Command Line Scripting with wsadmin and Jython
Unit Objectives

• This unit will discuss:
  – wsadmin basics
  – wsadmin preference
  – wsadmin help
  – AdminTask
  – Jython basics
  – Profiles scripts
wsadmin - Introduction

• Provides scripting capabilities

• Provides command line administration

• Common operational and configurational tasks can be performed from a command line instead of through the Administrative Console

• Examples:
  – Start and stop (deployment manager, nodes, application servers, enterprise applications and clusters)
  – Configure virtual hosts, JDBC providers
  – Create application servers
  – Create clusters and add members to a cluster
wsadmin Overview

• Scripting interface for WebSphere Application Server V6

• Support for:
  – Tcl commands through Java Command Language (JACL)
  – Python commands through jython (new as of V5.1)

• Available for all versions and platforms of WebSphere Application Server V6

• Based on Bean Scripting Framework (BSF)
  – Provides scripting and programming model similar to Java
wsadmin Administrative Functions

• WebSphere Application Server system management separates administrative functions into two categories:
  – Configuration of WebSphere Application Server installations (repository)
  – Running objects in WebSphere Application Server installations

Repository

Running Application Server

Application Servers >

server1

An application server is a server which provides services required to run enterprise applications.

<table>
<thead>
<tr>
<th>General Properties</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Id</td>
<td>2984</td>
<td>The native operating system’s process id for this server.</td>
</tr>
<tr>
<td>Cell Name</td>
<td>was5admin</td>
<td>The name of the cell in which this server is running.</td>
</tr>
<tr>
<td>Node Name</td>
<td>was5admin</td>
<td>The name of the node in which this server is running.</td>
</tr>
<tr>
<td>State</td>
<td>Started</td>
<td>The runtime execution state for this server.</td>
</tr>
</tbody>
</table>

OK
**wsadmin mBeans**

- **wsadmin** acts as an interface to Java objects for access by scripts.
- **wsadmin** uses the same interface (through JMX) as the Administrative Console to make configuration changes and control servers.
- There are five Java objects that perform different operations:
  - **AdminConfig** - create/change WebSphere Application Server "static" configuration.
  - **AdminApp** - install, modify or administer applications.
  - **AdminControl** - work with "live" running objects, perform traces and data type conversion.
  - **AdminTask** - access administrative commands to provide alternate way to access configuration commands.
  - **Help** – provides help.

![Diagram showing the relationship between Script(s), Wsadmin, and Resources through MBeans]
wsadmin: AdminTask

• AdminTask is new in V6
• Used to access set of administrative commands to provide alternate way to access configuration commands
• Runs simple and complex commands
  – $AdminTask commandName {-interactive}
• Grouped based on their function
  – Example: Commands related to security are grouped into a security management command group
• Can be run in batch or interactive mode
• Can be run in connected or local mode
  – If run in local mode not all commands available
Invoking wsadmin

Three ways to invoke wsadmin:

- **Interactively**: `wsadmin`

  ![Command Prompt - wsadmin](image1)

  ```
  C:\Program Files\WebSphere\AppServer\bin>wsadmin
  WASX72091: Connected to process "server1" on node was5host using SOAP connector;
  The type of process is: UnManagedProcess
  WASX70291: For help, enter: "$Help help"
  wsadmin>
  ```

- **Command option**: `wsadmin [-c command]`

  ![Command Prompt](image2)

  ```
  C:\Program Files\WebSphere\AppServer\bin>wsadmin -c "$AdminApp list"
  ```

- **Script file**: `wsadmin [-f scriptfile]`

  ![Command Prompt](image3)

  ```
  C:\Program Files\WebSphere\AppServer\bin>wsadmin -f C:\Program Files\WebSphere\AppServer\sampleinstallation.jacl
  ```
**wsadmin Command Line**

- Use `wsadmin -help` to get command line options

- Use `wsadmin` to start a shell
  - Make sure to run `wsadmin` from the appropriate `<profile_root>in`
  - Brings up a JVM
Help within wsadmin

• In order to get help within wsadmin with jython
  - print Help.help()

• To get help for individual mBeans, use:
  • print Help.AdminConfig()
  • print Help.AdminTask()
  • print Help.AdminControl()
  • print Help.AdminApp()

• For more complete documentation, please see the Information Center
  – The Information Center also includes a number of examples that demonstrates some useful wsadmin functionality (with both jacl and jython)
wsadmin with Jython and Connection Type

- `wsadmin -lang jython`
  - Brings wsadmin up with jython
  - The default language for wsadmin is jacl

- `wsadmin -conntype [SOAP | RMI | NONE ]`
  - The `-conntype` specifies the protocol type
  - Default is SOAP
  - NONE allows wsadmin to read/write directly to configuration files without going through an application server

- `wsadmin -profile <profile_script_name>`
  - Allows wsadmin to preload a profile script
Profile Scripts

Profile scripts can be used to preload wsadmin with predefined settings and functions
– Are run during wsadmin startup
– Called by either:
  • Using the –profile on the command line
  • Defining in wsadmin.properties
    com.ibm.ws.scripting.profiles=
– Could be used to create a standard wsadmin environment for all WebSphere administrators
– Profile scripts can be written in jacl or jython

```
# Print whereAmI
#..............................................................................................
def whereAmI():
    #printing cell and node names
    print "Cell: " + AdminConfig.showAttribute(AdminConfig.list("Cell"), "name")
    print "Node: " + AdminConfig.showAttribute(AdminConfig.list("Node"), "name")
    return

# Start of main
#..............................................................................................
print ""
print "Hello, and welcome to wsadmin using jython"
print ""
print "Running global_profile.py. Global definitions and settings could be added here. It would also be possible to extend wsadmin by defining new customized commands and procedures"
print ""
whereAmI()
print ""
```
wsadmin Properties

• Certain default behaviors for wsadmin can be changed by editing:
  
  `<profile_root>\<profile>\properties\wsadmin.properties`

• Properties include:
  
  - `com.ibm.ws.scripting.connectionType=SOAP`
  - `com.ibm.ws.scripting.port=8880`
  - `com.ibm.ws.scripting.host=localhost`
  - `com.ibm.ws.scripting.defaultLang=jacl`
  - `com.ibm.ws.scripting.traceFile=`
  - `com.ibm.ws.scripting.validationOutput=`
  - `com.ibm.ws.scripting.traceString=com.ibm.*=all=enabled`
  - `com.ibm.ws.scripting.profiles=`
  - `com.ibm.ws.scripting.emitWarningForCustomSecurityPolicy=true`
  - `com.ibm.ws.scripting.tempdir=`
  - `com.ibm.ws.scripting.validationLevel=`
  - `com.ibm.ws.scripting.crossDocumentValidationEnabled=`
  - `com.ibm.ws.scripting.classpath=`
Wsadmin - Important Points to Remember

• Commands are case-sensitive

• `wsadmin -f "scriptfile"` is much faster than `wsadmin -c "command"`
  – Better to run multiple commands in a file than individual commands

• Saving configuration changes is a two step process:
  – First part validates the changes
  – Second part performs the save but will throw an exception if changes conflict (example: creating two servers with the same name)

• Call `AdminConfig.save()` periodically in the script file to persist configuration updates to existing objects
Why use Jython instead of JACL?
- It comes down to a personal choice...
- Each language has its own style and syntax, but they end up being able to do the same things

What is Jython?
- It’s an Object Oriented scripting language
- Its syntax seems more natural to programmers used to Java or C
- Many support libraries to help ease programming
print "Hello, World!"
⇒ Hello, World!

import sys
sys.stdout.write("Hello World!\n")
⇒ Hello World!

• print identifies the standard output stream

• You can use the built-in module by running import statements
  • sys is a built-in module
Jython 101 - Variables

• To assign objects to names, the target of an assignment should be on the left side of an equal sign (=) and the object that you are assigning on the right side.

• The target on the left side can be a name or object component, and the object on the right side can be an arbitrary expression that computes an object. The following rules exist for assigning objects to names:
  • Assignments create object references.
  • Names are created when you assign them.
  • You must assign a name before referencing it.
# examples of numbers: 8, 3.133, 999L, 3+4j
num1 = int(10)
print num1
⇒10

# example of Strings: 'name', "name's", ''
print str(12345)
⇒'12345'
y = [0, 3, 1, 2]
y.append(5)
print y
⇒[0, 3, 1, 2, 5]

y.reverse()
print y
⇒[5, 2, 1, 3, 0]

y.sort()
print y
⇒[0, 1, 2, 3, 5]

print list("apple")
⇒['a', 'p', 'p', 'l', 'e']

print list((1,2,3,4,5))
⇒[1, 2, 3, 4, 5]

test = "This is a test"
test.index("test")
⇒10

test.index('s')
⇒3
```python
# 
# Addition, concatenation and subtraction examples:
#
print 6 + 7
⇒13

text1 = 'Something'
text2 = ' else'
print text1 + text2
⇒Something else

list1 = [0, 1, 2, 3]
list2 = [4, 5, 6, 7]
print list1 + list2
⇒[0, 1, 2, 3, 4, 5, 6, 7]

print 10 - 5
⇒5
```
The basic syntax to define a function is the following:

```python
def name (arg1, arg2, ... ArgN):
    statements
    return value
```

def intersect(seq1, seq2):
    try:
        res = []
        for x in seq1:
            if x in seq2:
                res.append(x)
        return res
    except:
        #
        # Calling the function
        #
        s1 = "SPAM"
s2 = "SCAM"
intersect(s1, s2)
⇒[S, A, M]

intersect([1,2,3], (1.4))
⇒[1]
The basic syntax for an “if” statement:

```python
if test1
    statements1
elif test2
    statements2
else test3
    statements3
```

# example of an if statement
#
weather = 'sunny'
if weather == 'sunny':
    print "Nice weather"
elif weather == 'raining':
    print "Bad weather"
else:
    print "Uncertain, don't plan anything"
The basic syntax for an “while” statement:

```python
while test1
    statements1
else
    statements2
```

```python
# example of a while statement
a = 0; b = 10
while a < b:
    print a
    a = a + 1
```
The basic syntax for an “for” statement:

```python
for target in object:
    statements
else
    statements
```

# example of a for statement
sum = 0
for x in [1, 2, 3, 4]:
    sum = sum + x
print sum
⇒10
Unit Summary

• This unit discussed:
  – wsadmin basics
  – wsadmin preference
  – wsadmin help
  – AdminTask
  – Jython basics
  – Profiles scripts
Exercise 8, 9:
wsadmin and Installing the Quote Web Service
Appendix

• Web Server Support in AdminScripting
Web Server Support in AdminScripting

$AdminTask:
- createWebServer –interactive
- Step 1 input
  - Node
  - Webserver name
  - Template
- Step 2 input
  - Port
  - webserverInstallRoot
  - configurationFile
  - errorLogFile
  - accessLogFile
  - serviceName (Windows only)
  - webserverProtocol (HTTP/HTTPS)
  - adminPort (remote IHS only)
  - adminProtocol
  - adminUserId
  - adminPassword
- deleteServer
- listWebServerTemplates - to display the new webserver templates,

$AdminConfig:
- remove
  - Input is config ID.
- Modify -
  - types - webserver definition.
  - show -
  - showAttributes

$AdminControl
- startServer – invoked thru Mbean using ProcessDefinition (server.xml)
  - StartCmd and StartCmdArgs.
- stopServer - – invoked thru Mbean using ProcessDefinition(server.xml)
  - StartCmd and StartCmdArgs.