

WebSphere Application Server 5.0 Accelerated Pr

\$Revision: 1.4 \$

Blaine Simpson. blaine.simpson@admc.com

Table of Contents

Introduction	2
Environment Setup	3
High-level Architecture	5
Filesystem Layout	6
Command-line	8
serverStatus.sh	8
startServer.sh	8
stopServer.sh	8
dumpNameSpace.sh	9
ivt.sh	9
ejbdeploy.sh	9
wsadmin.sh	9
ws_ant	10
ps and kill	10
Graphical	12
Admin Console	12
assembly.sh	14
Deployment	17
wsadmin	18
Limitations	18
WAS JACL Administration Objects	18
Useful WAS Administration Commands	18
Comparison to Administrative Console	19
Deployment script	19
Data Sources	21

Author: Blaine Simpson. blaine.simpson@admc.com

\$Revision: 1.4 \$

The software which is referred to in this document and which is hosted on the admc.com web site, is distributed for free according to the Gnu Public License.

All-in-one HTML

<http://admc.com/blaine/howtos/ws50/ws50-all.html> (`ws50-all.html`)

Split-up HTML

<http://admc.com/blaine/howtos/ws50/index.html> (`index.html`)

Info

<http://admc.com/blaine/howtos/ws50/ws50.info> (`ws50.info`) (useful if you are on UNIX and want to view this in a terminal window instead of with a web browser, and without losing the linking features).

Introduction

Keep a browser handy, with the WAS InfoCenter at <http://publib7b.boulder.ibm.com/webapp/wasinfo> bookmarked.

Most items discussed in this document apply to both Windows and UNIX setups, but the command-line commands all use Bourne-compatible shell syntax (unless otherwise noted), UNIX filesystem layout, and UNIX path delimiters. Note that there are a handful of basic features of the Administrative Console which just do not work with the lightweight server. When I discuss the Administrative Console, I mean an Administrative Console running on a standalone WAS instance.

This document does not explain installation, since that is covered by a different HOWTO. This document does explain how to **use** and **administer** the WAS server, with the specific goal of making the attentive reader a *power user* in a very short time.

Web Server administration is nearly entirely separate from WAS administration. Unlike other application servers, you need to take care of starting and stopping your front end web server independently of WAS.

Some names you will need to know. Instance name and node name appear in some directory paths, so I'm defining the following <KEYWORDS> to make this document easier to understand.

- <**INST**> If you are running an alternate instance (i.e. if your instance is not based at \$WAS_HOME), then this is the "name" of your instance. (The main instance has no name).
- <**NODE**> WAS's identifier for this instance. (Don't confuse this with "hostname", like the WAS docs often do. You can install 10 different instance/nodes on one host.) Regardless of instance name, node names themselves should be unique for each instance (otherwise it will be impossible to use them at the same time in a Network Deployment setup).
- <**CELL**> WAS's identifier for this cell. By default, this is the same as <NODE> if this is a main instance, or the name of the parent (creator) instance otherwise.

There are WAS programs for remotely administering nodes, and for building and deploying standalone J2EE application clients. I do not cover those programs here since these are atypical uses.

Environment Setup

You should run the programs in this document (other than web interfaces) as the owner of the web instance or via `sudo`. The environment for that account will need to be set up. If you own your own WAS instance, then, once your environment is set up, you can run all the WAS commands. If your instance is owned by a shared account, then you will have to log into that account or use `sudo`. Do not attempt to run these commands as any other user (especially `root`) unless you have explicitly set up environments for file sharing. (I recommend against it for WAS because it is very easy for one user to mess up everything for all other users).

N.b. In general, you should not do everything as the instance owner, but only stuff related to the WAS "server", such as deploying apps or restarting the WAS server. If you are a developer, you should do all of your editing and compiling and building using your personal account.

You must set and exported the following variables.

‘\$WAS_HOME’

Main directory where the WAS software is installed. WAS installs to `/opt/IBM/WebSphere/AppServer` by default.

‘\$ORACLE_HOME’

Home for the Oracle DB client or server software installation. This is typically something like `/opt/oracle/product/8.1.7.3`.

‘\$LD_LIBRARY_PATH’

On UNIX, this value should always be `$ORACLE_HOME/lib`.

If you are running an alternate instance, then you also must *source* `$USER_INSTALL_ROOT/bin/setupCmdLine.sh`, like

```
. /home/blaine/inst2/bin/setupCmdLine.sh
```

(If we have C-shell users, then let me know and I'll convert it to C-shell).

(In addition to other things, this will set the variable `$USER_INSTALL_ROOT` to the instance home directory. `$USER_INSTALL_ROOT` is not set for users using the default WAS instance.)

Add `$WAS_HOME/bin` to your search path.

To really make things easy, set up the login files for the account that you will run the WAS commands from.

1. If your user own the instance, then this is your account.
2. If your instance owner is a shared UNIX account and you plan to run the commands from that user's environment (like by running `su -` or `sudo su -`) then set up the instance owner account.
3. If your instance owner is a shared UNIX account and you plan to run the commands via `sudo` from your personal account, then set up your personal account.

(In many situations, you will want the ability to use #2 or #3. In that case, set up both the shared account and your personal account.)

Download <http://admc.com/blaine/howtos/was.profile> and <http://admc.com/blaine/howtos/wa> and rename them to `.profile` and `.kshrc` in the user's home directory. Then edit `.profile` to enter your local path(s) as the file says to.

If you use some other shell, or if you have existing `.profile` and/or `.kshrc` file(s) that you don't want to clobber, then merge the contents of the downloaded files into your existing `.profile` and `.kshrc` files.

After you do this, verify that you can log in as this user and successfully run `startServer.sh server1` and `startServer.sh server1` (without making any manual environment settings).

The rest of this section is only applicable to setups using a shared account as the instance owner (i.e, your personal account is not the instance owner). If you own your instance, then you can safely skip the rest of this section.

For the remainder of this section, I use the name *websphere* to represent the WAS instance owner.

N.b. Note that if you just get a shell as the new user by running something like `sudo -u websphere -s` or `sudo -u websphere ksh`, you will not get a proper environment for the target user. If you want to get a full environment set up as if you logged in as the instance owner, then you need to run something like `sudo su - websphere` or `su - websphere` (or actually log into the server as websphere).

High-level Architecture

The Architecture is simple once you understand the high-level architecture objects.

Server In an architectural context, a Server is a JVM process running WAS. You can very easily set up one WAS instance to have two servers, say *server1* and *server2*. If you ran `ps` after starting them, you would see two java processes. Each server has its own independent JVM, class loaders, etc. This is why JVM parameters and many performance-related settings are made on the server level. The default server name is *server1*.

Node or Instance

A Node or an Instance is a collection of servers housed under a single directory. All the servers of this Node/Instance may be administered through the same wsadmin or Admin Console session. The IBM documentation usually calls the main instance (which has the WAS software installed under the main directory) a *Node*, and the other instances *Configuration Instances*. This makes this very simple concept confusing for beginners. Just remember that even though IBM never says it, alternative WAS configuration instances are configured exactly the same as any standalone node. There is no architectural difference other than the location of the WAS software.

Alternate Configuration Instance

An Alternate Instance is just a Node that does not have the WAS software installed under the main directory. Whenever the WAS software is installed, a "main instance" is installed. All other instances are created by running "`wsinstance.sh`", and are alternative configuration instances. Node names should be unique, whether the node is a main instance or an alternate configuration instance.

Cell Normally, when you run a wsadmin or Admin Console command, you are sending commands to a Node (aka Instance). If, however, you have a *Network Deployment* installation, administrative commands can be applied to a group of nodes called a Cell. A cell is just a grouping, for administrative purposes, of nodes/instances. When doing cell administration, IBM always calls instances "nodes".

The default cell name is the same as your node name.

Cluster A collection of servers within a cluster which run as a single, distributed unit for load sharing and high availability reasons.

Filesystem Layout

\$WAS_HOME/bin

contains the WAS programs

\$WAS_HOME/bin/wsinstance

contains config files and programs for creating alternative instances.

I give the paths of all remaining items relative to `$USER_INSTALL_ROOT`. In the event that you are running the default instance, just substitute `$WAS_HOME` for `$USER_INSTALL_ROOT`.

\$USER_INSTALL_ROOT/logs/server1

Log files produced by running your WAS instance will go into this directory.

\$USER_INSTALL_ROOT/logs/server1/SystemOut.log

Most errors end up here (not in the `SystemErr.log` file). Unfortunately, a trace of AdminConsole usage goes into this file too.

\$USER_INSTALL_ROOT/logs

Log files produced by running utilities generally go to logs in this directory, but be aware that many files in this directory are binary.

\$USER_INSTALL_ROOT/installedApps.

When you *save* applications to WAS, J2EE ears and components get written under here.

Applications under here are basically expanded `.ear`'s, with component `.war`'s (but not `.jar`'s) also expanded. The deployment descriptors (including those within `.jar`'s) reflect deployment-time settings and bindings. (The deployment descriptors also get written under `$USER_INSTALL_ROOT/config`. See below).

\$USER_INSTALL_ROOT/config/cells/plugin-cfg.xml

The production web server plugin config file.

\$USER_INSTALL_ROOT/config/cells/<CELL>

Deployment and config files specific to this cell. (Note that in a single-node setup, the cell is named after the node name).

\$USER_INSTALL_ROOT/config/cells/<CELL>/nodes (and subdirs)

Deployment and config files pertaining to nodes.

\$USER_INSTALL_ROOT/config/cells/<CELL>/applications

Deployment and config files pertaining to deployed applications. Note that these are only configuration and deployment files under here. Class files and binaries other than `.ear` files are not stored here— go to `installedApps` for them.

\$USER_INSTALL_ROOT/config/cells/<CELL>/nodes/<NODE>_<INST>/servers/server1/server.xml ■

The main service configuration file.

\$USER_INSTALL_ROOT/config/cells/<CELL>/virtualhosts.xml

Another service configuration file.

`$USER_INSTALL_ROOT/wstemp`

This is where files are *tentatively* deployed to. When you *Save* with `wsadmin.sh` or the Admin Console, your items are promoted from `wstemp` to the `installedApps` and `config` branches.

Command-line

The WAS command-line programs which every WAS Administrator should know.

serverStatus.sh

TYPICAL USAGE

```
serverStatus.sh server1
```

HELP

```
serverStatus.sh -help
```

startServer.sh

TYPICAL USAGE

```
startServer.sh server1
```

HELP

```
startServer.sh -help
```

This is a good script, with a couple important exceptions. It very nicely tells you what it's doing and where it's sending output to, and the script returns when the app server is running successfully (unlike startup scripts for some other App servers).

Be aware that if you are running a default instance, you don't need your environment set up at all (\$WAS_HOME will be determined by location of the startup script). However, if you are running a non-default instance, you need to set \$WAS_HOME and then source \$USER_INSTALL_ROOT/bin/setupCmdLine.sh (not \$WAS_HOME/bin/setupCmdLine.sh!).

BEWARE! Many of the messages about the environment printed to the startup log (.../logs/server1/startServer.log) are WRONG. If you edit the IBM scripts, be aware that some variables (including \$WAS_CLASSPATH) are obsolete and are not used at all.

If you have Oracle data sources, be aware that the Oracle JDBC drivers will not work unless you have \$LD_LIBRARY_PATH and \$ORACLE_HOME set and exported properly when startServer.sh is invoked.

If you run startServer.sh using sudo, then be sure to get the patched version of startServer.sh (as documented in the Installation HOWTO), otherwise your \$LD_LIBRARY_PATH setting will be lost when you use sudo.

stopServer.sh

TYPICAL USAGE

```
stopServer.sh server1
```

HELP

```
stopServer.sh -help
```

The stopServer.sh program connects to the WAS SOAP service *hostname* that you supplied when you installed the WAS software. If stopServer.sh fails with a complaint that it can't access the SOAP service, then it is very likely because your IP address or host name or DNS has changed. If you supplied a name as hostname during installation, then make sure that that name resolves to the right IP address. Make sure that your server is

accessible by using that IP address (like by `telnet 1.2.3.4 5678`, where 1.2.3.4 is the IP address and 5678 is the WAS SOAP port for your instance (see the installation HOWTO for how to determine and set your WAS port numbers).

dumpNameSpace.sh

Dumps all JNDI names.

TYPICAL USAGE

```
dumpNameSpace.sh -port 1234
```

OR

```
dumpNameSpace.sh
```

For default instance running on default bootstrap port.

Where 1234 is your "BOOTSTRAP_PORT" (defaults to 2809). See the file `$WAS_HOME/bin/wsinstance/<NODE>_<INST>_portdef.props` if you are running an alternate instance and don't know your bootstrap port.

HELP `dumpNameSpace.sh -help`

Don't expect `dumpNameSpace` to show you any reference names, i.e., what is coded in your `.java`. Only the JNDI names themselves are dumped.

ivt.sh

Verify server health. **This command works only for the default instance.** Even with the default instance, it will only work if you have the `ivtApp.ear` application installed and running.

TYPICAL USAGE

```
ivt.sh
```

HELP None

ejbdeploy.sh

Builds all skeleton and stub classes needed for `ejb-jar` file(s).

TYPICAL USAGE

```
ejbdeploy.sh original.jar /tmp/z new.jar -cp a/lib.jar -quiet
```

OR

```
ejbdeploy.sh original.ear /tmp/z new.ear -cp a/lib.jar -quiet
```

HELP `ejbdeploy.sh -help`

This program doesn't *deploy* anything. For each `ejb-jar` file, generates skeleton and stub class and inserts them into a *copy* of the `ejb-jar`. The generated skeleton and stub class files are of the form `EJS*.class`, `_EJS*.class`, `*_Stub*.class`, `*Tie.class`. (The `.java` files for these classes are also created in the work area).

wsadmin.sh

wsadmin.sh gives you a JACL shell from which you can perform nearly any WAS administration interactively or with scripts. This is an advanced tool. See the separate section on wsadmin below.

ws_ant

TYPICAL USAGE

`ws_ant`

HELP

`ws_ant -help`

This gives help for the underlying ant, not for ws_ant.

This is a wrapper for whatever ant library lives in `$WAS_HOME/lib` (the name of the jar file doesn't matter). It makes available to you the ant targets documented at <http://publib7b.boulder.ibm.com/wasinfo1/en/info/aes/javadoc/ae/com/ibm/websphere/ant/targets.html>. Be aware that, for the most part, these targets suck. These targets are ant interfaces for wsadmin, but unlike wsadmin, these targets are not powerful at all because, in most cases, you can not change settings.

Most of the tasks will not work with alternative configuration instances. The classpath mechanism does not work for `WsEjbDeploy`, making it worthless.

The validation task is useful for ear files, but usually not for ejb jar files, since the mechanism for setting a class path does not work. This target also suffers from the same failing as that of validation by `ejbdeploy.sh` and `assembly.sh`— often it complains of an internal error for no known reason.

`ws_ant` sets `$WAS_HOME` if it is not already set. So, if you are using the default instance (i.e., there is no `$USER_INSTALL_ROOT`), you can invoke `ws_ant` like `/path/to/ws-home/bin/ws_ant` without setting up your environment at all.

To use our Nextel ant build file, see my *WAS Ant Builds at Nextel* HOWTO.

If you have not upgraded Ant to 1.5, then definitely upgrade it (it has many important features)

To upgrade the version of ant used by WAS, get rid of the `ant.jar` in `$WAS_HOME/lib` (WAS) or `$WSAD_SOFTWARE_ROOT/runtimes/base_v5/lib` (WSAD), and put the following files into that directory. Make sure that you either remove `ant.jar`, or move it out of that directory (renaming it in that directory won't work because it will continue to be used).

1. `ant-153.jar`, base jar file for Ant v. 1.5.3.
2. `optional-153.jar`, optional jar file for Ant v. 1.5.3.
3. `ant-contrib-0.3.jar`, v. 0.3 of the Ant contributed jar.

You can, of course, download these products direct from the authors at <http://ant.apache.org/bindownload.cgi> and http://sourceforge.net/project/showfiles.php?group_id=36177.

ps and kill

These are UNIX commands, not WAS commands. Like it or not, sometimes WebSphere gets *stuck*. In those cases, ps and kill can get you unstuck.

```
ps -eo pid,ppid,args | grep java      # Solaris
OR
ps axo pid,ppid,args | grep java      # Linux
# Look for a process like ".../java -Xbootclasspath..." with ppid of 1.
kill 1234      # Where 1234 is the desired pid
kill -KILL 1234      # If the default signal (INT) fails to stop it.
```

Repeat until all the target java processes are gone. Do give it a little while to quit after a signal. If you are running multiple instances, then supply different args to ps in order to differentiate the desired instance's processes.

You should run the ps command right now so that you know what the processes look like when things are healthy. (Don't use the kill command unless stopServer.sh won't work).

Graphical

The WAS graphical programs which every WAS Administrator should know.

Admin Console

This is *the* Administrative Gui for WAS 5.0. If you are running the default instance of a default setup of WAS on your local computer, you can access the Admin Console by using a browser to go to `http://localhost:9090/admin`.

In general, go to `/admin` on the host name, with the Administrative HTTP port.

[Edit](#) [View](#) [Go](#) [Bookmarks](#) [Tools](#) [Window](#) [Help](#)

Forward Reload Stop

[http://localhost:1302/admin/secure/logon.di](#) Search Print

WebSphere Administrative Console | WebSphere Application Server 5.0 Ac...

WebSphere Application Server Administrative Console
 Version 5

[Save](#) | [Preferences](#) | [Logout](#) | [Help](#)

[Enterprise Applications](#)
[Install New Application](#)

Enterprise Applications

A list of installed applications. A single application can be deployed onto multiple nodes.

Total: 6
 Filter
 Preferences

<input type="checkbox"/>	Name	Status
<input type="checkbox"/>	IVT Application	➔
<input type="checkbox"/>	LeadGen	➔
<input type="checkbox"/>	adminconsole	➔
<input type="checkbox"/>	passer	➔
<input type="checkbox"/>	voker1a	➔
<input type="checkbox"/>	voker1b	➔

WebSphere Status [< Previous](#) [Next >](#) March 28, 2003 11:58:40 AM

WebSphere Configuration Problems

Total Workspace Files 0 Total Configuration Problems 0

Normally, you can use any username to gain access. If global security is turned on, you may need to enter a valid local OS account username and password, or a username and password from some LDAP database.

Most of the data displayed is cached. If you want to see changes made by other Administrators using Admin Consoles or wsadmin, you have to use the Logoff menu item then log back in again.

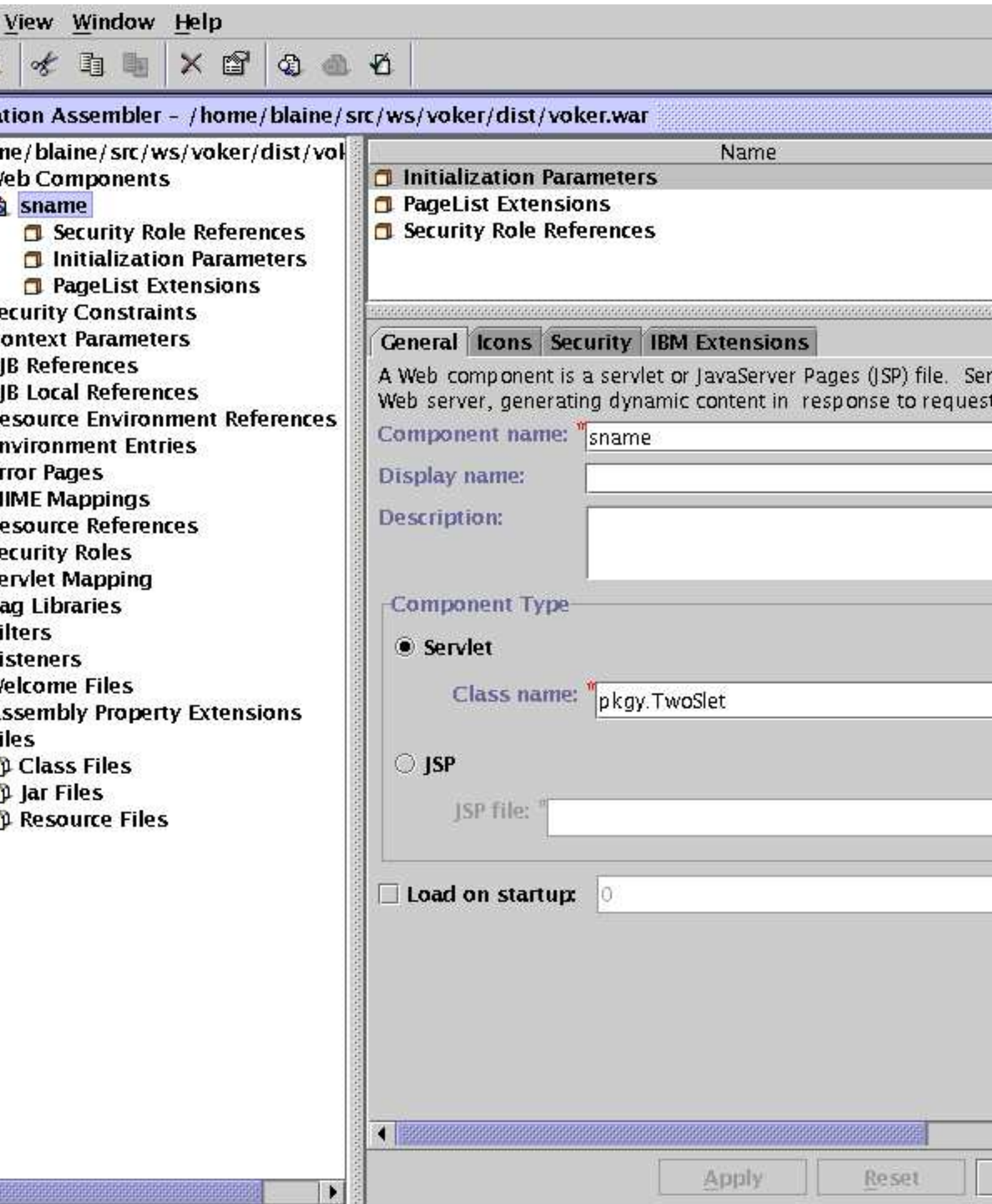
See the Deployment section below.

assembly.sh

IBM likes to call this *AAT* (stands for Application Assembly Tool). Run it like

```
assembly.sh > /dev/null 2>&1 &
```

since the program isn't smart enough to go into the background by itself. Notice that the stderr and stdout output are thrown away. Unlike IWS's `deploytool`, I don't know of any problems with `assembly.sh` where stderr output is needed for troubleshooting.



The validation feature is moderately useful. It misses many obvious errors which will get caught during deployment. It constantly reports an internal error and says to see the *log file* (there is no log file).

It is usually easier to use the JTree in the left pane to add/remove items (usually by right-clicking and picking "Add something" or "New").

When asked for "Module Visibility", choose "COMPATIBILITY".

Deployment

First of all, be aware that when IBM says *EJB deployment*, they usually mean the generation of EJB stubs and skeletons, not the *deployment* of anything. This is accomplished with the `ejbdeploy.sh` program. This section of this document is about *deployment* in the normal J2EE sense—deploying something to an application server, not IBM's idiosyncratic use of the word.

Note that in this section I use `$USER_INSTALL_ROOT` to indicate the home of your WAS instance, whether it is an alternate instance or not. If you are running the default instance, the variable used is actually `$WAS_HOME`.

Note that what most of the J2EE world calls *redployment*, IBM calls *update*. I.e.

`update = redeploy`

What happens during deployment?

1. You supply an ear file and an *application name* to `wsadmin.sh` or to the Admin Console. (For an *update*, the app name must match an existing application name).
2. Your ear file is validated and written under `$USER_INSTALL_ROOT/wstemp`
3. When you *save* applications to WAS ("Save to global..." with WAS Admin, or "\$AdminConfig save" with `wsadmin.sh`)
 1. J2EE ears and components get copied from `$USER_INSTALL_ROOT/wstemp` to `$USER_INSTALL_ROOT/installedApps`
 2. Deployment descriptors get written to `$USER_INSTALL_ROOT/config/cells/<CELL>/application`
 3. The server index for the target instances (nodes) get updated at `$USER_INSTALL_ROOT/config/cells/<CELL>/nodes/<NODE>_<INST>/serverindex.xml` ■
4. If you indicated to start the application during deployment, it will still not be started until WAS is restarted (you can, however start it at this time with the Admin Console). (If you indicated to start an application during an *update*, then it will be started when you *save*).

For "application name", do **not** include a file extension, like ".ear", because WAS deploys applications in a directory like "appname.ear". (So, if you name your app "x.ear", it would be deployed to "x.ear.ear".) Therefore, do not just accept Admin Console's *default* application name, which is just the ear file name.

Do not use spaces or special characters in your application name. Since we host on UNIX, these spaces cause difficulties for shell scripts (it isn't difficult to work around these difficulties, but most shell script writers do not know how to do it).

If you add any application that contains a context root change, or which contains a new .war, you will generally have to *Update Web Server Plugin*. This will add the new context roots to the `plugin-cfg.xml` file. Unless this context root is covered by an existing mapping in the file, your web server will not forward requests for this context to the app server. If you are running multiple instances, be aware that you are only updating the `plugin-cfg.xml` in your `$USER_INSTALL_ROOT`, and that `plugin-cfg.xml` file is only used if it is referenced by a web server config file.

Front end web servers do not need to be restarted after deploying new applications, nor after regenerating the `plugin-cfg.xml` file.

There is a deployment script in the `wsadmin` section of this document.

wsadmin

TYPICAL USAGE

`wsadmin.sh`

HELP `wsadmin.sh.sh -help`

`wsadmin.sh` is a JACL shell with access to the WAS Mbeans and four WAS-specific Java extension administration objects. JACL is tcl with Java extensions... it gives you access to both Java language functionalities and to Java objects.

Can run without connecting to server, like: `wsadmin.sh -conntype NONE`

Limitations

I know of no way to use `wsadmin` to "start" an existing application that is currently not running. If you update an app that is currently running and don't save until you are finished, the app will continue to run (there is an example of this below). For an app with a new *name*, if you want it to *run*, you will have to use an Admin Console to do that (see next item about visibility), or restart WAS.

Like the Administrative Console, `wsadmin` does not see changes made by outside means after you enter a shell.

You can not start a (normal standalone) WAS instance with `wsadmin`.

WAS JACL Administration Objects

The four WAS administration objects provided are aliased to variables.

\$Help

\$AdminControl

Stuff about the running server

\$AdminApp

\$AdminConfig

For configuration of the installation

Useful WAS Administration Commands.

These are all sub-commands of the four Administration objects listed above.

\$Help help

\$Help <MBeanHelpCommand> MBean

Help about something about given MBean.

\$Help AdminControl

lists a good description of the AdminControl object, plus a summary of the AdminControl sub-commands.

\$AdminControl help <AdminCtlCommand>, etc.

for the other main commands. (Many commands have *_jmx counterparts for lower-level use)

\$AdminControl getMBeanCount
(returns 55 on my server)

\$AdminControl stopServer server1
(can't use startServer without a NodeAgent, even with -conntype NONE).

\$AdminConfig types

\$AdminConfig list JDBCProvider

\$AdminConfig list Deployment

\$AdminApp list

\$AdminApp list Modules

\$AdminApp uninstall {AppName}

\$AdminApp taskInfo /tmp/webonly.ear MapWebModToVH

\$AdminApp options

Lists all available options, incl. for "install"

\$AdminConfig save

(Must do this to perform the uninstall) (Damned this still appears in Admin Console as a non-running App. Everything shows up right after a WAS restart.)
N.b. Uninstall doesn't work for me sometimes unless I stop and start IWS after the install.

\$AdminApp installInteractive src/ws/man/man.ear

(then examine wsadmin.traceout for lines containing WASX7278I) (See my script <http://admc.com/blaine/howtos/nextel-util.jacl> for installing or updating apps. Note that the arg to MapWebModToGH consists of triplets of the form [list \$webmodule_dispname \$warfile,WEB-INF/web.xml default_host]], where default_host is the default virtual host).

Comparison to Administrative Console

The following sequence is equal to Console Update + Apply

```
$AdminApp uninstall {AppName}
```

```
$AdminApp install...
```

```
$AdminConfig save
```

```
OR
```

```
$AdminConfig reset
```

```
The app continues running!
```

Deployment script

If you download <http://admc.com/blaine/howtos/nextel-util.jacl> and run the command

```
source /app/iws/was/scripts/nextel-util.jacl
```

in wsadmin, then that jacl shell will have access to the procedures installApp and updateApp.

installApp ApplicationName path/to/file.ear nodeName serverName

Installs the given ear with the given name, using bindings specified in the deployment descriptors in the ear file. Application will start the next time WAS is started.

installApp ApplicationName path/to/file.ear nodeName serverName prefix

The same, but prefixes "prefix/" to all EJB jndi names. Application will start the next time WAS is started.

updateApp ApplicationName path/to/file.ear nodeName serverName

Same as the corresponding bInstallApp procedure. The given ear file will *replace* the existing application with the given name. Application will be started if the existing application was running.

updateApp ApplicationName path/to/file.ear nodeName serverName prefix

Installs the given ear with the given name, using bindings specified in Application will be started if the existing application was running.

The installApp and updateApp procedures can be used from ant. See the -deploy target in the Ant build file <http://admc.com/blaine/howtos/nextelapp-build.xml>.

If you have not upgraded Ant to 1.5, then see the instructions on that procedure in the installation HOWTO.

Data Sources

See the installation HOWTO for instructions and explanation about creating Oracle data sources. The instructions here are generic.

1. First define the JDBC Provider (i.e., the JDBC driver).
2. Then restart WAS.
3. Then define 5.0 datasource underneath the JDBC Provider.
4. JNDI Name: Unlike IAS, always start them with jdbc/.
5. If new, leave Authent. Alias. blank and just Apply. (You have to come back and set it later once it's in the drop-down list).
6. Authent. Alias.

If your component has Res. Auth. of application, then use "Component-managed"
If Res. Auth. of Container, then use "Container-managed".

7. Modifying Authent. Aliases. Click "J2C Authentication Data Entries" on the specific data source screen.
8. Use button up top to save to global configuration.
9. Definitely have to restart WAS after some DS changes (like changing the JNDI name and changing the URL).