CHAPTER 4: ADMINISTRATION VIA SCRIPTS

Theory

WebSphere Application Server provides several command line tools for you to stop, start, and check status of application server processes or nodes. Those command line tools can run only local servers and nodes. They are located under 'bin' directory of application server or deployment manager profiles.

While using command line tools, you have to keep in mind that all names of application servers, nodes, or cells are case sensitive. If you fail to supply proper name, scripts will fail.

You should be also careful with running those scripts with proper rights. If you have installed Websphere as administrator on Microsoft operating systems, you have to run some of the commands as administrator.

You cannot run command line tools on a remote server. In order to manage servers remotely, you can use "wsadmin" scripting that connects to deployment manager using SOAP port.

"wsadmin" scripting also allows you to automate repeating tasks more easily and consistently. It gives you a full range of administrative activities.

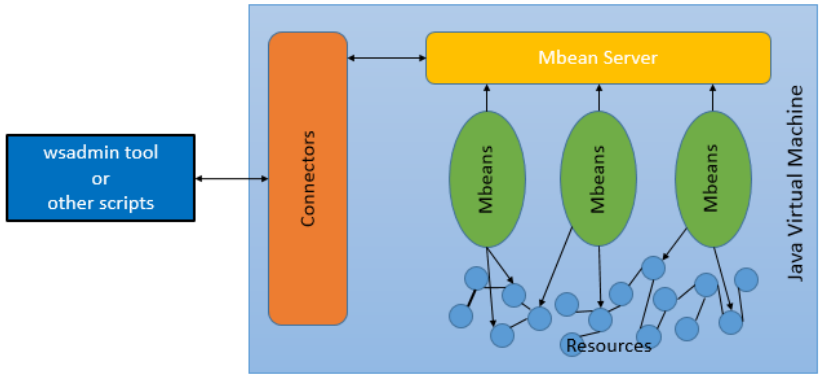
The wsadmin tool has 5 objects available:

- AdminApp contains set of commands to manage installation, removal and editing of the applications.
- AdminConfig contains set of commands to create, remove and modify the elements of WebSphere Application Server.
- AdminControl contains commands that are used for operational control of the objects within the configurations such as start or stop an application server.
- AdminTask has a set of commands that are used for more administration specific tasks such as creating an application server or creating a cluster. Most of the things you can do with AdminTask commands can be done by using other objects, but they will take more lines to perform same thing.

- Help object provides help for each command and object or for the methods, attributes and notifications of MBeans.

The wsadmin tool can work both on local and remote systems. On local mode, you can even work when the application server is down where on remote mode, you have to have the application server up and running. On the other hand, local mode can have issues due to multiple access triggered synchronization of changes.

The wsadmin tool supports Jacl and Jython scripting languages. Although Jacl is the default language for wsadmin, you can change it by giving "-lang jython" parameter or editing the "wsadmin.properties" file under the properties folder of the given profile.



Jacl is a Tcl implementation written in Java and works on Java Virtual Machine. It can also enable communication between Java and Tcl interpreters and that allows you to use scripting functionality in to an existing Java applications.

Jython is a Pyhton programming language implementation written in Java. Jython uses both Pyhton modules and Java classes. You can also import and use any Java class. Jython compiles Python source code to Java bytecodes either on demand or statically.

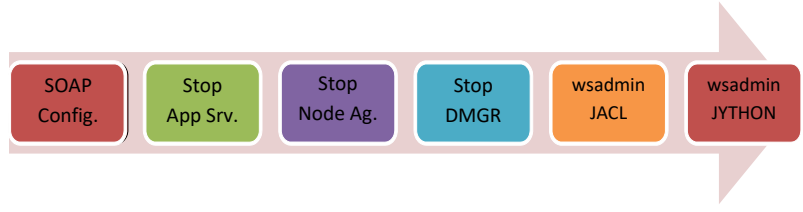
AIM

The aim of the lab exercise is to perform fundamental administrative operations such as stop, start an application server, a node agent and a deployment manager. You will be able to perform those using built-in command line tools and also with Jacl and Jython scripts.

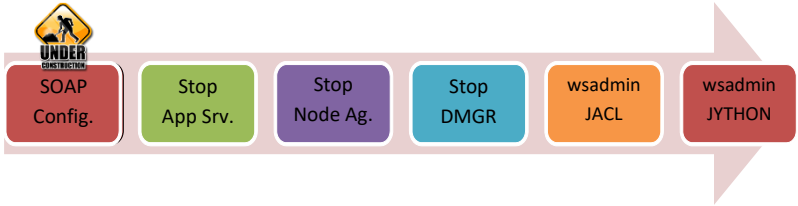
In order to achieve this goal, you will need to perform following tasks:

- SOAP Configuration
- Stop & Start Application Server
- Stop & Start Node Agent
- Stop & Start Deployment Manager
- wsadmin scripting with Jacl
- wsadmin scripting with Jython

Lab Exercise 4: ADMINISTRATION VIA SCRIPTS



1. **SOAP Configuration**
2. **Stop & Start Application Server**
3. **Stop & Start Node Agent**
4. **Stop & Start Deployment Manager**
5. **wsadmin scripting with Jacl**
6. **wsadmin scripting with Jython**

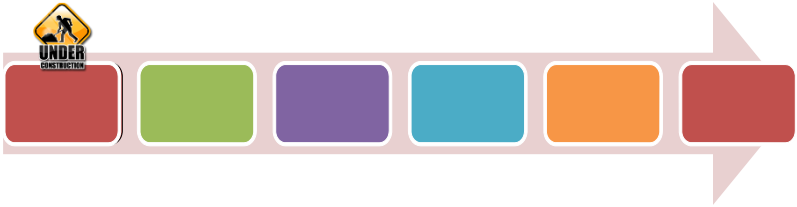


Task 1: SOAP Configuration

This task is only required if you have enabled “Global security” in the previous chapter. The username used below is the one assigned as administrative user defined in the 2nd Lab Exercise, in task 2, “Secure Administration Console”.

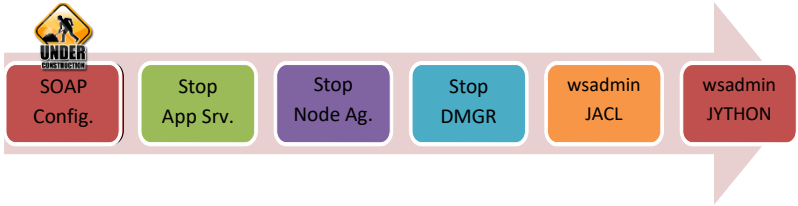
This task will save use to type each time the username and password we used administrative commands. We will store the administrative username and password in “soap.client.props” file for each profile and then we will encrypt the password to mitigate security considerations.

Step 1: Change directory to deployment manager profile properties folder (Install_path/IBM/WebSphere/AppServer/profiles/Dmgr01/properties) and open the file “**soap.client.props**” to edit.



Step 2: Update the file with administrative user credentials into the fields “com.ibm.SOAP.loginUserid” and “com.ibm.SOAP.loginPassword”, then save the file and exit.

```
#  The PropFilePasswordEncoder utility may be used to encode passwords in a
#  properties file. To edit an encoded password, replace the whole password
#  string (including the encoding tag {...}) with the new password and then
#  encode the password with the PropFilePasswordEncoder utility. Refer to
#  product documentation for additional information.
#
###############################################################################

#-------------------------------------------------------------------------------
# SOAP Client Security Enablement
# - security enabled status  ( false[default], true  )
#-------------------------------------------------------------------------------
com.ibm.SOAP.securityEnabled=false

#-------------------------------------------------------------------------------
# - authenticationTarget     ( BasicAuth[default], KRB5. These are the only supported selection
#                              on a pure client for JMX SOAP Connector Client. )
#-------------------------------------------------------------------------------
com.ibm.SOAP.authenticationTarget=BasicAuth

com.ibm.SOAP.loginUserid=wasadmin
com.ibm.SOAP.loginPassword=wasadmin

#-------------------------------------------------------------------------------
# SOAP Login Prompt
```

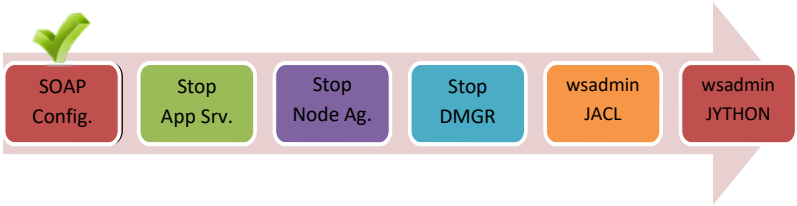


Step 3: Encode the "**soap.client.props**" file by using the following command. Change directory to the bin directory of the profile and run *"PropFilePasswordEncoder.sh /opt/IBM/WebSphere/AppServer/profiles/Dmgr01/properties/soap.client.props com.ibm.SOAP.loginPassword"*

```
        wve_encodePassword.sh
executeXPath.sh             osgiConsole.sh              unaugmentProxyServer.sh
        xdaSetupCmdLine.sh
executeXQuery.sh            pluginCfgMerge.sh           unlinkCells.sh
        xd_hadmgrAdd.sh
executeXSLT.sh              pluginMerge.sh              uteconfig.sh
        xd_hadmgrRemove.sh
Extractor.sh                pmt.sh                      versionInfo.sh
        xjc.sh
findEJBTimers.sh            postinstall.sh              VEUpgrade.sh
[root@wasv90 bin]# ./PropFilePasswordEncoder.sh "/opt/IBM/WebSphere/AppServer/pr
ofiles/Dmgr01/properties/soap.client.props" com.ibm.SOAP.loginPassword
```

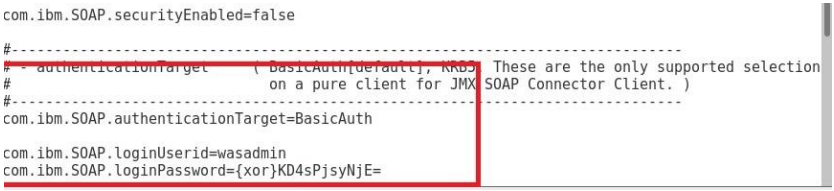
Step 4: Check the "**soap.client.props**" if the password field is encrypted.

```
root@wasv90:/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/properties
File  Edit  View  Search  Terminal  Help
[root@wasv90 dmgr01]# cd properties/
[root@wasv90 properties]# cat soap.client.props
################################################################################
#
#                          JMX SOAP Connector Client Properties File
#
#  This file contains properties that are used by the JMX SOAP Connector Client
#  of the WebSphere Application Server product. SOAP Connector executes on WebSphere
#  java servers and client systems with java applications that access WebSphere servers.
#
#  ** Encoding Passwords in this File **
#
#  The PropFilePasswordEncoder utility may be used to encode passwords in a
#  properties file. To edit an encoded password, replace the whole password
#  string (including the encoding tag {...}) with the new password and then
#  encode the password with the PropFilePasswordEncoder utility. Refer to
#  product documentation for additional information.
#
################################################################################

#--------------------------------------------------------------------------------
# SOAP Client Security Enablement
# - security enabled status  ( false[default], true  )
#--------------------------------------------------------------------------------
com.ibm.SOAP.securityEnabled=false

#--------------------------------------------------------------------------------
# - authenticationTarget     ( BasicAuth[default], KRB5 - These are the only supported selection
#                              on a pure client for JMX SOAP Connector Client. )
#--------------------------------------------------------------------------------
com.ibm.SOAP.authenticationTarget=BasicAuth

com.ibm.SOAP.loginUserid=wasadmin
com.ibm.SOAP.loginPassword={xor}KD4sPjsyNjE=
```

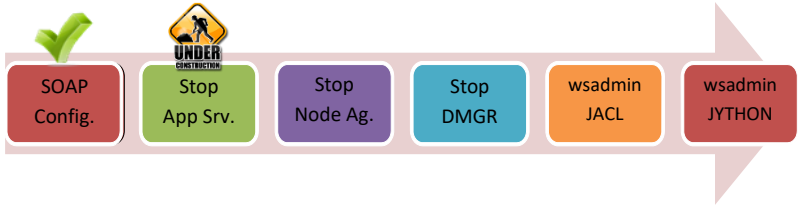


Step 5: Repeat the same steps for the “AppSrv01” profile.

Step 6: Make sure that the password is encrypted.

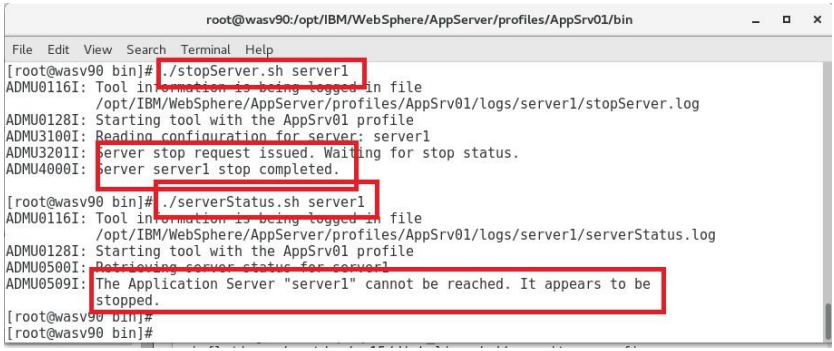


Task 1 is complete!

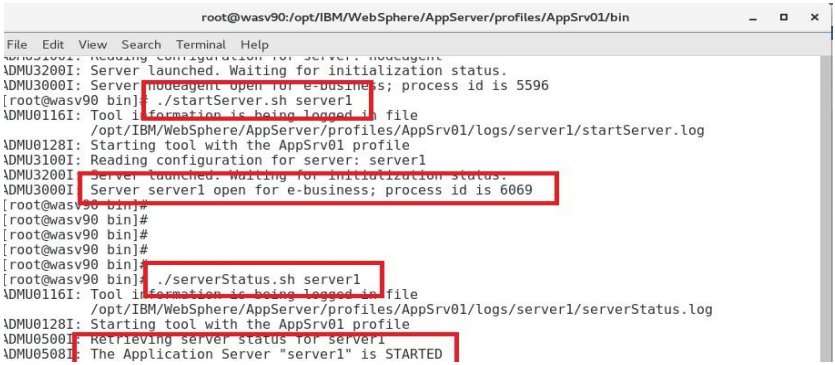


Task 2: Stop & Start Application Server

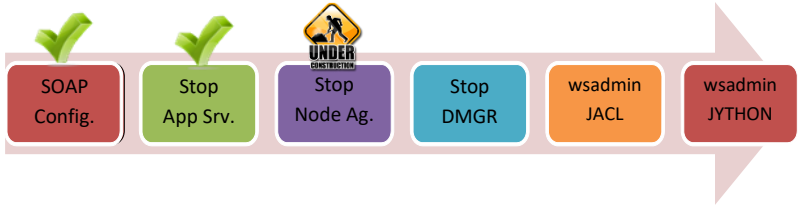
Step 1: To stop the application server, change directory to the "bin" directory of the profile that server runs and issue command "**stopServer.bat server_name**".



Step 2: For starting, run "**startServer.bat server_name**". You can check the status of the server via "**serverStatus.bat server_name**".

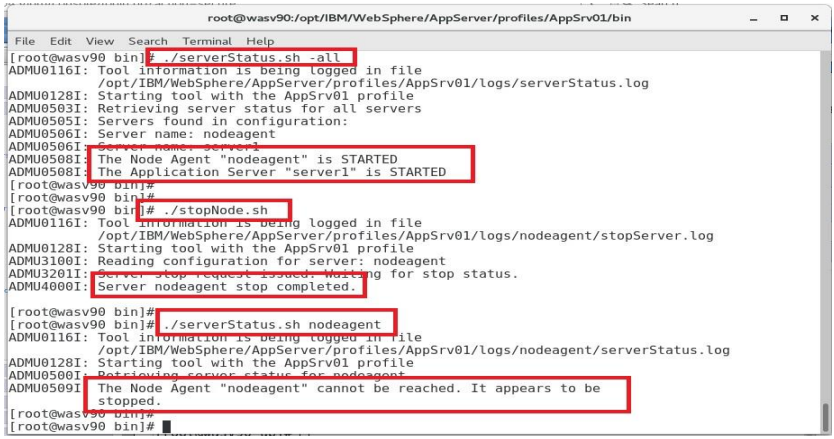


Task 2 is complete!

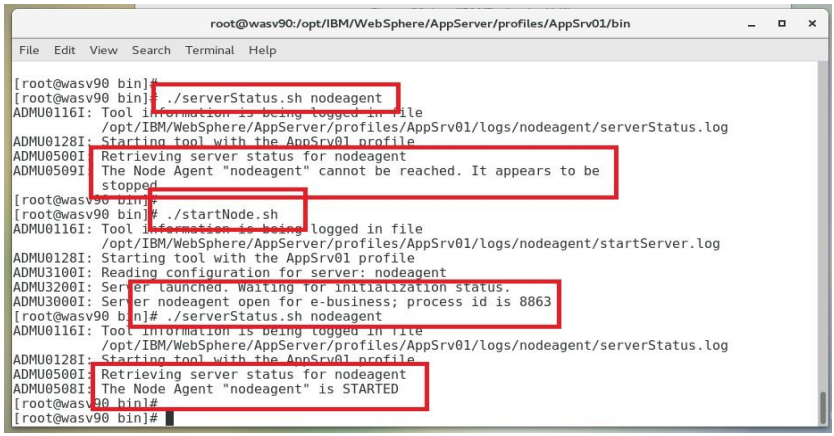


Task 3: Stop & Start Node Agent

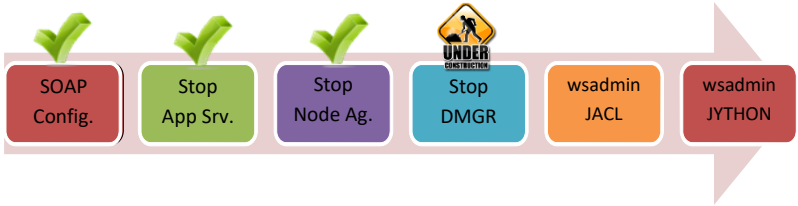
Step 1: In order to stop the node agent, please issue the command "**stopNode.bat**".



Step 2: Run "**startNode.bat**" to start the node agent.

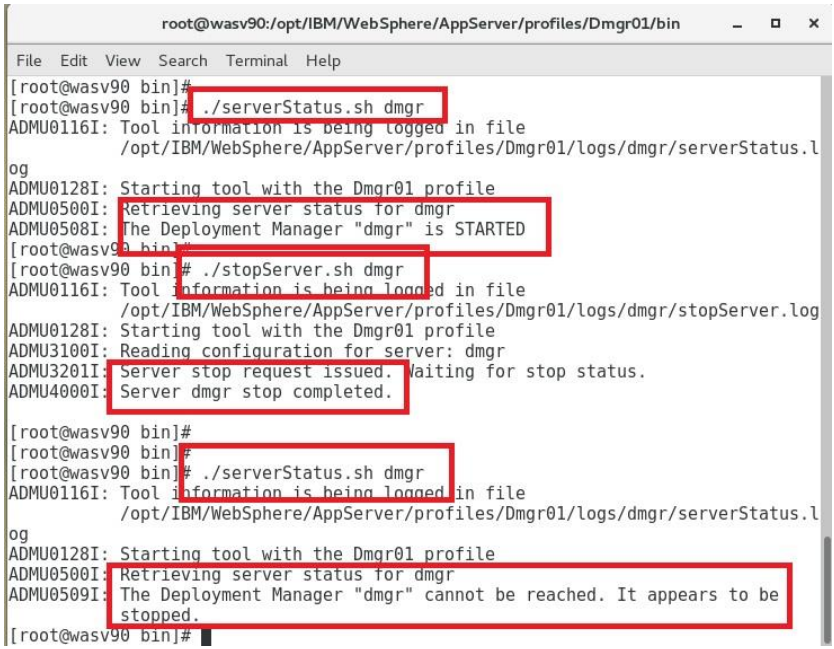


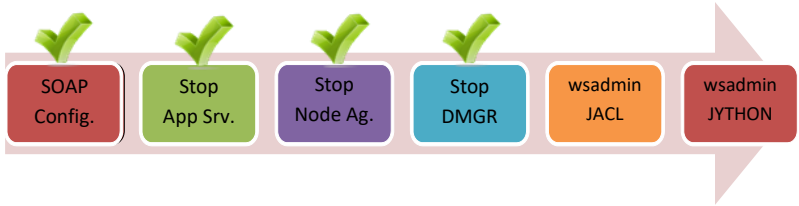
Task 3 is complete!



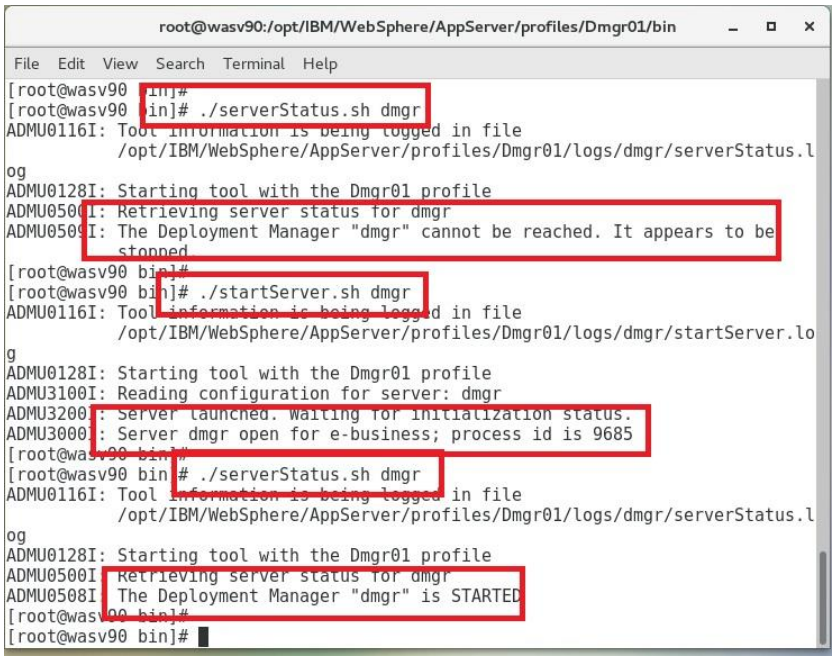
Task 4: Stop & Start Deployment Manager

Step 1: Change directory to the "bin" directory of the deployment manager and run "**stopManager.bat**" to stop.

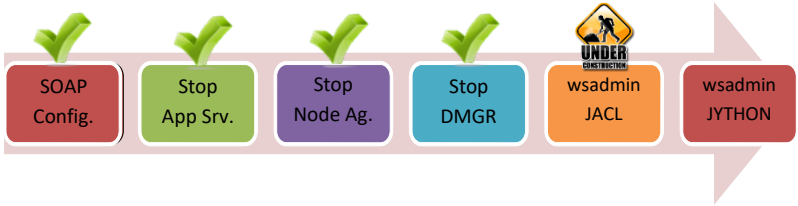




Step 2: Run "startManager.sh" to start deployment manager. You can always run "serverStatus.bat dmgr" to check the status of deployment manager.



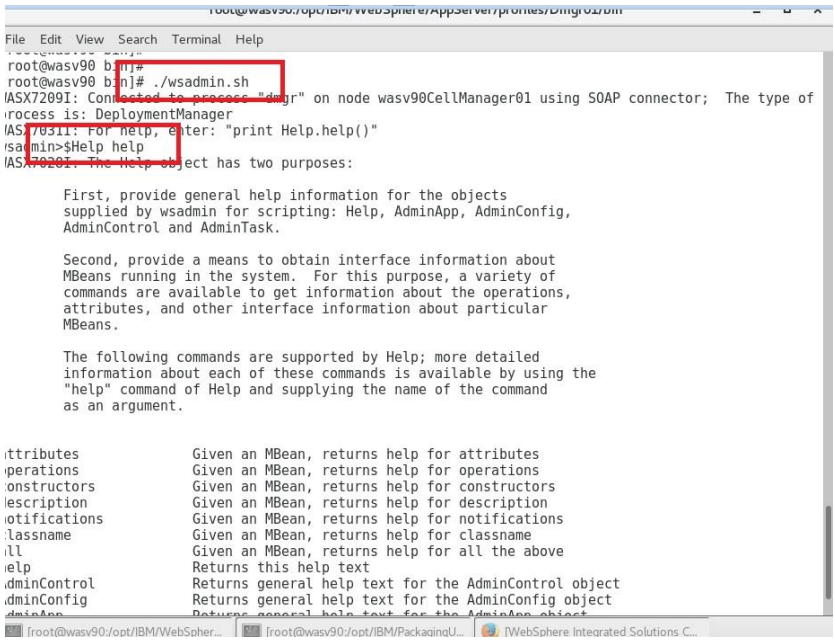
Task 4 is complete!

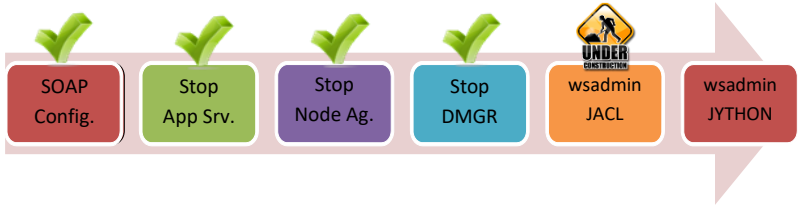


Task 6: "wsadmin" Scripting with Jacl

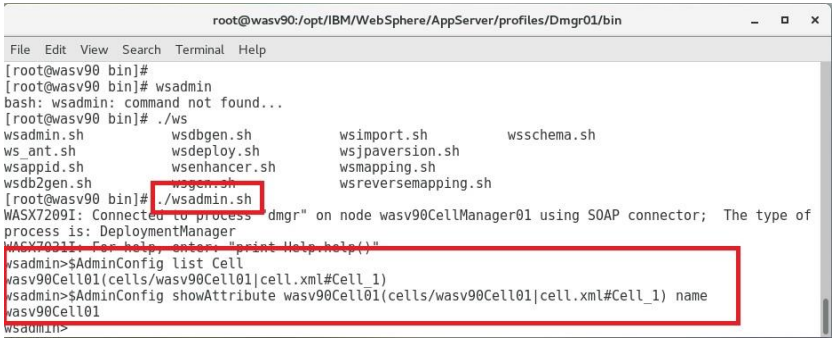
Step 1: Change directory to "/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/bin" run the command "**wsadmin.sh**".

"**$Help help**" command will show you the basic help information.

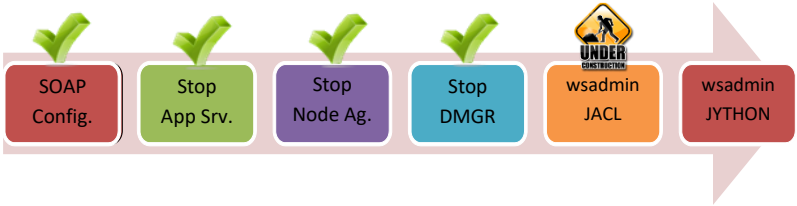




Step 2: Run the command "**$AdminConfig list Cell**" to list all the cells. When you get the list of the cells, run "**$AdminConfig showAttribute *cell_id* name**" where *cell_id* is the output of previous command, to get the name of the cell.



Step 3: Run "**$AdminConfig list Node *cell_id***" to list all the nodes under the cell provided.
"**$AdminConfig showAttribute *node_id* name**" will give you the name of the node. *node_id* can be gathered from the previous command.



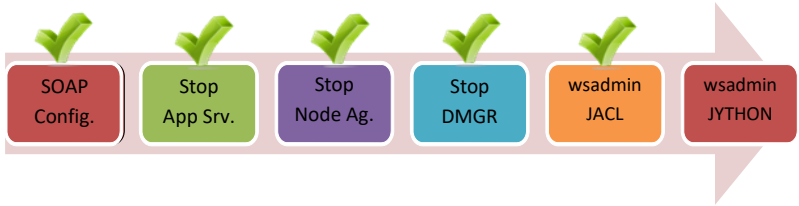
Step 4: In order to stop an application server, please issue the following command:

"**$AdminControl stopServer *server_name node_name***"

```
wsadmin>
wsadmin>
wsadmin>$AdminControl stopServer server1 wasv90Node01
WASX7264I: Stop completed for server "server1" on node "wasv90Node01"
wsadmin>
wsadmin>
wsadmin>
```

Step 5: Run "**$AdminControl startServer *server_name node_name***" to start an application server.

```
wsadmin>
wsadmin>
wsadmin>$AdminControl startServer server1 wasv90Node01
WASX7262I: Start completed for server "server1" on node "wasv90Node01"
wsadmin>
wsadmin>
```



Step 6: In order to stop a node agent, please issue following 2 commands in "wsadmin" environment:

"set node_agent_name [$AdminControl queryNames type=NodeAgent,node=*node_name*,*]"

"$AdminControl invoke $node_agent_name stopNode"

```
wsadmin>
wsadmin>
wsadmin>
wsadmin>set node_agent_name [$AdminControl queryNames type=NodeAgent,node=wasv90Node01,*]
WebSphere:name=NodeAgent,process=nodeagent,platform=common,node=wasv90Node01,diagnosticProvider=true,v
ersion=9.0.0.0,type=NodeAgent,mbeanIdentifier=NodeAgent,cell=wasv90Cell01,spec=1.0
wsadmin>$AdminControl invoke $node_agent_name stopNode

wsadmin>
wsadmin>_
```

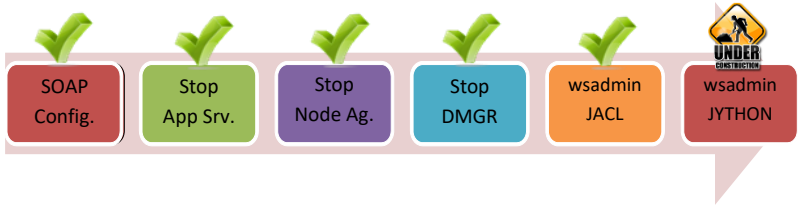
Step 7: Run following commands to check the status of an application server.

"set server_name [$AdminControl completeObjectName cell=*cell_name*, node=*node_name*, name=*server_name*, type=Server,*]"

"$AdminControl getAttribute $server_name state"

```
wsadmin>
wsadmin>
wsadmin>set server_name [$AdminControl completeObjectName cell=wasv90Cell01,node=wasv90Node01,name=server1,type=Server,*]
WebSphere:name=server1,process=server1,platform=proxy,node=wasv90Node01,j2eeType=J2EEServer,version=9.0.0.0,type=Server,mbeanIdent
ifier=cells/wasv90Cell01/nodes/wasv90Node01/servers/server1/server.xml#Server_1525949690683,cell=wasv90Cell01,spec=1.0,processType
=ManagedProcess
wsadmin>$AdminControl getAttribute $server_name state
STARTED
wsadmin>
```

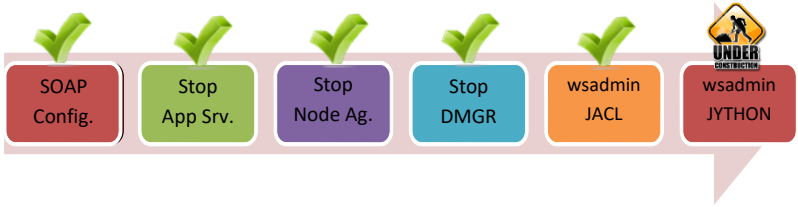
Task 5 is complete!



Task 6: "wsadmin" Scripting with Jython

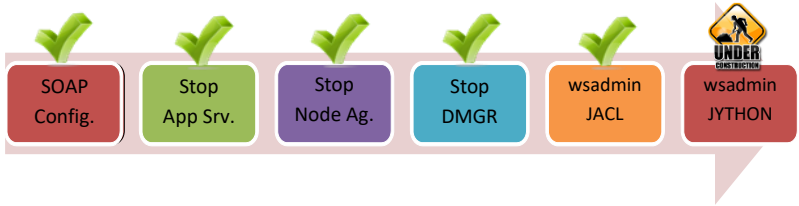
Step 1: Download "wsadminlib.py.zip" from http://public.dhe.ibm.com/software/dw/wes/samplescripts/wsadminlib.py.zip and "unzip wsadminlib.py.zip" to /opt/IBM/WebSphere/AppSrv/bin and then run "wsadmin".

```
[root@wasv90 bin]#
[root@wasv90 bin]#
[root@wasv90 bin]#
[root@wasv90 bin]# ./wsadmin.sh
WASX7209I: Connected to process "dmgr" on node wasv90CellManager01 using SOAP connector;  The type of
process is: DeploymentManager
WASX7031I: For help, enter: "print Help.help()"
wsadmin>
wsadmin>
wsadmin>
```



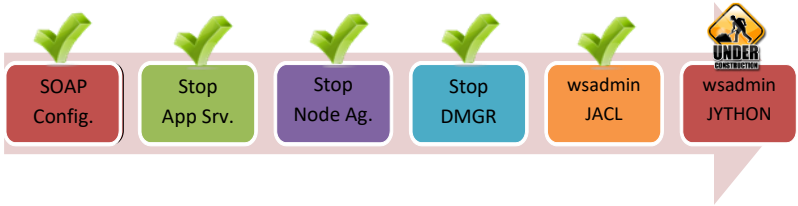
Step 2: Issue command "**execfile('wsadminlib.py')**" to load the file.

Step 3: Issue command "whatEnv()" to see your installation whether stand-alone or Network Deployment.



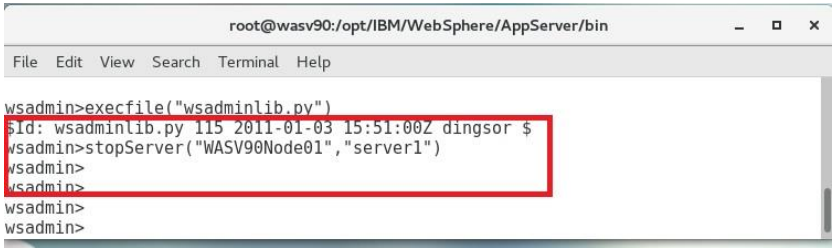
Step 4: "getCellName()" will give you the name of your cell.

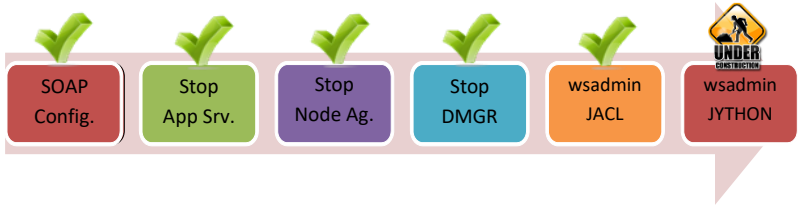
Step 5: "listNodes()" will list all the nodes except for deployment manager node and to see that run "getDmgrNodeName()" command.



Step 6: "listAllServers()" command will list all application servers.

Step 7: To stop an application server, "stopServer("*node_name","server_name*")".

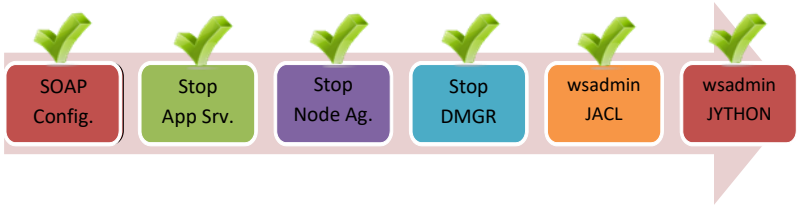




Step 8: "startServer('*node_name*','*server_name*')" will start an application server.

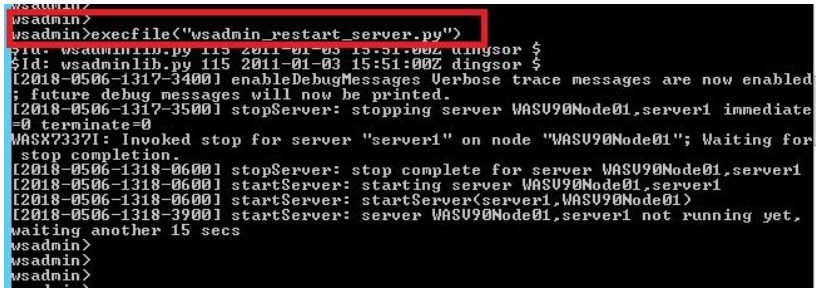
Step 9: It is also possible to create our own python scripts using the existing ones. Create a file named "wsadmin_restart_server.py" and add following lines to the file.

```
execfile('./wsadminlib.py')
enableDebugMessages()

servername = 'server1'  #change with your server name
nodename = 'WASV90Node01' #change with your node name

#Stop Server server1
stopServer(nodename, servername)

#Start Server server1
startServer(nodename,servername)
```



Run the following command to restart the configured application server.

“execfile(“wsadmin_restart_server.py”)”



SUMMARY

WebSphere Applications Server provides you to use several options to perform administration. There are already built in command line tools that gives you the possibility to perform operations such as stop, start, add, remove an application server, node, and cell and so on. It is also possible to use scripting for administration purposes. For that, you can use either Jacl or Jython using ‘wsadmin’ tool. It is both possible to send commands one by one through ‘wsadmin’ tool and feeding files of scripts to ‘wsadmin’ tool. Scripting allows you to manage Websphere remotely.

REFERENCES

- http://en.wikipedia.org/wiki/Wsadmin
- http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html
- http://pic.dhe.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Fcxml_javamanagementx.html

INDEX

Jacl 97
Jython 97
wsadmin 96, 97