



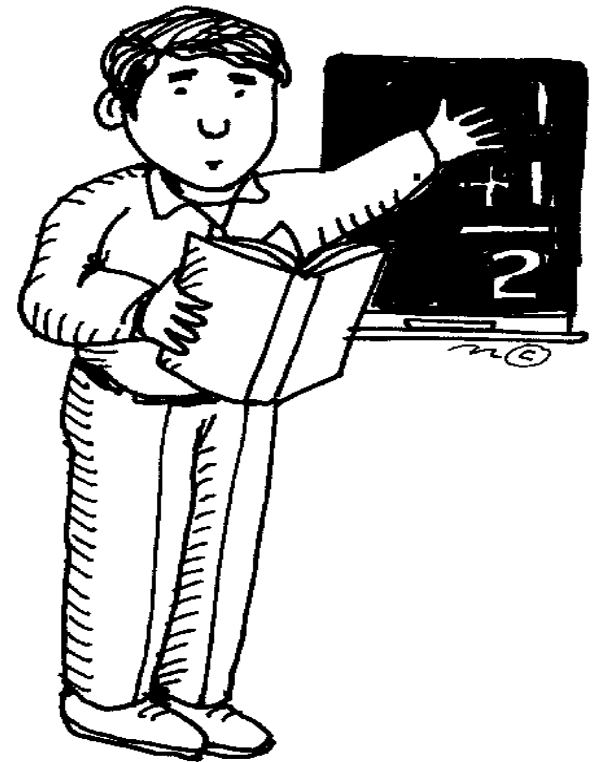
IBM Washington Systems Center

WebSphere Scripting: WSADMIN, jacl, jython

Lee-Win Tai
tai@us.ibm.com

Our Agenda

- **WSADMIN basics**
- **Scripting with WSADMIN**
 - JACL
 - Jython
- **Script examples**



WSADMIN Concepts

WSADMIN is not a keyboard recording mechanism!



WSADMIN is a scripting interface used at the command line.

Windows, Linux, USS

WSADMIN can:

- Install / uninstall applications
- Stop / start servers
- Create new servers

```
root@omega Mon Mar 21 22:03:26 /opt/IBM/WebSphereStudio/ApplicationDevelo
$ ./wsadmin.sh
WASX7209I: Connected to process "server1" on node localhost using SOAP co
WASX7029I: For help, enter: "$Help help"
wsadmin>$Help help
WASX7028I: The Help object has two purposes:

First, provide general help information for the the objects
supplied by wsadmin for scripting: Help, AdminApp, AdminConfig,
and AdminControl.
```

Interactive vs Inline Commands

Interactive commands



```
$ ./wsadmin.sh
WASX7209I: Connected to process "server1" on node localhost using SOAP connect
or; The type of process is: UnManagedProcess
WASX7029I: For help, enter: "$Help help"
wsadmin>$AdminControl stopServer server1
WASX7337I: Invoked stop for server "server1" Waiting for stop completion.
WASX7264I: Stop completed for server "server1" on node "localhost"
wsadmin>|
```

Enter command

Receive output

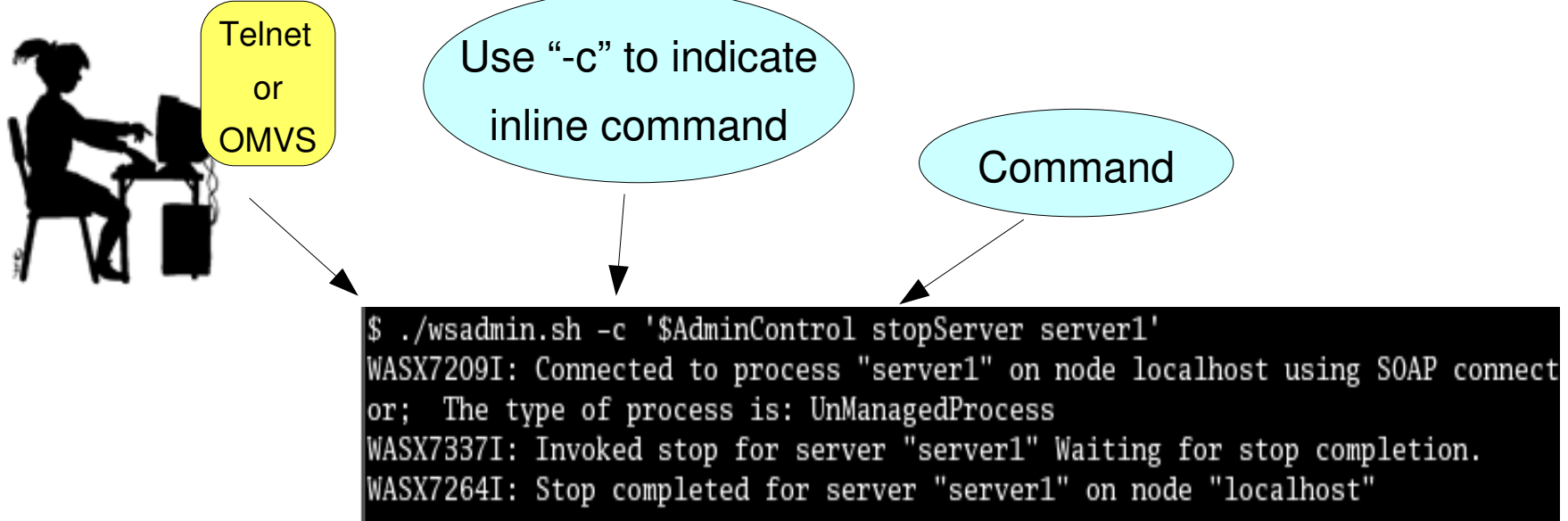
Enter command

Useful for quick administrative tasks

- stop server
- install / uninstall applications

Interactive vs Inline Commands

Inline commands



BBOWIAPP job uses inline command to install the administrative console

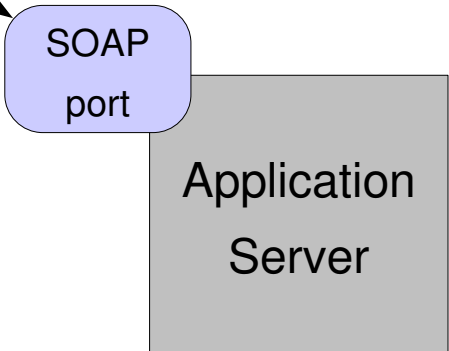
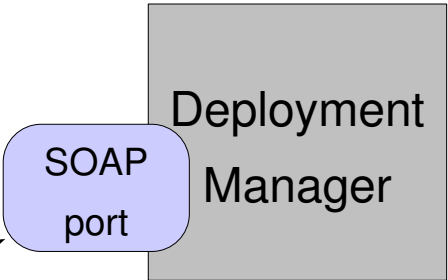
Using wsadmin From Your Workstation

wsadmin.bat – Windows

wsadmin.sh – Unix and Linux



```
wsadmin.sh -host <host> -port <port>
```



Is global security turned on?

No

Yes

No worries

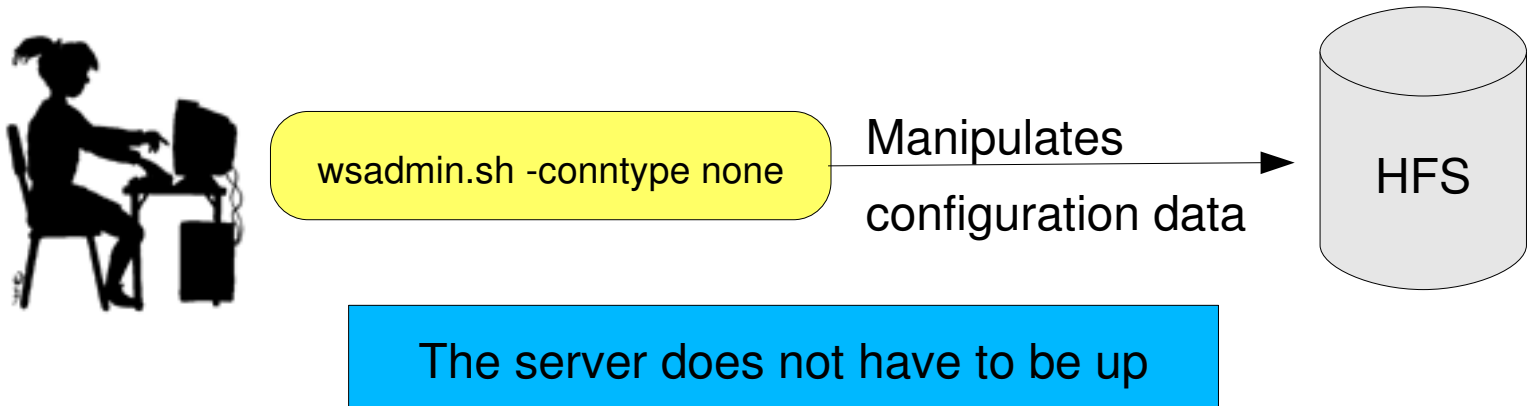
CA certificate must be on client's keyring

Provide userid and password

Can We Use WSADMIN If the Server is Down?

It depends!

wsadmin in local mode



Obviously, this does not work from your workstation...

Can We Use WSADMIN if the Server is Down?

It depends!

wsadmin in remote mode



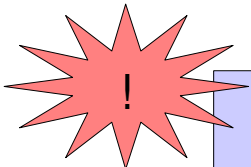
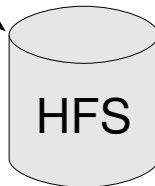
wsadmin.sh -conntype soap

wsadmin.sh -conntype rmi

Connects to server



Manipulates configuration data



It is not recommended to use remote mode and the admin console simultaneously

Using WSADMIN With Global Security On



```
wsadmin.sh -conntype soap -user <user> -password <pw>
```



```
soap.client.props  
securityEnabled=true
```

```
sas.client.props  
securityEnabled=true
```

Import server's CA certificate onto client keyring using ikeyman.(sh|bat)

Should not be necessary if executing wsadmin from USS

Is This a Security Exposure?



```
wsadmin.sh -conntype soap -user <user> -password <pw>
```

Hmm.. I could use "ps" to display running processes



Grr...

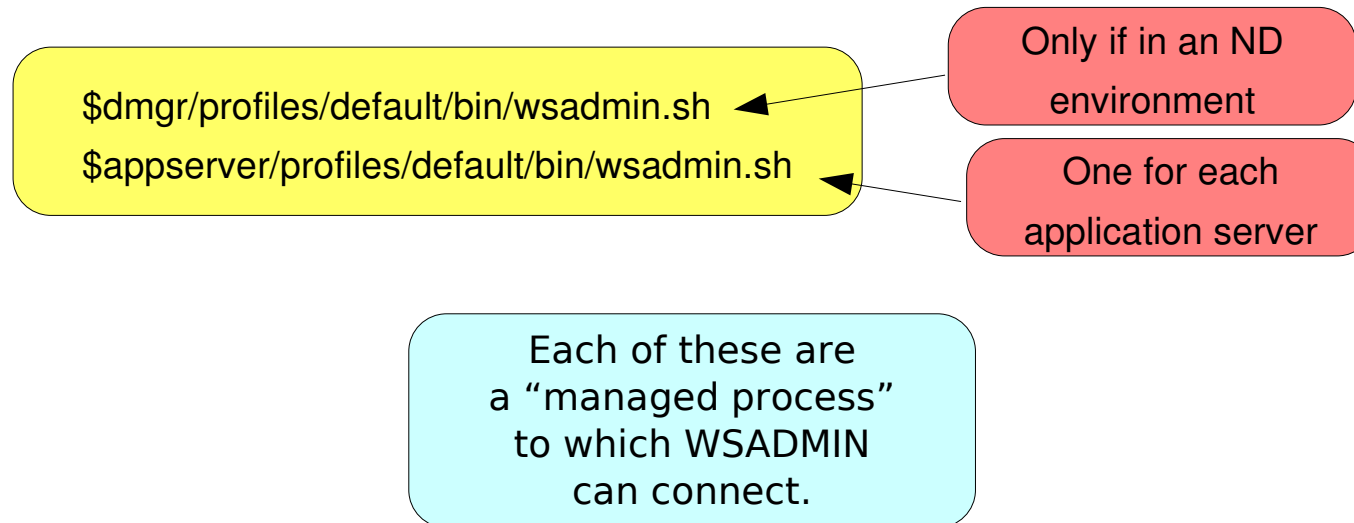
Disgruntled employee

Specify userid and password

soap.client.props

sas.client.props

Are There Multiple WSADMINs?



Using WSADMIN – Objects

WSADMIN uses four objects:

AdminApp
manipulate applications

AdminControl
manipulate “objects”
(e.g., servers)

AdminConfig
manipulate
configuration data

Help
self-explanatory

What Can I Do With the AdminApp Object?

install

uninstall

edit

list

AdminApp help listing:

```

deleteUserAndGroupEntries
    Deletes all the user/group information for all
    the roles and all the username/password information for RunAs
    roles for a given application.
edit
    Edit the properties of an application
editInteractive
    Edit the properties of an application interactively
export
    Export application to a file
exportDDL
    Export DDL from application to a directory
help
    Show help information

install
    Installs an application, given a file name and an option string.

installInteractive
    Installs an application in interactive mode, given a
    file name and an option string.
isAppReady
    Checks whether the application is ready to be run
list
    List all installed applications
listModules
    List the modules in a specified application
options
    Shows the options available, for a given file, application,
    or in general.
publishWSDL
    Publish WSDL files for a given application
searchJNDIReferences
    List application that refer to the given JNDIName on a given node
taskInfo
    Shows detailed information pertaining to a given install task
    for a given file
uninstall
    Uninstalls an application, given an application name and
    an option string
update
    Updates an installed application
updateAccessIDs
    Updates the user/group binding information with accessID
    from user registry for a given application
updateInteractive
    Updates an installed application interactively
view
    View an application or module,
    given an application or module name
  
```

AdminApp – Installing an Application

`$AdminApp install <filename> <options>`

Q: What options?

A: `$AdminApp options`

Not all options are required
for installation

```
wsadmin>$AdminApp options
WASX7105I: The following install options are valid for any ear file:
preCompileJSPs
nopreCompileJSPs
distributeApp
nodistributeApp
useMetaDataFromBinary
nouseMetaDataFromBinary
deployejb
nodeployejb
createMBeansForResources
nocreateMBeansForResources
reloadEnabled
noreloadEnabled
deployws
nodeployws
processEmbeddedConfig
noprocessEmbeddedConfig
usedefaultbindings
defaultbinding.force
allowPermInFilterPolicy
noallowPermInFilterPolicy
verbose
update
update.ignore.old
update.ignore.new
installed.ear.destination
appname
reloadInterval
validateinstall
deployejb.rmic
deployejb.dbtype
deployejb.dbschema
deployejb.classpath
deployws.classpath
deployws.jardirs
defaultbinding.datasource.jndi
defaultbinding.datasource.username
defaultbinding.datasource.password
defaultbinding.cf.jndi
defaultbinding.cf.resauth
defaultbinding.ejbjndi.prefix
defaultbinding.virtual.host
defaultbinding.strategy.file
server
node
cell
cluster
contextroot
custom
```

Remember to save and
synchronize after installation:
Details to come...

AdminApp – Installation Options

\$AdminApp options <earfile>
Use this to find options specific to the ear file

What is the syntax for these options?

\$AdminApp taskInfo <earfile> MapModulesToServers

Works

\$AdminApp taskInfo <earfile> <option>
This works for only for some options.

Doesn't work

\$AdminApp taskInfo <earfile> preCompileJSPs



AdminApp – Installation Options

\$AdminApp taskInfo <earfile> MapModulesToServers

```
wsadmin>$AdminApp taskInfo /u/tai/SuperSnoop.ear MapModulesToServers
```

MapModulesToServers: Selecting servers

Specify targets such as application servers or clusters of application servers where you want to install the modules contained in your application. Modules can be installed on the same application server or dispersed among several application servers. Also, specify the Web servers as targets that will serve as routers for requests to this application. The plug-in configuration file (plugin-cfg.xml) for each Web server is generated based on the applications which are routed through it.

WASX7348I: Each element of the MapModulesToServers task consists of the following 3 fields: "Module", "URI", "Server".

Of these fields, the following may be assigned values: "Server"

and the following are required: "Server"

The current contents of the task after running default bindings are:

Module: SuperSnoopWeb

URI: SuperSnoopWeb.war,WEB-INF/web.xml

Server: WebSphere:cell=wsc1.washington.ibm.com,node=wsc1.washington.ibm.com,server=server1

AdminApp – Uninstalling an Application

Wow, that was easy!



```
$AdminApp uninstall myapp
```

You must use the application name,
not the ear file name

Saving is not necessary
after uninstallation

```
ADMA5106I: Application myapp uninstalled successfully.
```

AdminConfig

Used to create, remove, or modify “config objects” - virtual hosts, servers, etc.
 Let’s take a look at the WebContainer object

\$AdminConfig attributes WebContainer
 Use this to find a list of all “types”

\$AdminConfig types
 Use this to find a list of all “types”

- .
- .
- WSGWInstance
- WSGWProxyService
- WSGWTARGETService
- WebContainer
- WebContainerInboundChannel
- WebModuleConfig
- WebModuleDeployment
- WebModuleRef
- .
- .

- .
- .
- "defaultVirtualHostName String"
- "enableServletCaching boolean"
- .
- .

General Properties

Default virtual host:

Enable servlet caching

AdminConfig – Which WebContainer?

\$AdminConfig show <object>

Show attributes of a given config object

```
wsadmin>$AdminConfig show WebContainer
```

```
WASX7015E: Exception running command: "$AdminConfig show WebContainer"; exception information:
com.ibm.ws.scripting.ScriptingException: WASX7077E: Incomplete config id: need closing parenthesis in "WebContainer"
```

\$AdminConfig list <object>

Lists all config objects of a given type

```
wsadmin>$AdminConfig list WebContainer
```

```
(cells/h6cell/nodes/h6dmgrnd/servers/dmgr|server.xml#WebContainer_1)
(cells/h6cell/nodes/h6nodea/servers/h6sr01a|server.xml#WebContainer_1112019030113)
(cells/h6cell/nodes/h6nodea/servers/h6sr01b|server.xml#WebContainer_1118253253830)
(cells/h6cell/nodes/h6nodea/servers/h6sr02a|server.xml#WebContainer_1112020441556)
(cells/h6cell/nodes/h6nodea/servers/h6sr03a|server.xml#WebContainer_1112139804785)
```

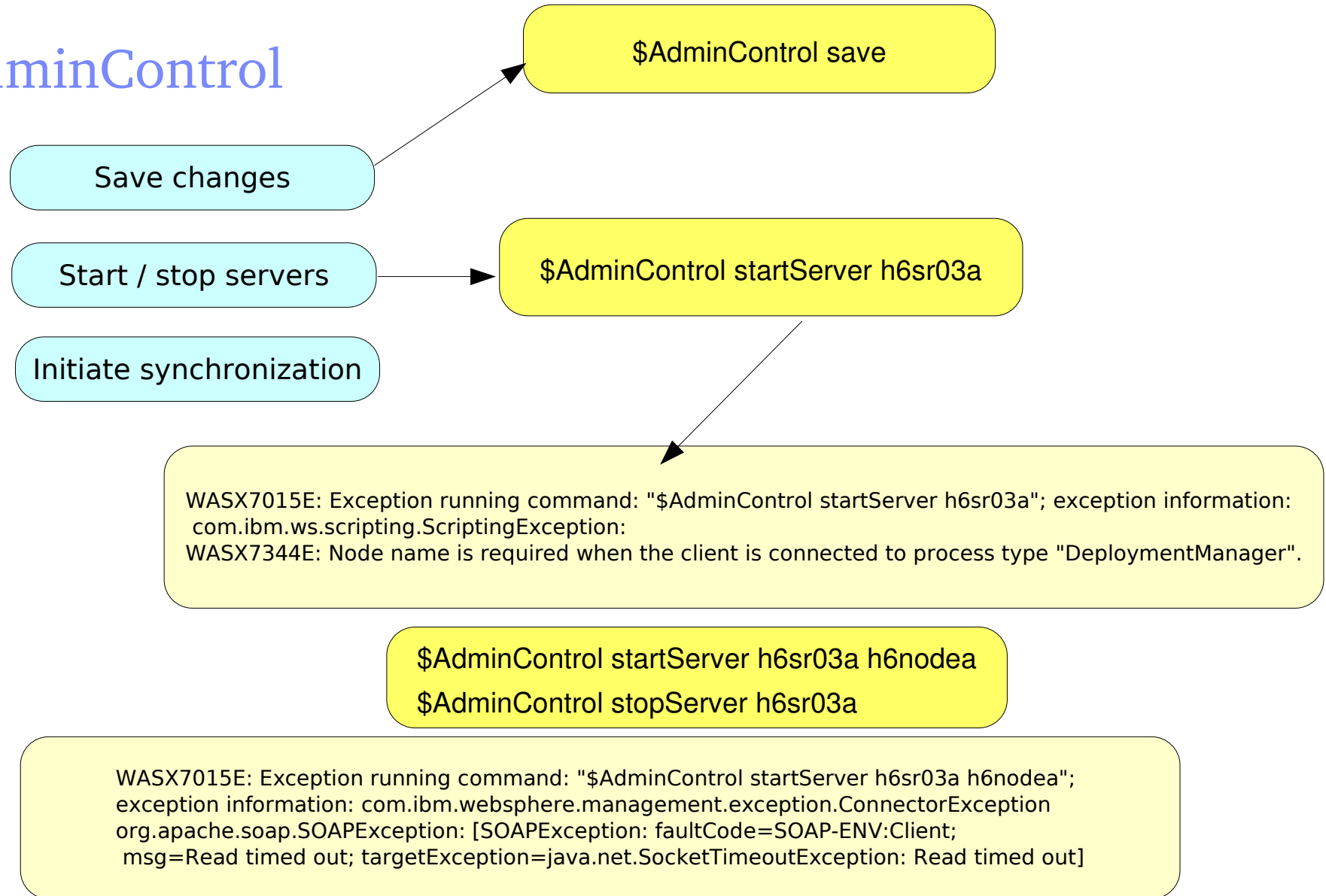
\$AdminConfig modify <object> <attributes>

Change specified attributes of a given config object

```
wsadmin>$AdminConfig modify
```

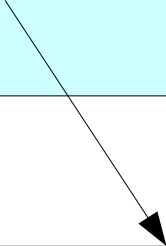
```
(cells/h6cell/nodes/h6nodea/servers/h6sr03a|server.xml#WebContainer_1112139804785)
{{ enableServletCaching true }}
```

AdminControl



AdminControl – Node Synchronization

```
set var [$AdminControl completeObjectName type=NodeSync,node=h6nodea,*]  
$AdminControl invoke $var sync
```



```
WebSphere:platform=common,cell=h6cell,version=6.0.2.1,name=nodeSync,  
mbeanIdentifier=nodeSync,type=NodeSync,node=h6nodea,process=nodeagent
```

This must be done for each node that you want synchronized.

External Scripts – Jacl and Jython

Benefits of using an external script

- passing in parameters
- control flow
- error checking and handling



WAS expects scripts to be in ASCII

- Use script.encoding property for WSADMIN
- Use ASCII file editor on z/OS
- Edit on your PC and FTP to HFS

Use -f option to tell WSADMIN to use an external script:

```
$ ./wsadmin.sh -f install.jacl
```

Jacl

Implementation of the Tcl interpreter written in Java

Default scripting language expected by WAS



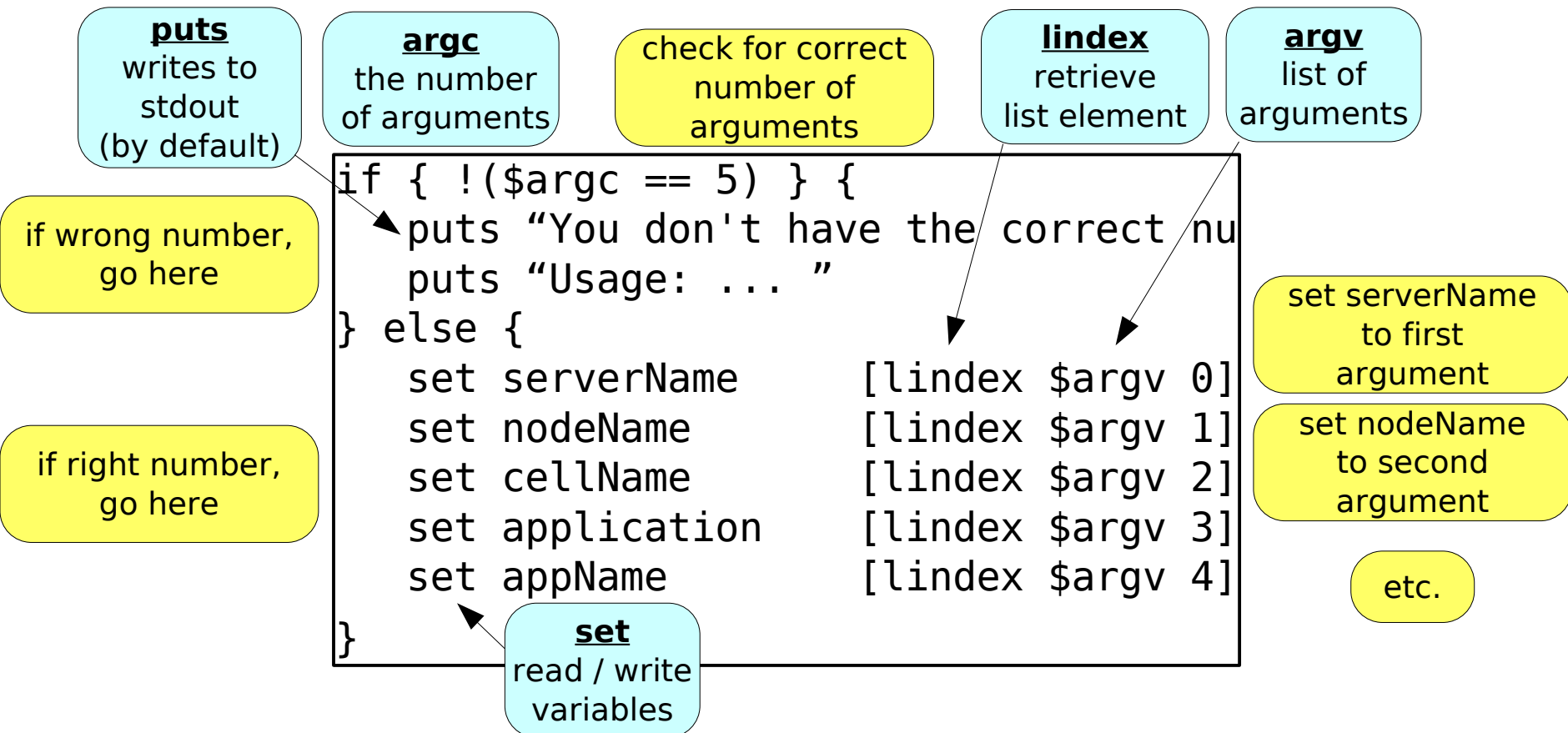
Jython

Implementation of Python written in Java

Command Line Parameters – Jacl

Taking parameters from the command line

- Use this script for multiple servers, nodes, applications, etc.



Command Line Parameters – Jython



print
writes to
stdout

len
number of
items in a
list

sys.argv
list of
arguments

check for correct
number of
arguments

if wrong number,
go here

if right number,
go here

```

if (len(sys.argv) != 5):
    print "You don't have the correct nu
    print "Usage: ... "
} else {
    serverName = sys.argv[0]
    nodeName = sys.argv[1]
    cellName = sys.argv[2]
    application = sys.argv[3]
    appName = sys.argv[4]
}
    
```

set serverName
to first
argument

set nodeName
to second
argument

etc.

getopt
not
available

Verifying the Node Exists

length
returns number of items in a list

```
set node [$AdminConfig getid /Node:$nodeName/]
if {[length $node] == 0} {
    puts "Error -- node not found for name $nodeName"
    return
}
```

Jacl

print error message if node is not found

getid
returns configid of an object

check length of result

nodeName passed in as an argument

```
node = AdminConfig.getid("/Node:" + nodeName + "/")
print "ex1: checking for existence of node " + nodeName
if len(node) == 0:
    print "Error -- node not found for name " + nodeName
    return
```

Jython

print error message if node is not found



Verifying the Server Exists

Determine if the given server is available

```
set runningServer [$AdminControl completeObjectName type=server,node=$nodeName,process=$serverName,*]
if {[llength $runningServer] == 0 } {
  puts "Error: Server not running..."
  return
}
```

Jacl

completeObjectName
returns object name

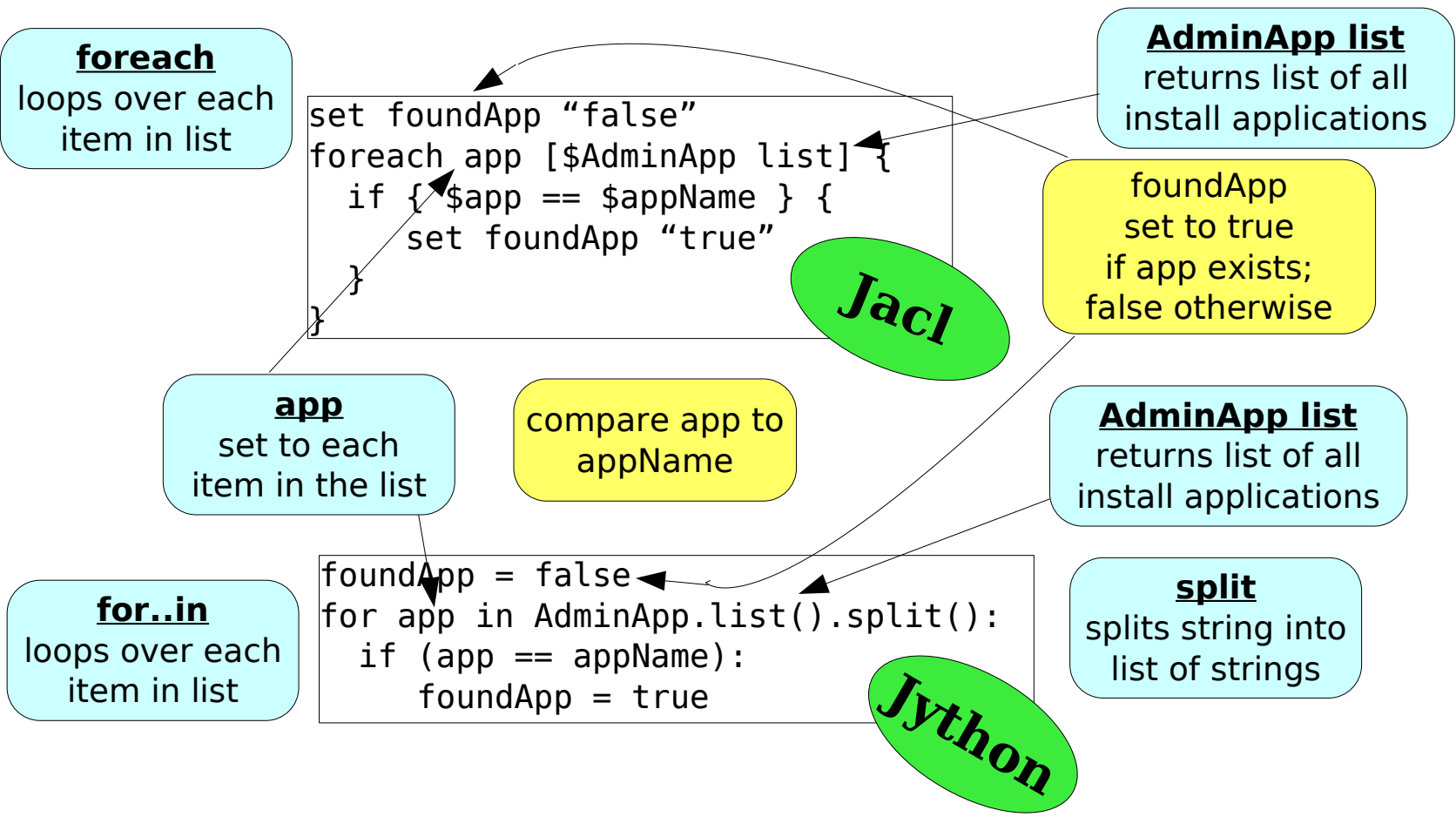
serverName
passed in
as an argument

```
runningServer = AdminControl.completeObjectName("type=Server,node="+nodeName+",process="+serverName+",*")
if len(runningServer) == 0:
  puts "Error: Server not running..."
  return
```

Jython

Check Existing Applications

Check to see if the application is already on the server...



Jacl

Installing and Removing an Application

```
$AdminApp uninstall $appName
```

```
set options [list -server $serverName]
lappend options -node $nodeName -appname $appName
$AdminApp install $app $options
$AdminConfig save
```

uninstall
self-explanatory

appName
passed in
as an argument

lappend
append list
elements

install
self-explanatory

app
ear file we are
deploying

save
commit unsaved
changes

```
AdminApp.uninstall(appName)
```

```
options = "[-server " + serverName
options = options + " -node " + nodeName
options = options + " -appname " + appName + "]"
AdminApp.install(app,options)
AdminConfig.save()
```

Jython

Creating a New Server - Jacl

```
set name [list name $serverName]
set server [$AdminConfig create Server $nodeName name $name]
```

nodeName
passed in as
an argument

create Server
self-explanatory

Nested Options

- some options have their own options
- e.g., address option include host and port options
- use nested lists or concatenation

set http and
https transports

```
set hostlist [list host {}]
set portlist [list port 9080]
set addresslist [list $hostlist $portlist]
```

```
set http [list [list sslEnabled false] [list address [list [list host {}] [list port 9080]]]]
set https [list [list sslEnabled true] [list address [list [list host {}] [list port 9081]]]
  [list sslConfig DefaultSSLSettings]]
set transports [list transports:HTTPTransport [list $http $https]]
set webcontainer [$AdminConfig list WebContainer $server]
$AdminConfig modify $webcontainer [list $transports]
```

modify
change specified attributes

set http and
https ports

Creating a New Server – Jython

```
name = ["name", serverName]
server = AdminConfig.create("Server", node, name)
```

Nested Options

- some options have their own options
- e.g., address option include host and port options
- use nested lists or concatenation

```
hostlist = ["host", ""]
portlist = ["port", 9080]
addresslist = [hostlist, portlist]
```

```
http = ["sslEnabled", "false"], ["address", [ ["host", ""], ["port", 9080] ] ]
https = ["sslEnabled", "true"], ["address", [ ["host", ""], ["port", 9081] ] ], ["sslConfig",
    "DefaultSSLSettings"]
transports = ["transports:HTTPTransport", [http, https]]
webcontainer = AdminConfig.list("WebContainer", server)
AdminConfig.modify(webcontainer, transports)
```

Synchronization Of All Nodes

```
set node_ids [$AdminConfig list Node]
foreach node $node_ids {
  set nodename [$AdminConfig showAttribute $node name]
  set nodesync [$AdminControl completeObjectName type=NodeSync,node=$nodename,*]
  if { !($nodesync=="") } then {
    $AdminControl invoke $nodesync sync
  }
}
```

list

get a list of all nodes

Jacl

Check for empty string
because we don't to sync
the dmgr's node

Use for loop to
cycle through all
nodes

list

get a list of all nodes

split

split list into elements

```
node_ids = AdminConfig.list("Node").split()
for node in node_ids:
  nodename = AdminConfig.showAttribute(node,"name")
  nodesync = AdminControl.completeObjectName("type=NodeSync,node=" + nodename + ",*")
  if nodesync != "":
    AdminControl.invoke(nodesync,"sync")
```

Jython

Questions?

