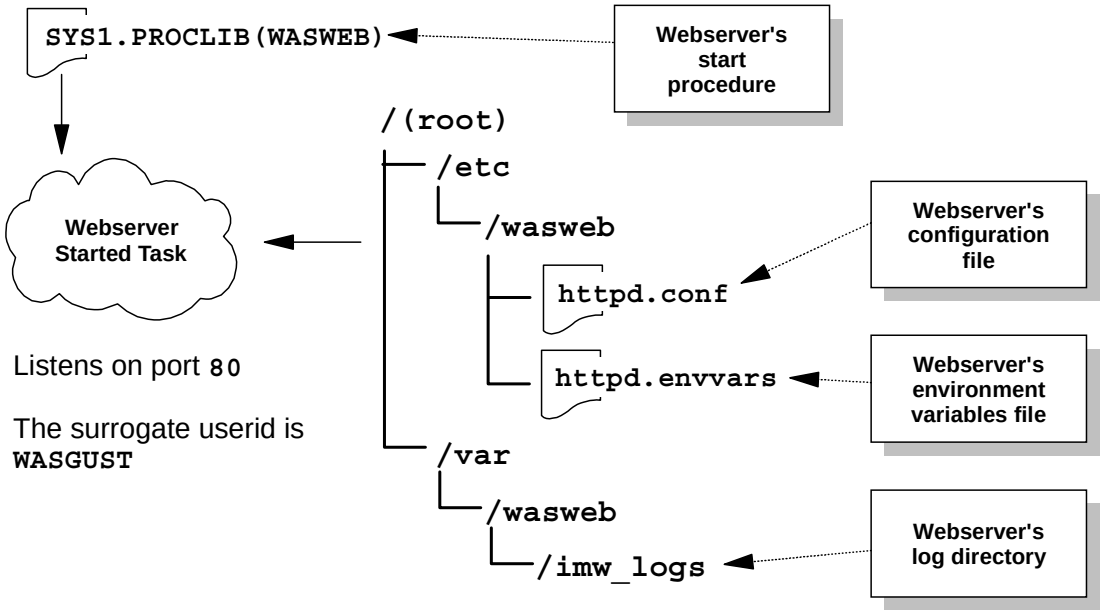
Lab:
Configuring Web Applications

(This page intentionally left blank)

Reference: Initial State of the Environment

The initial configuration structure looks like this:



Verify that the Webserver Comes Up

As an initial test, start the webserver and make sure you can reach the default home page:

- ☐ Start the webserver with proc `WASWEB` (already in `SYS1.PROCLIB`)
- ☐ `http://<host name>`

  You should get a page that looks like this:

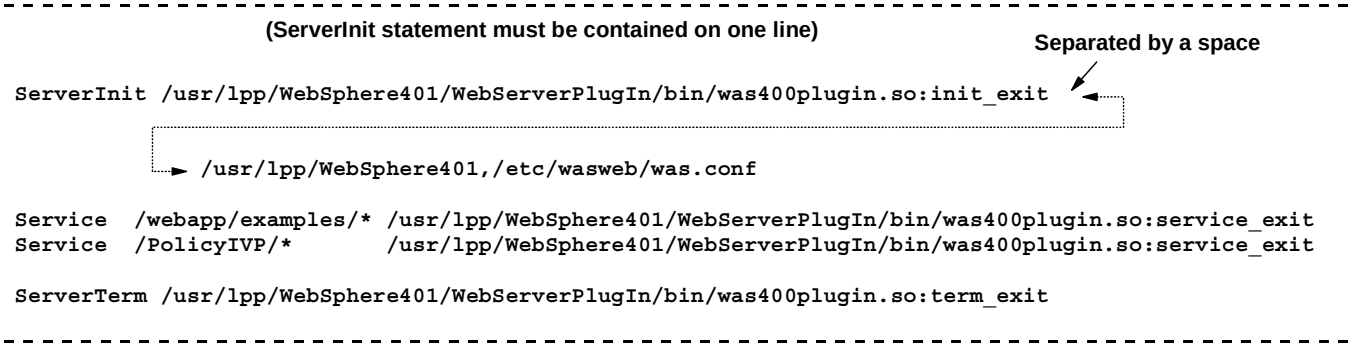
- ☐ Stop the webserver.

Configure the WAS 4.01 Plugin

This process involves updating the `httpd.conf` file to tell the webserver to initialize the plugin, the `httpd.envvars` file to give the webserver some knowledge of the environment, and creating a `was.conf` file for use by the plugin.

Update httpd.conf file

- ☐ Locate the string `WAS directives` in the file, then *comment out* the `Service` statement that immediately follows it. See "Reference: Initial State of the Environment" on page 1 for an indication of where the `httpd.conf` file resides.
- ☐ Add the following, being careful with spelling and case:



Update httpd.envvars file

- ☐ Add a **`JAVA_HOME`** variable and set its value to the location where the JDK 1.3 libraries reside (**`/usr/lpp/java2/J1.3`**). See "Reference: Initial State of the Environment" on page 1 for information on where the `httpd.envvars` file resides.
- ☐ Add the WAS 4.01 plugin's message catalog to the **`NLSPATH`** variable: **`/usr/lpp/WebSphere401/WebServerPlugIn/msg/%L/%N`**
- ☐ Provide the following two variables to give the plugin knowledge of the naming server:

  `RESOLVE_IPNAME=`**`WG31.WASHINGTON.IBM.COM`**
  `RESOLVE_PORT=`**`900`**

- ☐ Save the file.

Create log directory and work directory for WAS 4.0 plugin

The plugin has a logging facility that may prove useful, and it's best to have the WAS 4.01 plugin write those logs to an HFS directory. In addition, you need to provide the WAS 4.01 plugin needs a "scratch" directory in which to do some work.

- ☐ Create the directory **`/var/wasweb/work`** and give it permissions **`777`**
- ☐ Create the directory `/var/wasweb/was_logs` and give it permissions **`777`**

Create and update the was.conf file

At the present time no `was.conf` file exists in your `/etc/wasweb` directory. You need to provide one because on the `ServerInit` statement in `httpd.conf`, the second parameter on the statement points to `/etc/wasweb/was.conf`. If the plugin can't find the `was.conf` file explicitely named on the `ServerInit`, it'll fail to initialize.

- ☐ Copy the supplied default `was.conf` file:

From: `/usr/lpp/WebSphere401/WebServerPlugIn/properties/was.conf`

To: `/etc/wasweb/was.conf`

- ☐ Edit the file and make the following two updates:
  - Add the directory **`/var/wasweb/was_logs`** to the **`appserver.logdirectory`** statement.
  - Add the directory **`/var/wasweb/work`** to the **`appserver.workingdirectory`** statement.
- ☐ Save the file.

Restart the webserver and verify the plugin initialized

- ☐ Start the webserver.
- ☐ Now browse the `SYSOUT` of the `WASWEB` started task and find the following string:

  `:-)`

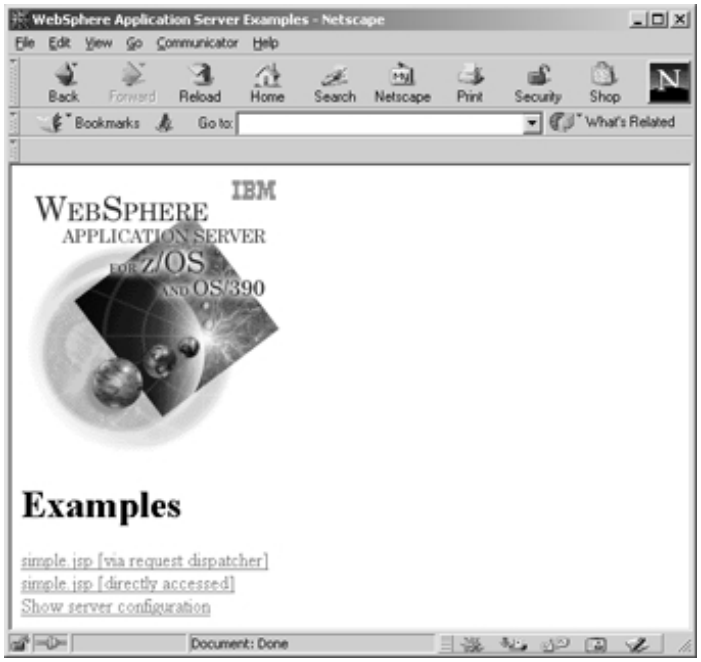
  Yes, that's a "smiley face", and that's an indication that the plugin initialized okay. The plugin sometimes takes a few moments to initialize, and may not be up even though the webserver is operational. Give it a few moments and try again if you don't see it initially.

  If you still can't find the smiley face, search on the "frowny face" **`:-(`**. The webserver will throw that message if something prevented the plugin from initializing. Common causes for plugin initialization failure:

  - `JAVA_HOME` variable in `httpd.envvars` not set correctly.
  - Mistyped directory or file name on `ServerInit` statement in `httpd.conf`. Check for case problems.
  - Second parameter on `ServerInit` statement points to `was.conf` file and directory that does not exist.
- ☐ Once you've verified the smiley face, issue the following URL from your browser:

  `http://<host>/webapp/examples/index.html`

  You should see a screen that looks something like this:



If you receive this, it is an indication that your URL was successfully mapped over to the WAS 4.01 plugin using the `Service` statements in `httpd.conf`. *But beware, at this point you have not exercised the WAS 4.01 Runtime. This screen came out of the plugin locally, and no interaction with the WAS 4.01 runtime has yet taken place.*

- ☐ Browse the `httpd.conf` file and find the `Service` statement that would be matched when a URL of:

  `http://<host>/webapp/examples/index.html`

  is received. Do you see how a URL mapping to the `/webapp/examples/*` mask gets mapped to the `was400plugin.so:service_exit` routine?

- ☐ Now browse the `was.conf` file and find the `deployedwebapp` statements (at the bottom) and note the statement with `rooturi=/webapp/examples`. This block of statements (`deployedwebapp` and `webapp`) is what defines the plugin's local verification program.

  Because these definitions exist in the `was.conf` file, the request received was processed locally, rather than being passed over to the WAS 4.01 Runtime.

Web Container Setup

The web container is actually created when the server instance is created, but it doesn't "come to life" until the `webcontainer.conf` file is brought into play.

Do the following:

- ☐ Locate the private directory for your application server instance. This is the directory in which the server instance's `current.env` file resides. Write the directory here:

  _____

  Hint: Start at `/WebSphere390/WAS401` and work your way down from there.

- ☐ The sample `webcontainer.conf` file is located in the `/usr/lpp/WebSphere401/bin` directory. Copy the file over to the private directory of your server instance and make sure it has permission **`644`** and an owning ID of **`WASSMSS`**.
- ☐ Edit the `jvm.properties` file in the server instance's private directory and point to the `webcontainer.conf` file you just copied into the directory. The statement that defines the web container's configuration file is this:

  `com.ibm.ws390.wc.config.filename=`

  Code the *full path and file name* of the `webcontainer.conf` file.
- ☐ Edit the `webcontainer.conf` file and update to properties in the file:

  `host.default_host.alias=wg31.washington.ibm.com`

  `host.default_host.contextroots=/`

That's it ... the web container has its configuration file and the virtual host and contextroots are configured in the most basic fashion. Now you need to test to see if the IVP you deployed earlier works.

Before you do that, you need to refresh your environment:

- ☐ Stop the application server control region
- ☐ Stop `WASWEB`
- ☐ Start the control region
- ☐ Start `WASWEB`
- ☐ Review the SYSPRINT of the server region and insure the proper `webcontainer.conf` file was used, and the `/PolicyIVP` application was bound to your virtual host.
- ☐ Invoke the `/webapp/examples/showCfg` application and make sure the PolicyIVP application appears in the appropriate format.

Test the IVP

Get the PolicyIVP HTML Front Page

- ☐ Issue the URL:

  `http://wg31.washington.ibm.com/PolicyIVP/cebit.html`

  You should receive a screen that looks like this:

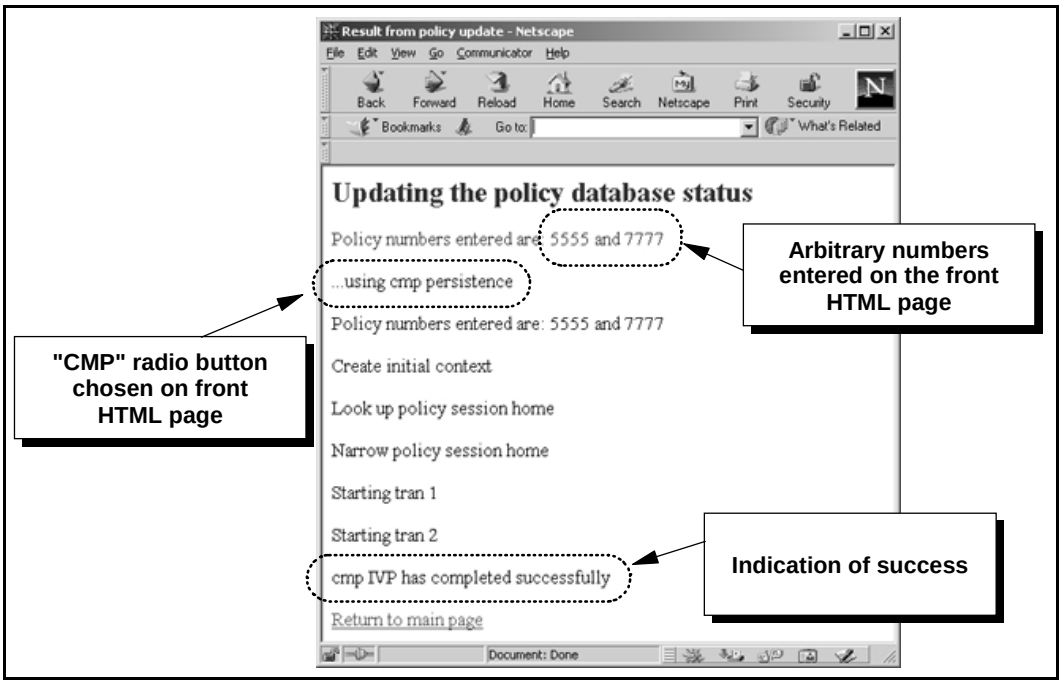
If you receive this screen, it means the link between the plugin and the WAS 4.01 runtime has been exercised, and the PolicyIVP web application components have been properly deployed into the web container. At this point, however, no RMI/IIOP flows to EJBs have occurred. This is just a static HTML page that WAS 4.01 has served out.

Drive the EJB

- ☐ With your browser still at this HTML screen, view the source of the HTML. Find the line that starts `<form METHOD=GET ACTION=...` . What is the "action" that will take place when the "submit" button is clicked?"

  _____

- ☐ Provide a number in the input box "Policy Number 1" and a number in the input box "Policy Number 2", select either CMP or BMP and then click "Submit". If things are working properly, you should see an output page that looks like this:



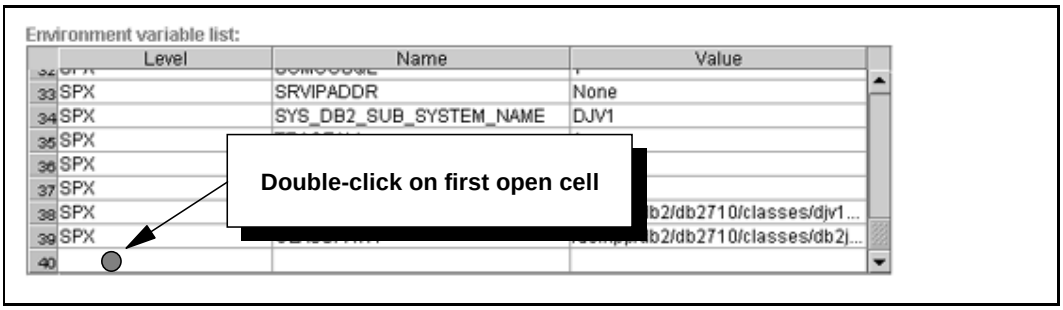
Getting this screen means the HTML submit button flow drove the servlet in the web container, which in turn invoked an RMI/IIOP flow to the session bean, which then invoked the method on the CMP or BMP bean, which then put the information you entered into the database. Once done, the servlet formatted the HTML and sent the results to your browser.

Configure the Transport Handler

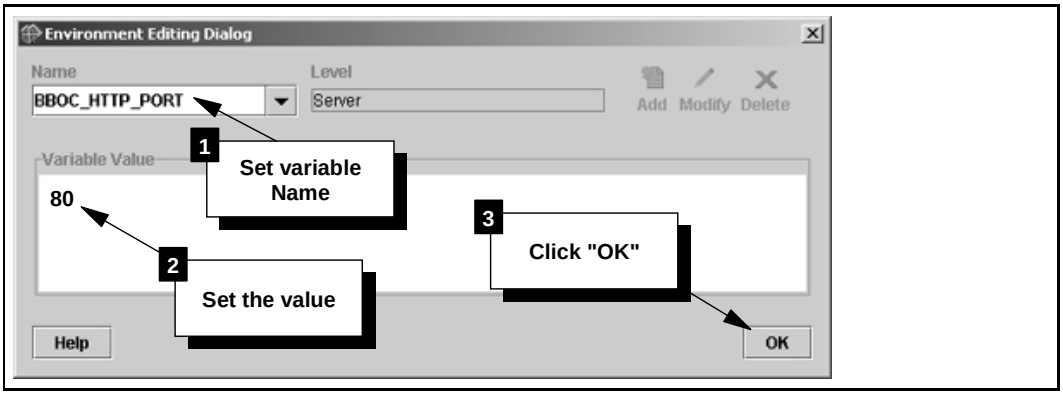
The objective here is to configure the Transport Handler to bind to port 80 and to make use of the existing virtual host and context settings in the `webcontainer.conf` file.

Note: We are going to have the Transport Handler bind to port 80. That means for this to work you must shut down the Webserver (both can't bind to 80 at the same time). Doing so will make certain that you are actually using the Transport Handler: with the Webserver shut down, the browser will have only one way to get to PolicyIVP, and that's through the Transport Handler.

- ☐ Stop the Webserver.
- ☐ Start the SMS EUI tool. Create a new conversation, locate the `APSRV3` server
- ☐ Modify the server, and locate the "Environment Variable List." Double-click on the first empty cell after the last "SPX" (SYSPLEX) level variable in the list:



- ☐ Set the variable as **`BBOC_HTTP_PORT`**, the level as *Server* (which should be what it shows by default because you're modifying the server), and set the value as **`80`**:



- ☐ Save the changes, then validate, commit and activate the conversation.
- ☐ After the conversation has been activated, *browse* the `current.env` file and make sure the `BBOC_HTTP_PORT` variable has been set.
- ☐ Activating a conversation would have restarted your server. Check to see if the `APSRV3` server is started. If not, start it.
- ☐ Issue the **`NETSTAT`** command from the TSO Option 6 panel and see if the `APSRV3C` ID has hold of port 80.
- ☐ Clear your browser's cache, then issue the URL:

  `http://wg31.washington.ibm.com/PolicyIVP/cebit.html`

  Do you get the front page of the PolicyIVP application?
- ☐ Set the policy and BMP/CMP values and drive the application again.

End of Document