Server.
- ☐ Right-click on the PolicyIVP application and select "Delete."
- ☐ Select the conversation name, right-click and select "Validate"

  ??? Hint: "Validate, Commit and Activate the Conversation" on page 15.

- ☐ After validation, right-click and select "Commit."
- ☐ Select "Build," then "Complete" and then "All Tasks"
- ☐ Activate the conversation.
- ☐ Check to see if the server WASASR2 is started on the OS/390 system. If not, start it.

  ??? The act of *de-registering* a deleted application is completed when the server in which the application was deployed is restarted. If the server was already active when you deleted the application, the server would have been automatically stopped and restarted by WAS. If the server was down when you deleted the application, you need to restart it to complete the de-registration of the application.

Deploy <u>Your</u> Version of PolicyIVP Application

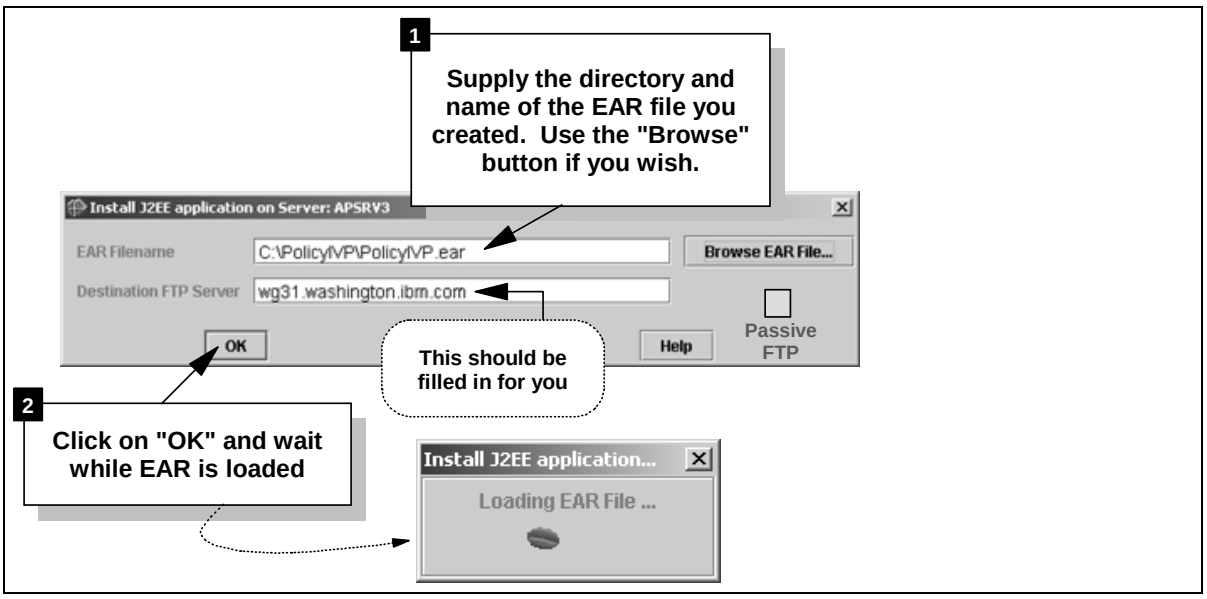
Add New Conversation

- ☐ You should be familiar with this process by now. If you want a reminder, see "Start SMS EUI and Add Conversation" on page 11.

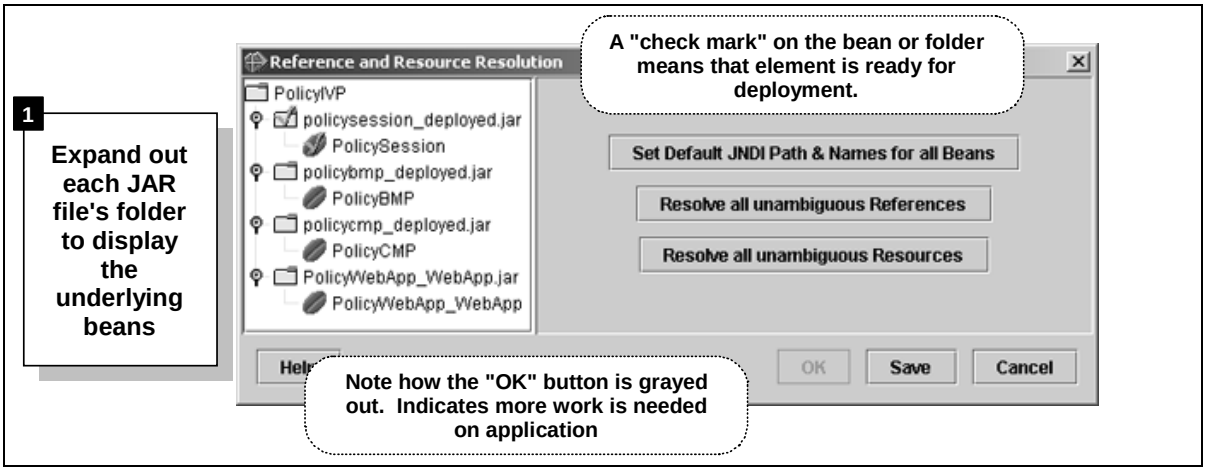
Import the Application

- ☐ 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:

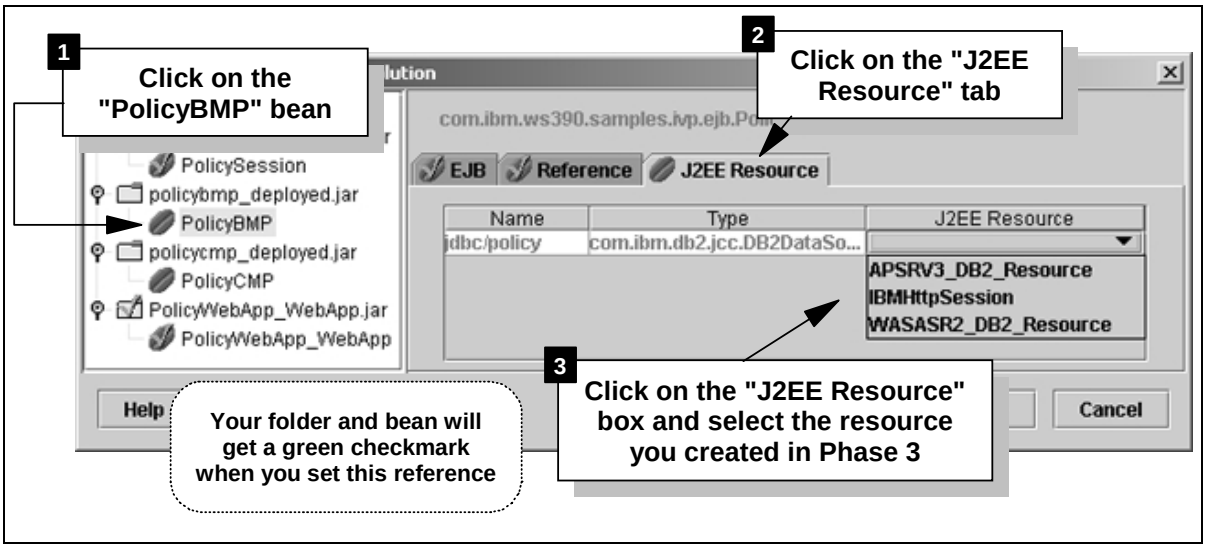


- ☐ You should now have a "Reference and Resource Resolution" panel. Toggle the switches next to each folder to expose the "beans":

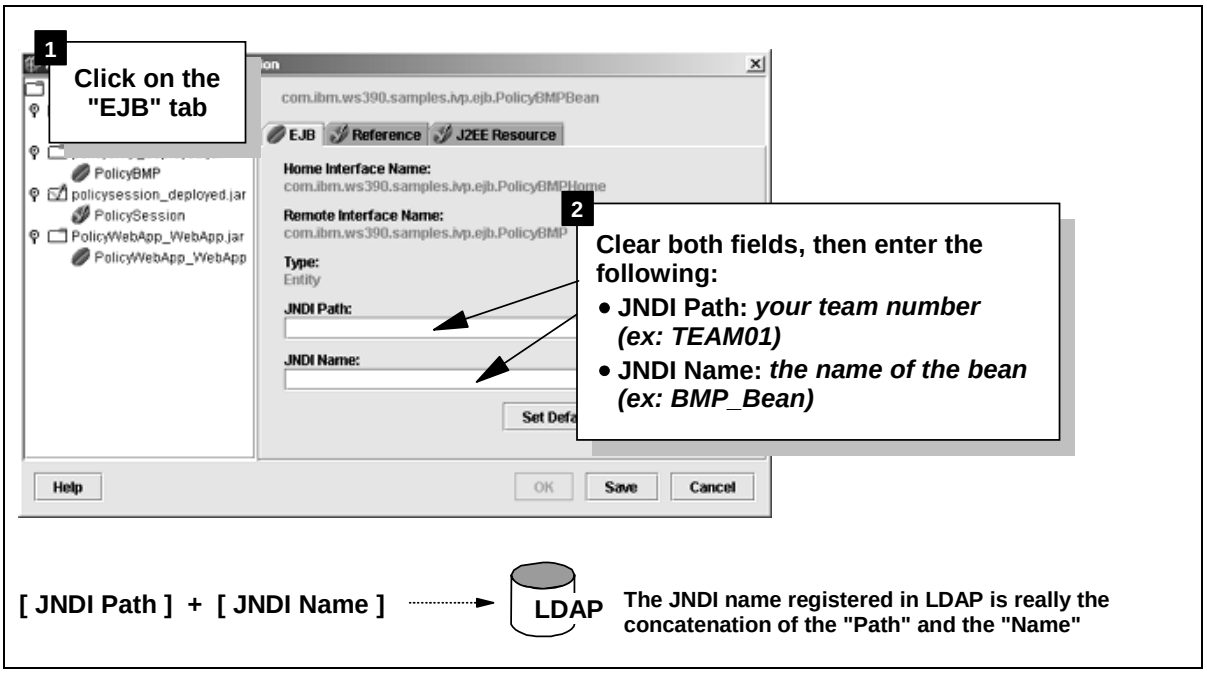


BMP Bean: resolve resource reference and set JNDI name

☐ 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:



??? You could (but don't here) click on the "Set Default JNDI Name and Path" button. 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[note], 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.

- ☐ 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 (`TEAM##` 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 (`TEAM##` and `Session_Bean`)

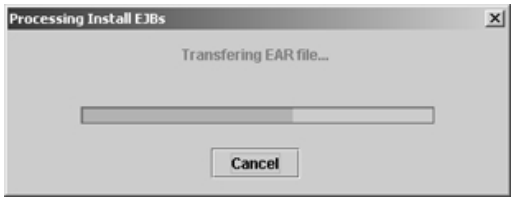
WebApp: set JNDI name

- ☐ Set the JNDI name for the webapp using the same naming convention (`TEAM##` and `WebApp`).

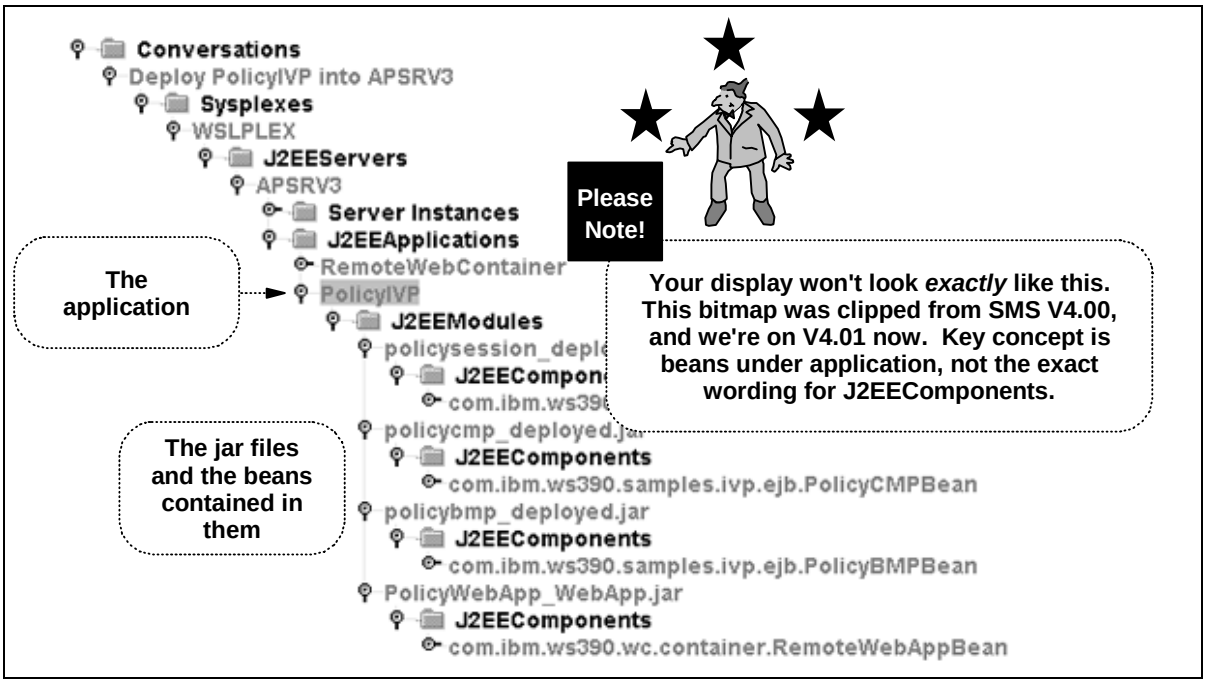
Transfer the EAR File to the WAS Server

??? 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:

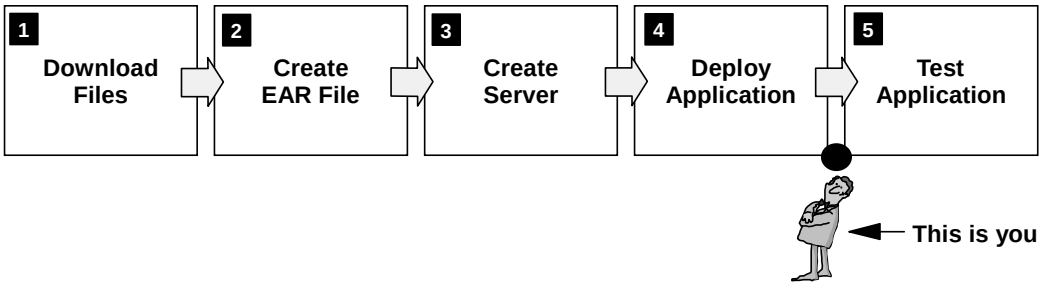


Validate, Commit and Activate the Conversation

☐ Same process as before. (See "Validate, Commit and Activate the Conversation" on page 15 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.

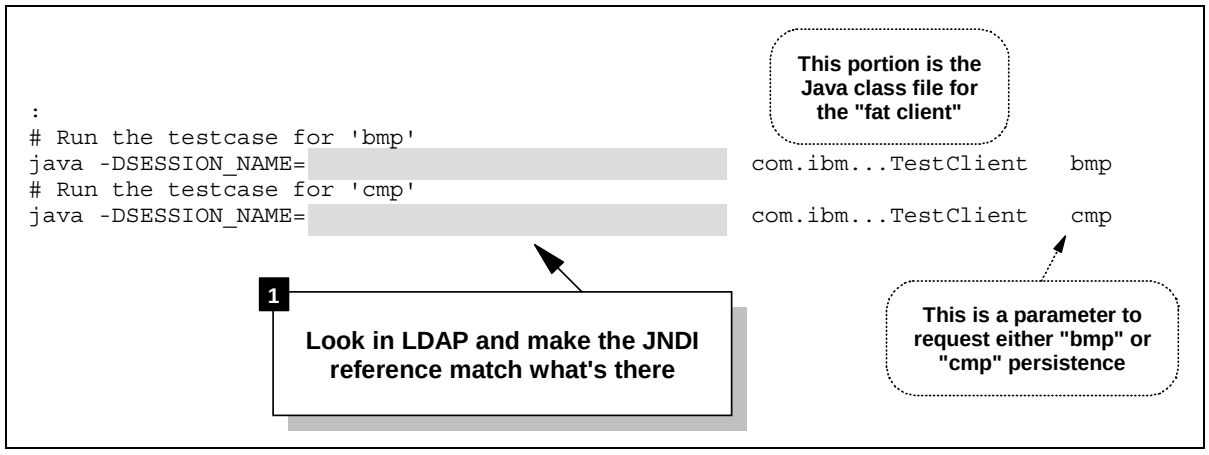
Phase 5: Test Application

??? 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

??? The copy of `ejbivp.sh` you already have in your `/u/user1/IVP` directory was used earlier in this class to validate the PolicyIVP application you deployed initially. You need to modify that so the `-DSESSION_NAME=` value points to the JNDI name of your newly-deployed Session bean.

- ☐ Edit `/u/user1/IVP/ejbivp.sh` and make the following changes:



Run the shell script

- ☐ Go into OMVS (or via a Telnet terminal), change to the `/u/user1/IVP` directory, and issue the command:

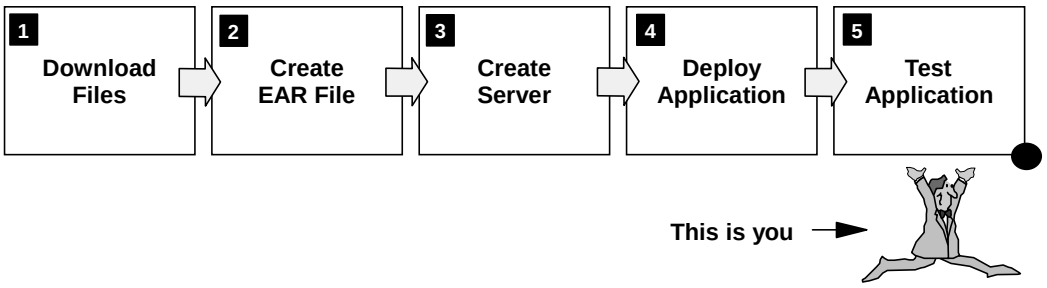
  `./ejbivp.sh`

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

```
***********  bmp bean will be run!
Look up policy session home
Obtaining policysession bean jndi name...
The policysession 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 policysession bean jndi name...
The policysession 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: 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

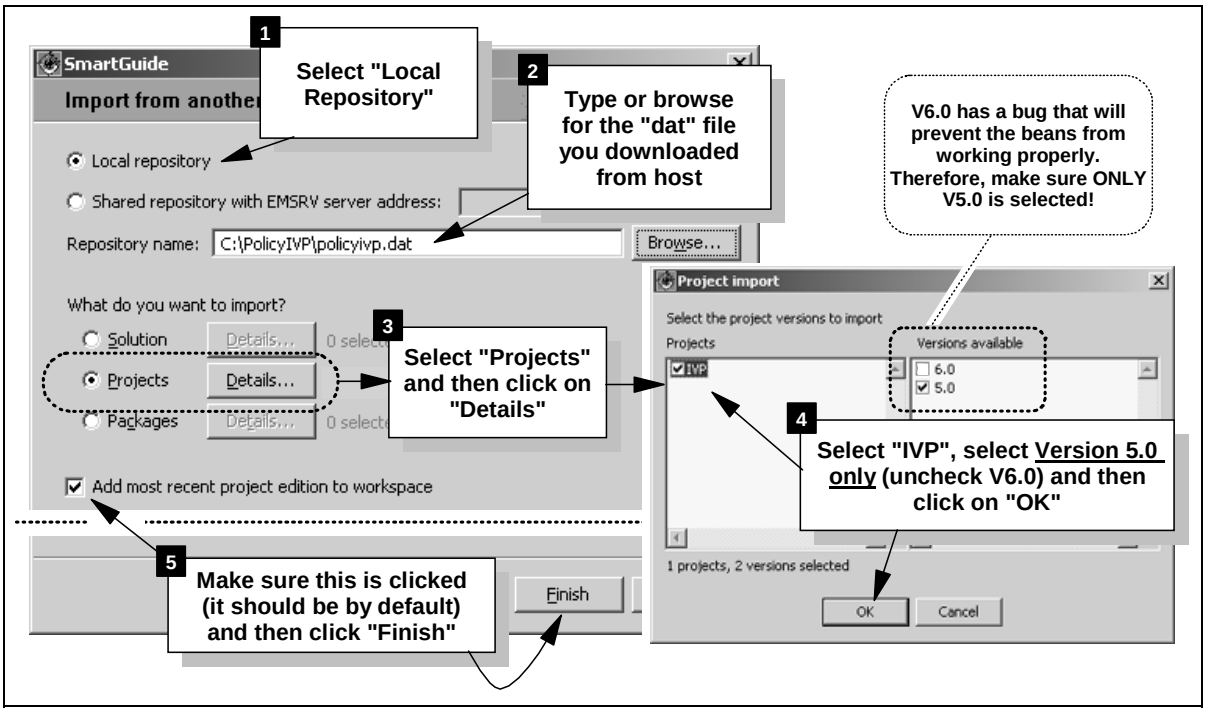
???	VisualAge for Java maintains its work in a single large database-like file called a "repository." That repository has a file extension of "dat." One feature of VAJ is the ability to *export* a project out of VAJ into another "dat" file, which can then be *imported* into another copy of VAJ. This 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.

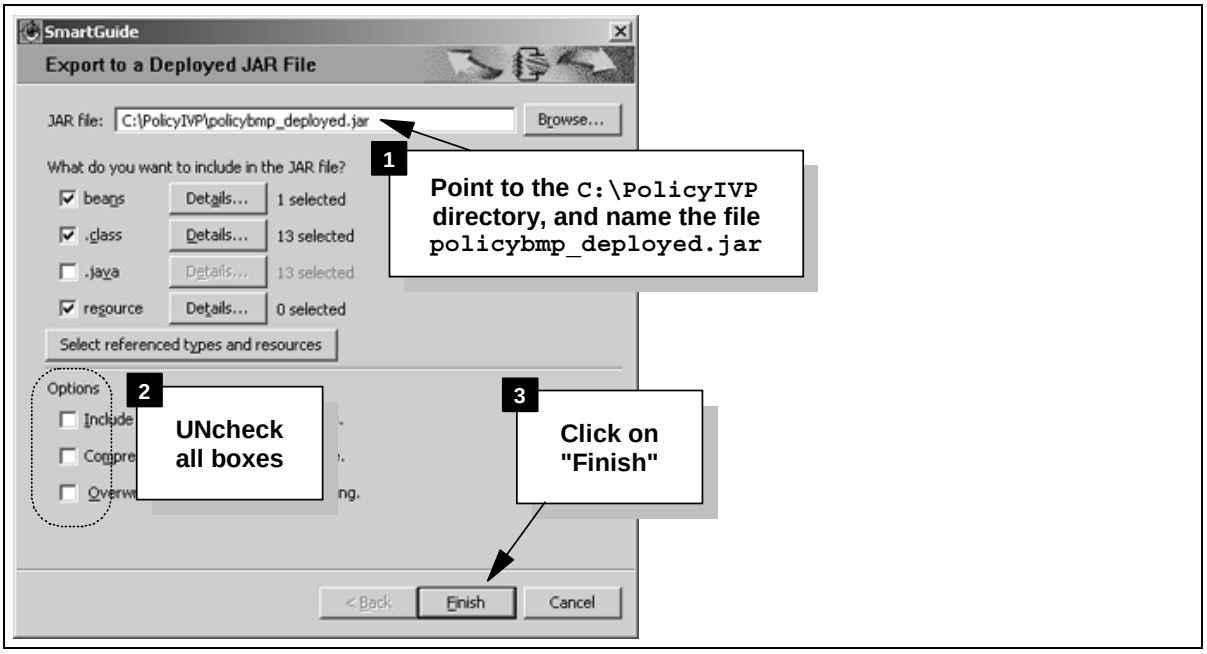
Export JAR Files

> ***???*** 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\policycmp_deployed.jar`

- ☐ Repeat the process once more, this time for the ***PolicySession*** bean, exporting it to:

  `C:\PolicyIVP\policysession_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: `policysession_deployed.jar`, `policybmp_deployed.jar`, `policycmp_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:

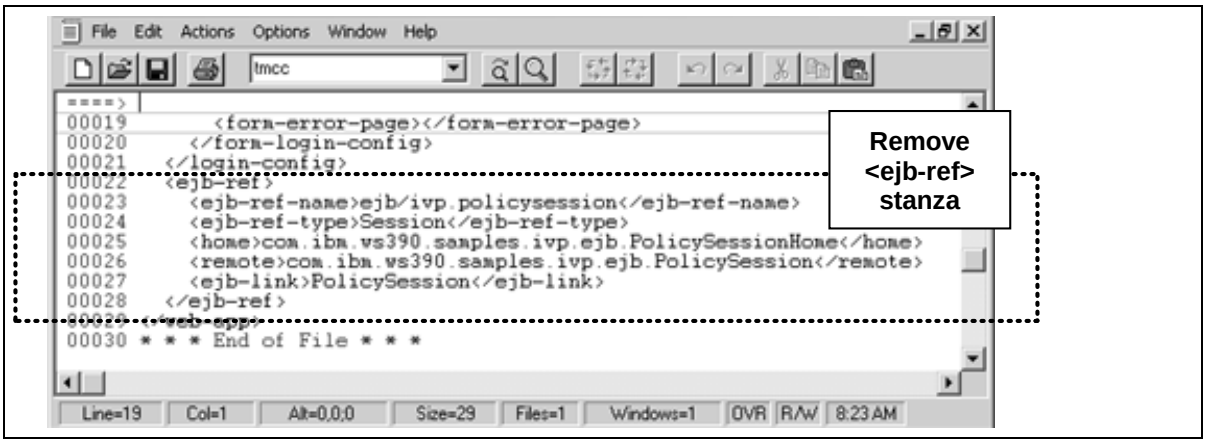
  `/usr/lpp/WebSphere401/samples/PolicyIVP/ejb/PolicyIVP.ear`

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.

End of Document